

Lecture Notes in Computer Science

Edited by G. Goos, J. Hartmanis, and J. van Leeuwen

2723

Springer

Berlin

Heidelberg

New York

Hong Kong

London

Milan

Paris

Tokyo

Erick Cantú-Paz James A. Foster
Kalyanmoy Deb Lawrence David Davis
Rajkumar Roy Una-May O'Reilly
Hans-Georg Beyer Russell Standish
Graham Kendall Stewart Wilson
Mark Harman Joachim Wegener
Dipankar Dasgupta Mitch A. Potter
Alan C. Schultz Kathryn A. Dowsland
Natasha Jonoska Julian Miller (Eds.)

Genetic and Evolutionary Computation – GECCO 2003

Genetic and Evolutionary Computation Conference
Chicago, IL, USA, July 12-16, 2003
Proceedings, Part I



Springer

Series Editors

Gerhard Goos, Karlsruhe University, Germany
Juris Hartmanis, Cornell University, NY, USA
Jan van Leeuwen, Utrecht University, The Netherlands

Main Editor

Erick Cantú-Paz
Center for Applied Scientific Computing (CASC)
Lawrence Livermore National Laboratory
7000 East Avenue, L-561, Livermore, CA 94550, USA
E-mail: cantupaz@llnl.gov

Cataloging-in-Publication Data applied for

A catalog record for this book is available from the Library of Congress

Bibliographic information published by Die Deutsche Bibliothek
Die Deutsche Bibliothek lists this publication in the Deutsche Nationalbibliografie;
detailed bibliographic data is available in the Internet at <<http://dnb.ddb.de>>.

CR Subject Classification (1998): F.1-2, D.1.3, C.1.2, I.2.6, I.2.8, I.2.11, J.3

ISSN 0302-9743

ISBN 3-540-40602-6 Springer-Verlag Berlin Heidelberg New York

This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, re-use of illustrations, recitation, broadcasting, reproduction on microfilms or in any other way, and storage in data banks. Duplication of this publication or parts thereof is permitted only under the provisions of the German Copyright Law of September 9, 1965, in its current version, and permission for use must always be obtained from Springer-Verlag. Violations are liable for prosecution under the German Copyright Law.

Springer-Verlag Berlin Heidelberg New York
a member of BertelsmannSpringer Science+Business Media GmbH

<http://www.springer.de>

© Springer-Verlag Berlin Heidelberg 2003
Printed in Germany

Typesetting: Camera-ready by author, data conversion by PTP Berlin GmbH
Printed on acid-free paper SPIN 10928998 06/3142 5 4 3 2 1 0

Preface

These proceedings contain the papers presented at the 5th Annual Genetic and Evolutionary Computation Conference (GECCO 2003). The conference was held in Chicago, USA, July 12–16, 2003.

A total of 417 papers were submitted to GECCO 2003. After a rigorous doubleblind reviewing process, 194 papers were accepted for full publication and oral presentation at the conference, resulting in an acceptance rate of 46.5%. An additional 92 submissions were accepted as posters with two-page extended abstracts included in these proceedings.

This edition of GECCO was the union of the 8th Annual Genetic Programming Conference (which has met annually since 1996) and the 12th International Conference on Genetic Algorithms (which, with its first meeting in 1985, is the longest running conference in the field). Since 1999, these conferences have merged to produce a single large meeting that welcomes an increasingly wide array of topics related to genetic and evolutionary computation.

Possibly the most visible innovation in GECCO 2003 was the publication of the proceedings with Springer-Verlag as part of their Lecture Notes in Computer Science series. This will make the proceedings available in many libraries as well as online, widening the dissemination of the research presented at the conference.

Other innovations included a new track on Coevolution and Artificial Immune Systems and the expansion of the DNA and Molecular Computing track to include quantum computation.

In addition to the presentation of the papers contained in these proceedings, the conference included 13 workshops, 32 tutorials by leading specialists, and presentation of late-breaking papers.

GECCO is sponsored by the International Society for Genetic and Evolutionary Computation (ISGEC). The ISGEC by-laws contain explicit guidance on the organization of the conference, including the following principles:

(i) GECCO should be a broad-based conference encompassing the whole field of genetic and evolutionary computation.

(ii) Papers will be published and presented as part of the main conference proceedings only after being peer-reviewed. No invited papers shall be published (except for those of up to three invited plenary speakers).

(iii) The peer-review process shall be conducted consistently with the principle of division of powers performed by a multiplicity of independent program committees, each with expertise in the area of the paper being reviewed.

(iv) The determination of the policy for the peer-review process for each of the conference's independent program committees and the reviewing of papers for each program committee shall be performed by persons who occupy their positions by virtue of meeting objective and explicitly stated qualifications based on their previous research activity.

(v) Emerging areas within the field of genetic and evolutionary computation shall be actively encouraged and incorporated in the activities of the conference by providing a semiautomatic method for their inclusion (with some procedural flexibility extended to such emerging new areas).

(vi) The percentage of submitted papers that are accepted as regular full-length papers (i.e., not posters) shall not exceed 50%.

These principles help ensure that GECCO maintains high quality across the diverse range of topics it includes.

Besides sponsoring the conference, ISGEC supports the field in other ways. ISGEC sponsors the biennial Foundations of Genetic Algorithms workshop on theoretical aspects of all evolutionary algorithms. The journals *Evolutionary Computation* and *Genetic Programming and Evolvable Machines* are also supported by ISGEC. All ISGEC members (including students) receive subscriptions to these journals as part of their membership. ISGEC membership also includes discounts on GECCO and FOGA registration rates as well as discounts on other journals. More details on ISGEC can be found online at <http://www.isgec.org>.

Many people volunteered their time and energy to make this conference a success. The following people in particular deserve the gratitude of the entire community for their outstanding contributions to GECCO:

James A. Foster, the General Chair of GECCO for his tireless efforts in organizing every aspect of the conference.

David E. Goldberg and John Koza, members of the Business Committee, for their guidance and financial oversight.

Alwyn Barry, for coordinating the workshops.

Bart Rylander, for editing the late-breaking papers.

Past conference organizers, William B. Langdon, Erik Goodman, and Darrell Whitley, for their advice.

Elizabeth Ericson, Carol Hamilton, Ann Stolberg, and the rest of the AAAI staff for their outstanding efforts administering the conference.

Gerardo Valencia and Gabriela Coronado, for Web programming and design.

Jennifer Ballentine, Lee Ballentine and the staff of Professional Book Center, for assisting in the production of the proceedings.

Alfred Hofmann and Ursula Barth of Springer-Verlag for helping to ease the transition to a new publisher.

Sponsors who made generous contributions to support student travel grants:

Air Force Office of Scientific Research

DaimlerChrysler

National Science Foundation

Naval Research Laboratory

New Light Industries

Philips Research

Sun Microsystems

The track chairs deserve special thanks. Their efforts in recruiting program committees, assigning papers to reviewers, and making difficult acceptance decisions in relatively short times, were critical to the success of the conference:

A-Life, Adaptive Behavior, Agents, and Ant Colony Optimization,
 Russell Standish
 Artificial Immune Systems, Dipankar Dasgupta
 Coevolution, Graham Kendall
 DNA, Molecular, and Quantum Computing, Natasha Jonoska
 Evolution Strategies, Evolutionary Programming, Hans-Georg Beyer
 Evolutionary Robotics, Alan Schultz, Mitch Potter
 Evolutionary Scheduling and Routing, Kathryn A. Dowsland
 Evolvable Hardware, Julian Miller
 Genetic Algorithms, Kalyanmoy Deb
 Genetic Programming, Una-May O'Reilly
 Learning Classifier Systems, Stewart Wilson
 Real-World Applications, David Davis, Rajkumar Roy
 Search-Based Software Engineering, Mark Harman, Joachim Wegener

The conference was held in cooperation and/or affiliation with:

American Association for Artificial Intelligence (AAAI)
 Evonet: the Network of Excellence in Evolutionary Computation
 5th NASA/DoD Workshop on Evolvable Hardware
Evolutionary Computation
Genetic Programming and Evolvable Machines
Journal of Scheduling
Journal of Hydroinformatics
Applied Soft Computing

Of course, special thanks are due to the numerous researchers who submitted their best work to GECCO, reviewed the work of others, presented a tutorial, organized a workshop, or volunteered their time in any other way. I am sure you will be proud of the results of your efforts.

May 2003

Erick Cantú-Paz
 Editor-in-Chief GECCO 2003
 Center for Applied Scientific Computing
 Lawrence Livermore National Laboratory

GECCO 2003 Conference Organization

Conference Committee

General Chair: James A. Foster

Proceedings Editor-in-Chief: Erick Cantú-Paz

Business Committee: David E. Goldberg, John Koza, J.A. Foster

Chairs of Program Policy Committees:

A-Life, Adaptive Behavior, Agents, and Ant Colony Optimization,
Russell Standish

Artificial Immune Systems, Dipankar Dasgupta
Coevolution, Graham Kendall

DNA, Molecular, and Quantum Computing, Natasha Jonoska

Evolution Strategies, Evolutionary Programming, Hans-Georg Beyer

Evolutionary Robotics, Mitchell A. Potter and Alan C. Schultz

Evolutionary Scheduling and Routing, Kathryn A. Dowsland

Evolvable Hardware, Julian Miller

Genetic Algorithms, Kalyanmoy Deb

Genetic Programming, Una-May O'Reilly

Learning Classifier Systems, Stewart Wilson

Real-World Applications, David Davis, Rajkumar Roy

Search-Based Software Engineering, Mark Harman and Joachim Wegener

Workshops Chair: Alwyn Barry

Late-Breaking Papers Chair: Bart Rylander

Workshop Organizers

Biological Applications for Genetic and Evolutionary Computation (Bio GEC 2003), Wolfgang Banzhaf, James A. Foster

Application of Hybrid Evolutionary Algorithms to NP-Complete Problems, Francisco Baptista Pereira, Ernesto Costa, Günther Raidl

Evolutionary Algorithms for Dynamic Optimization Problems, Jürgen Branke

Hardware Evolutionary Algorithms and Evolvable Hardware (HEAEH 2003), John C. Gallagher

Graduate Student Workshop, Maarten Keijzer, Sean Luke, Terry Riopka

Workshop on Memetic Algorithms 2003 (WOMA-IV), Peter Merz, William E. Hart, Natalio Krasnogor, Jim E. Smith

Undergraduate Student Workshop, Mark M. Meysenburg

Learning, Adaptation, and Approximation in Evolutionary Computation, Sibylle Mueller, Petros Koumoutsakos, Marc Schoenauer, Yaochu Jin, Sushil Louis, Khaled Rasheed

Grammatical Evolution Workshop (GEWS 2003), Michael O'Neill, Conor Ryan

Interactive Evolutionary Search and Exploration Systems, Ian Parmee

Analysis and Design of Representations and Operators (ADoRo 2003), Franz Rothlauf, Dirk Thierens

Challenges in Real-World Optimisation Using Evolutionary Computing, Rajkumar Roy, Ashutosh Tiwari

International Workshop on Learning Classifier Systems, Wolfgang Stolzmann, Pier-Luca Lanzi, Stewart Wilson

Tutorial Speakers

Parallel Genetic Algorithms, Erick Cantú-Paz

Using Appropriate Statistics, Steffan Christensen

Multiobjective Optimization with EC, Carlos Coello

Making a Living with EC, Yuval Davidor

A Unified Approach to EC, Ken DeJong

Evolutionary Robotics, Dario Floreano

Immune System Computing, Stephanie Forrest

The Design of Innovation & Competent GAs, David E. Goldberg

Genetic Algorithms, Robert Heckendorn

Evolvable Hardware Applications, Tetsuya Higuchi

Bioinformatics with EC, Daniel Howard

Visualization in Evolutionary Computation, Christian Jacob

Data Mining and Machine Learning, Hillol Kargupta

Evolvable Hardware, Didier Keymeulen

Genetic Programming, John Koza

Genetic Programming Theory I & II, William B. Langdon, Riccardo Poli

Ant Colony Optimization, Martin Middendorf

Bionics: Building on Biological Evolution, Ingo Rechenberg

Grammatical Evolution, C. Ryan, M. O'Neill

Evolution Strategies, Hans-Paul Schwefel

Quantum Computing, Lee Spector

Anticipatory Classifier Systems, Wolfgang Stolzmann

Mathematical Theory of EC, Michael Vose

Computational Complexity and EC, Ingo Wegener

Software Testing via EC, J. Wegener, M. Harman

Testing & Evaluating EC Algorithms, Darrell Whitley

Learning Classifier Systems, Stewart Wilson

Evolving Neural Network Ensembles, Xin Yao

Neutral Evolution in EC, Tina Yu

Genetics, Annie S. Wu

Keynote Speakers

John Holland, "The Future of Genetic Algorithms"

Richard Lenski, "How the Digital Leopard Got His Spots: Thinking About Evolution Inside the Box"

Members of the Program Committee

Hussein Abbass	Stefano Cagnoni	Stefan Droste
Adam Adamopoulos	Xiaoqiang Cai	Marc Ebner
Alexandru Agapie	Erick Cantú-Paz	R. Timothy Edwards
José Aguilar	Uday Chakraborty	Norberto Eiji Nawa
Jesús Aguilar	Weng-Tat Chan	Aniko Ekart
Hernán Aguirre	Alastair Channon	Christos Emmanouilidis
Chang Wook Ahn	Ying-Ping Chen	Hector Erives
Uwe Aickelin	Shu-Heng Chen	Felipe Espinoza
Enrique Alba	Junghuei Chen	Matthew Evett
Javier Alcaraz Soria	Prabhas Chongstitvatana	Zhun Fan
Dirk Arnold	John Clark	Marco Farina
Tughrul Arslan	Lattaud Claude	Robert Feldt
Atif Azad	Manuel Clergue	Francisco Fernández
Meghna Babbar	Carlos Coello Coello	Sevan Ficici
Vladan Babovic	David Coley	Peter John Fleming
B.V. Babu	Philippe Collard	Stuart Flockton
Thomas Bäck	Pierre Collet	Dario Floreano
Julio Banga	Clare Bates Congdon	Cyril Fonlupt
Francisco Baptista Pereira	David Corne	Carlos Fonseca
Alwyn Barry	Ernesto Costa	Stephanie Forrest
Cem Baydar	Peter Cowling	Alex Freitas
Thomas Beielstein	Bart Craenen	Clemens Frey
Theodore Belding	Jose Cristóbal Riquelme	Chunsheng Fu
Fevzi Belli	Santos	Christian Gagne
Ester Bernado-Mansilla	Keshav Dahal	M. Gargano
Tom Bersano-Begey	Paul Darwen	Ivan Garibay
Hugues Bersini	Dipankar Dasgupta	Josep Maria Garrell i Guieu
Hans-Georg Beyer	Lawrence Davis	Alessio Gaspar
Filipic Bogdan	Anthony Deakin	Michel Gendreau
Andrea Bonarini	Kalyanmoy Deb	Zhou Gengui
Lashon Booker	Ivanoe De Falco	Pierre Gérard
Peter Bosman	Hugo De Garis	Andreas Geyer-Schulz
Terry Bossomaier	Antonio Della Cioppa	Tushar Goel
Klaus Bothe	A. Santos Del Riego	Fabio Gonzalez
Leonardo Bottaci	Brahma Deo	Jens Gottlieb
Jürgen Branke	Dirk Devogelaere	Kendall Graham
Wilker Bruce	Der-Rong Din	Buster Greene
Peter Brucker	Phillip Dixon	John Grefenstette
Anthony Bucci	Jose Dolado Cosin	Darko Grundler
Dirk Bueche	Marco Dorigo	Dongbing Gu
Magdalena Bugajska	Keith Downing	Steven Gustafson
Larry Bull	Kathryn Dowsland	Charles Guthrie
Edmund Burke	Gerry Dozier	Pauline Haddow
Martin Butz	Rolf Drechsler	Hani Hagras

Hisashi Handa	Kalmanje Krishnakumar	Martin Middendorf
Georges Harik	Renato Krohling	Risto Miikkulainen
Mark Harman	Sam Kwong	Julian Miller
Emma Hart	Gary Lamont	Brian Mitchell
William Hart	William Langdon	Chilukuri Mohan
Inman Harvey	Pedro Larrañaga	David Montana
Michael Herdy	Jesper Larse	Byung-Ro Moon
Jeffrey Hermann	Marco Laumanns	Frank Moore
Arturo Hernández	Paul Layzell	Alberto Moraglio
Aguirre	Martin Lefley	Manuel Moreno
Francisco Herrera	Claude Le Pape	Yunjun Mu
Jürgen Hesser	Kwong Sak Leung	Sibylle Mueller
Robert Hierons	Warren Liao	Masaharu Munetomo
Mika Hirvensalo	Derek Linden	Kazuyuki Murase
John Holmes	Michael Littman	William Mydlowec
Tadashi Horiuchi	Xavier Llorà	Zensho Nakao
Daniel Howard	Fernando Lobo	Tomoharu Nakashima
William Hsu	Jason Lohn	Olfa Nasraoui
Jianjun Hu	Michael Lones	Bart Naudts
Jacob Hurst	Sushil Louis	Mark Neal
Hitoshi Iba	Manuel Lozano	Chrystopher Nehaniv
Kosuke Imamura	Jose Antonio Lozano	David Newth
Iñaki Inza	Jose Lozano	Miguel Nicolau
Christian Jacob	Pier Luca Lanzi Sean Luke	Nikolay Nikolaev
Thomas Jansen	John Lusth	Fernando Nino
Segovia Javier	Evelyne Lutton	Stefano Nolfi
Yaochu Jin	Nicholas Macias	Peter Nordin
Bryan Jones	Ana Madureira	Bryan Norman
Natasha Jonoska	Spiros Mancoridis	Wim Nuijten
Hugues Juille	Martin Martin	Leandro Nunes De Castro
Bryant Julstrom	Pete Martin	Gabriela Ochoa
Mahmoud Kaboudan	Arita Masanori	Victor Oduguwa
Charles Karr	Iwata Masaya	Charles Ofria
Balakrishnan Karthik	Keith Mathias	Gustavo Olague
Sanza Kazadi	Dirk Mattfeld	Markus Olhofer
Maarten Keijzer	Giancarlo Mauri	Michael O'Neill
Graham Kendall	David Mayer	Una-May O'Reilly
Didier Keymeulen	Jon McCormack	Franz Oppacher
Michael Kirley	Robert McKay	Jim Ouimette
Joshua Knowles	Nicholas McPhee	Charles Palmer
Gabriella Kokai	Lisa Meeden	Liviu Panait
Arthur Kordon	Jörn Mehnen	Gary Parker
Bogdan Korel	Karlheinz Meier	Anil Patel
Erkan Korkmaz	Ole Mengshoel	Witold Pedrycz
Tim Kovacs	Mark Meysenburg	Martin Pelikan
Natalio Krasnogor	Zbigniew Michalewicz	Marek Perkowski

Sanja Petrovic
Hartmut Pohlheim
Riccardo Poli
Tom Portegys
Reid Porter
Marie-Claude Portmann
Mitchell A. Potter
Walter Potter
Jean-Yves Potvin
Dilip Pratihari
Alexander Pretschner
Adam Prügel-Bennett
William Punch
Günther Raidl
Khaled Rasheed
Tom Ray
Tapabrata Ray
Victor Rayward-Smith
Patrick Reed
John Reif
Andreas Reinholz
Rick Riolo
Jose Riquelme
Denis Robilliard
Katya Rodriguez-Vazquez
Marc Roper
Brian Ross
Franz Rothlauf
Jon Rowe
Rajkumar Roy
Günter Rudolph
Thomas Runarsson
Conor Ryan
Bart Rylander
Kazuhiro Saitou
Ralf Salomon
Eugene Santos
Kumara Sastry
Yuji Sato
David Schaffer
Martin Schmidt
Thorsten Schnier
Marc Schoenauer
Sonia Schulenburg
Alan C. Schultz

Sandip Sen
Bernhard Sendhoff
Kisung Seo
Franciszek Seredynski
Jane Shaw
Martin Shepperd
Alaa Sheta
Robert Shipman
Olivier Sigaud
Anabela Simões
Mark Sinclair
Abhishek Singh
Andre Skusa
Jim Smith
Robert Smith
Donald Sofge
Alan Soper
Terence Soule
Lee Spector
Andreas Spillner
Russell Standish
Harmen Sthamer
Adrian Stoica
Wolfgang Stolzmann
Matthew Streeter
V. Sundararajan
Gil Syswerda
Walter Tackett
Keiki Takadama
Uwe Tangen
Alexander Tarakanov
Ernesto Tarantino
Gianluca Tempesti
Hugo Terashima-Marin
Sam Thangiah
Scott Thayer
Lothar Thiele
Dirk Thierens
Adrian Thompson
Jonathan Thompson
Jonathan Timmis
Ashutosh Tiwari
Marco Tomassini
Andy Tomlinson
Jim Torresen

Paolo Toth
Michael Trick
Shigeyoshi Tsutsui
Andy Tyrrell
Jano Van Hemert
Clarissa Van Hoyweghen
Leonardo Vanneschi
David Van Veldhuizen
Robert Vanyi
Manuel Vazquez-
 Otomuro
Oswaldo Vélez-Langs
Hans-Michael Voigt
Roger Wainwright
Matthew Wall
Jean-Paul Watson
Ingo Wegener
Joachim Wegener
Karsten Weicker
Peter Whigham
Ronald White
Darrell Whitley
R. Paul Wiegand
Kay Wiese
Dirk Wiesmann
Janet Wile
Janet Wiles
Wendy Williams
Stewart Wilson
Mark Wineberg
Alden Wright
Annie Wu
Zheng Wu
Chia-Hsuan Yeh
Ayse Yilmaz
Tian-Li Yu
Tina Yu
Hongnian Yu
Ricardo Zebulum
Andreas Zell
Byoung-Tak Zhang
Lyudmila A. Zinchenko

A Word from the Chair of ISGEC

You may have just picked up your proceedings, in hard copy and CD-ROM, at GECCO 2003, or purchased it after the conference. You've doubtless already noticed the new format – publishing our proceedings as part of Springer's Lecture Notes in Computer Science (LNCS) series will make them available in many more libraries, broadening the impact of the GECCO conference dramatically!

If you attended GECCO 2003, we, the organizers, hope your experience was memorable and productive, and you have found these proceedings to be of continuing value. The opportunity for first-hand interaction among authors and other participants in GECCO is a big part of what makes it exciting, and we all hope you came away with many new insights and ideas.

If you were unable to come to GECCO 2003 in person, I hope you'll find many stimulating ideas from the world's leading researchers in evolutionary computation reported in these proceedings, and that you'll be able to participate in a future GECCO – for example, next year, in Seattle!

The International Society for Genetic and Evolutionary Computation, the sponsoring organization of the annual GECCO conferences, is a young organization, formed through the merger of the International Society for Genetic Algorithms (sponsor of the ICGA conferences) and the organization responsible for the annual Genetic Programming conferences. It depends strongly on the voluntary efforts of many of its members. It is designed to promote not only the exchange of ideas among innovators and practitioners of well-known methods such as genetic algorithms, genetic programming, evolution strategies, evolutionary programming, learning classifier systems, etc., but also the growth of newer areas such as artificial immune systems, evolvable hardware, agentbased search, and others. One of the founding principles is that ISGEC operates as a confederation of groups with related but distinct approaches and interests, and their mutual prosperity is assured by their representation in the program committees, editorial boards, etc., of the conferences and journals with which ISGEC is associated. This also insures that ISGEC and its functions continue to improve and evolve with the diversity of innovation that has characterized our field.

ISGEC has seen many changes this year, in addition to its growth in membership. We have completed the formalities for recognition as a tax-exempt charitable organization. We have created the new designations of Fellow and Senior Fellow of ISGEC to recognize the achievements of leaders in the field, and by the time you read this, we expect to have elected the first cohort. Additional Fellows and Senior Fellows will be added annually. GECCO continues to be subject to dynamic development – the many new tutorials, workshop topics, and tracks will evolve again next year, seeking to follow and encourage the developments of the many fields represented at GECCO. The best paper awards were presented for the second time at GECCO 2003, and we hope many of you participated in the balloting. This year, for the first time, most presentations at GECCO

were electronic, displayed with the LCD projectors that ISGEC has recently purchased. Our journals, *Evolutionary Computation* and *Genetic Programming and Evolvable Machines*, continue to prosper, and we are exploring ways to make them even more widely available. The inclusion of the proceedings in Springer's Lecture Notes in Computer Science series, making them available in many more libraries worldwide, should have a strong positive impact on our field.

ISGEC is your society, and we urge you to become involved or continue your involvement in its activities, to the mutual benefit of the whole evolutionary computation community. Three members were elected to new five-year terms on the Executive Board at GECCO 2002 – Wolfgang Banzhaf, Marco Dorigo, and Annie Wu.

Since that time, ISGEC has been active on many issues, through actions of the Board and the three Councils – the Council of Authors, Council of Editors, and Council of Conferences.

The organizers of GECCO 2003 are listed in this frontmatter, but special thanks are due to James Foster, General Chair, and Erick Cantú-Paz, Editor-in-Chief of the Proceedings, as well as to John Koza and Dave Goldberg, the Business Committee. All of the changes this year, particularly in the publication of the proceedings, have meant a lot of additional work for this excellent team, and we owe them our thanks for a job well done.

Of course, we all owe a great debt to those who chaired or served on the various core and special program committees that reviewed all of the papers for GECCO 2003. Without their effort it would not have been possible to put on a meeting of this quality.

Another group also deserves the thanks of GECCO participants and ISGEC members – the members of the ISGEC Executive Board and Councils, who are listed below. I am particularly indebted to them for their thoughtful contributions to the organization and their continuing demonstrations of concern for the welfare of ISGEC.

I invite you to communicate with me (goodman@egr.msu.edu) if you have questions or suggestions for ways ISGEC can be of greater service to its members, or if you would like to get more involved in ISGEC and its functions.

Don't forget about the 8th Foundations of Genetic Algorithms (FOGA) workshop, also sponsored by ISGEC, the biennial event that brings together the world's leading theorists on evolutionary computation, which will be held in 2004.

Finally, I hope you will join us at GECCO 2004 in Seattle. Get your ideas to Ricardo Poli, the General Chair of GECCO 2004, when you see him at GECCO 2003, and please check the ISGEC Website, www.isgec.org, regularly for details as the planning for GECCO 2004 continues.

Erik D. Goodman

ISGEC Executive Board

Erik D. Goodman (Chair)

David Andre

Wolfgang Banzhaf

Kalyanmoy Deb

Kenneth De Jong

Marco Dorigo

David E. Goldberg

John H. Holland

John R. Koza

Una-May O'Reilly

Ingo Rechenberg

Marc Schoenauer

Lee Spector

Darrell Whitley

Annie S. Wu

Council of Authors

Erick Cantú-Paz (chair), Lawrence Livermore National Laboratory

David Andre, University of California – Berkeley

Plamen P. Angelov, Loughborough University

Vladan Babovic, Danish Hydraulic Institute

Wolfgang Banzhaf, University of Dortmund

Forrest H. Bennett III, FX Palo Alto Laboratory, Inc.

Hans-Georg Beyer, University of Dortmund

Jergen Branke, University of Karlsruhe

Martin Butz, University of Illinois at Urbana-Champaign

Runwei Cheng, Ashikaga Institute of Technology

David A. Coley, University of Exeter

Marco Dorigo, IRIDIA, Université Libre de Bruxelles

Rolf Drechsler, University of Freiburg

Emanuel Falkenauer, Optimal Design and Brussels University (ULB)

Stephanie Forrest, University of New Mexico

Mitsuo Gen, Ashikaga Institute of Technology

Andreas Geyer-Schulz, Abteilung fuer Informationswirtschaft

David E. Goldberg, University of Illinois at Urbana-Champaign

Jens Gottlieb, SAP, AG

Wolfgang A. Halang, Fernuniversitaet

John H. Holland, University of Michigan and Sante Fe Institute

Hitoshi Iba, University of Tokyo

Christian Jacob, University of Calgary

Robert E. Keller, University of Dortmund

Dimitri Knjazew, SAP, AG

John R. Koza, Stanford University
Sam Kwong, City University of Hong Kong
William B. Langdon, University College, London
Dirk C. Mattfeld, University of Bremen
Pinaki Mazumder, University of Michigan
Zbigniew Michalewicz, University of North Carolina at Charlotte
Melanie Mitchell, Oregon Health and Science University
Ian Parmee, University of North Carolina at Charlotte
Frederick E. Petry, University of North Carolina at Charlotte
Riccardo Poli, University of Essex
Moshe Sipper, Swiss Federal Institute of Technology
William M. Spears, University of Wyoming
Wallace K.S. Tang, Swiss Federal Institute of Technology
Adrian Thompson, University of Sussex
Michael D. Vose, University of Tennessee
Man Leung Wong, Lingnan University

Council of Editors

Erick Cantú-Paz (chair), Lawrence Livermore National Laboratory
Karthik Balakrishnan, Fireman's Fund Insurance Company
Wolfgang Banzhaf, University of Dortmund
Peter Bentley, University College, London
Lance D. Chambers, Western Australian Department of Transport
Dipankar Dasgupta, University of Memphis
Kenneth De Jong, George Mason University
Francisco Herrera, University of Granada
William B. Langdon, University College, London
Pinaki Mazumder, University of Michigan
Eric Michielssen, University of Illinois at Urbana-Champaign
Witold Pedrycz, University of Alberta
Rajkumar Roy, Cranfield University
Elizabeth M. Rudnick, University of Illinois at Urbana-Champaign
Marc Schoenauer, INRIA Rocquencourt
Lee Spector, Hampshire College
Jose L. Verdegay, University of Granada, Spain

Council of Conferences, Riccardo Poli (Chair)

The purpose of the Council of Conferences is to provide information about the numerous conferences that are available to researchers in the field of Genetic and Evolutionary Computation, and to encourage them to coordinate their meetings to maximize our collective impact on science.

ACDM, Adaptive Computing in Design and Manufacture, 2004, Ian Parmee (Ian.Parmee@uwe.ac.uk)

EuroGP, European Conference on Genetic Programming, Portugal, April 2004, Ernesto Costa (ernesto@dei.uc.pt)

EvoWorkshops, European Evolutionary Computing Workshops, Portugal, April 2004, Stefano Cagnoni (cagnoni@ce.unipr.it)

FOGA, Foundations of Genetic Algorithms Workshop, 2004

GECCO 2004, Genetic and Evolutionary Computation Conference, Seattle, June 2004, Riccardo Poli (rpoli@essex.ac.uk)

INTROS, INtroductory TutoRials in Optimization, Search and Decision Support Methodologies, August 12, 2003, Nottingham, UK, Edmund Burke (ekb@cs.nott.ac.uk)

MISTA, 1st Multidisciplinary International Conference on Scheduling: Theory and Applications August 8-12, 2003, Nottingham, UK, Graham Kendall (gxk@cs.nott.ac.uk)

PATAT 2004, 5th International Conference on the Practice and Theory of Automated Timetabling, Pittsburgh, USA, August 18-20, 2004, Edmund Burke (ekb@cs.nott.ac.uk)

WSC8, 8th Online World Conference on Soft Computing in Industrial Applications, September 29th - October 10th, 2003, Internet (hosted by University of Dortmund), Frank Hoffmann (hoffmann@esr.e-technik.uni-dortmund.de)

An up-to-date roster of the Council of Conferences is available online at <http://www.isgec.org/conferences.html>.

Please contact the COC chair Riccardo Poli (rpoli@essex.ac.uk) for additions to this list.

Papers Nominated for Best Paper Awards

In 2002, ISGEC created a best paper award for GECCO. As part of the double blind peer review, the reviewers were asked to nominate papers for best paper awards. The chairs of core and special program committees selected the papers that received the most nominations for consideration by the conference. One winner for each program track was chosen by secret ballot of the GECCO attendees after the papers were presented in Chicago. The titles and authors of the winning papers are available at the GECCO 2003 website (www.isgec.org/GECCO-2003).

Finite Population Models of Co-evolution and Their Application to Haploidy versus Diploidy, Anthony M.L. Liekens, Huub M.M. ten Eikelder, and Peter A.J. Hilbers

A Game-Theoretic Memory Mechanism for Coevolution, Sevan G. Ficici and Jordan B. Pollack

A Non-dominated Sorting Particle Swarm Optimizer for Multiobjective Optimization, Xiaodong Li

Emergence of Collective Behavior in Evolving Populations of Flying Agents, Lee Spector, Jon Klein, Chris Perry, and Mark Feinstein

- Immune Inspired Somatic Contiguous Hypermutation for Function Optimisation, Johnny Kelsey and Jon Timmis
- Efficiency and Reliability of DNA-Based Memories, Max H. Garzon, Andrew Neel, and Hui Chen
- Hardware Evolution of Analog Speed Controllers for a DC Motor, D.A. Gwaltney and M.I. Ferguson
- Integration of Genetic Programming and Reinforcement Learning for Real Robots, Shotaro Kamio, Hideyuki Mitsuhashi, and Hitoshi Iba
- Co-evolving Task-Dependent Visual Morphologies in Predator-Prey Experiments, Gunnar Buason and Tom Ziemke
- The Steady State Behavior of $(\mu/\mu_I, \lambda)$ -ES on Ellipsoidal Fitness Models Disturbed by Noise, Hans-Georg Beyer and Dirk V. Arnold
- On the Optimization of Monotone Polynomials by the (1+1) EA and Randomized Local Search, Ingo Wegener and Carsten Witt
- Ruin and Recreate Principle Based Approach for the Quadratic Assignment Problem, Alfonsas Misevicius
- Evolutionary Computing as a tool for Grammar Development, Guy De Pauw
- Adaptive Elitist-Population Based Genetic Algorithm for Multimodal Function Optimization, Kwong-Sak Leung and Yong Liang
- Scalability of Selectorecombinative Genetic Algorithms for Problems with Tight Linkage, Kumara Sastry and David E. Goldberg
- Effective Use of Directional Information in Multi-objective Evolutionary Computation, Martin Brown and R.E. Smith
- Are Multiple Runs of Genetic Algorithms Better Than One? Erick Cantú-Paz and David E. Goldberg
- Selection in the Presence of Noise, Jürgen Branke and Christian Schmidt
- Difficulty of Unimodal and Multimodal Landscapes in Genetic Programming, Leonardo Vanneschi, Marco Tomassini, Manuel Clergue, and Philippe Collard
- Dynamic Maximum Tree Depth: a Simple Technique for Avoiding Bloat in Tree-Based GP, Sara Silva and Jonas Almeida
- Generative Representations for Evolving Families of Designs, Gregory S. Hornby
- Identifying Structural Mechanisms in Standard Genetic Programming, Jason M. Daida and Adam M. Hilss
- Visualizing Tree Structures in Genetic Programming, Jason M. Daida, Adam M. Hilss, David J. Ward, and Stephen L. Long
- Methods for Evolving Robust Programs, Liviu Panait and Sean Luke
- Population Implosion in Genetic Programming, Sean Luke, Gabriel Catalin Balan, and Liviu Panait
- Designing Efficient Exploration with MACS: Modules and Function Approximation, Pierre Gérard and Olivier Sigaud
- Tournament Selection: Stable Fitness Pressure in XCS, Martin V. Butz, Kumara Sastry, and David E. Goldberg
- Towards Building Block Propagation in XCS: a Negative Result and Its Implications, Kurian K. Tharakunnel, Martin V. Butz, and David E. Goldberg

- Quantum-Inspired Evolutionary Algorithm-Based Face Verification, Jun-Su Jang, Kuk-Hyun Han, and Jong-Hwan Kim
- Mining Comprehensive Clustering Rules with an Evolutionary Algorithm, Ioannis Sarafis, Phil Trinder and Ali Zalzal
- System-Level Synthesis of MEMS via Genetic Programming and Bond Graphs, Zhun Fan, Kisung Seo, Jianjun Hu, Ronald C. Rosenberg, and Erik D. Goodman
- Active Guidance for a Finless Rocket Using Neuroevolution, Faustino J. Gomez and Risto Miikkulainen
- Extracting Test Sequences from a Markov Software Usage Model by ACO, Karl Doerner and Walter J. Gutjahr
- Modeling the Search Landscape of Metaheuristic Software Clustering Algorithms, Brian S. Mitchell and Spiros Mancoridis

Table of Contents

Volume I

A-Life, Adaptive Behavior, Agents, and Ant Colony Optimization

Swarms in Dynamic Environments	1
<i>T.M. Blackwell</i>	
The Effect of Natural Selection on Phylogeny Reconstruction Algorithms	13
<i>Dehua Hang, Charles Ofria, Thomas M. Schmidt, Eric Torng</i>	
AntClust: Ant Clustering and Web Usage Mining	25
<i>Nicolas Labroche, Nicolas Monmarché, Gilles Venturini</i>	
A Non-dominated Sorting Particle Swarm Optimizer for Multiobjective Optimization	37
<i>Xiaodong Li</i>	
The Influence of Run-Time Limits on Choosing Ant System Parameters	49
<i>Krzysztof Socha</i>	
Emergence of Collective Behavior in Evolving Populations of Flying Agents	61
<i>Lee Spector, Jon Klein, Chris Perry, Mark Feinstein</i>	
On Role of Implicit Interaction and Explicit Communications in Emergence of Social Behavior in Continuous Predators-Prey Pursuit Problem	74
<i>Ivan Tanev, Katsunori Shimohara</i>	
Demonstrating the Evolution of Complex Genetic Representations: An Evolution of Artificial Plants	86
<i>Marc Toussaint</i>	
Sexual Selection of Co-operation	98
<i>M. Afzal Upal</i>	
Optimization Using Particle Swarms with Near Neighbor Interactions ...	110
<i>Kalyan Veeramachaneni, Thanmaya Peram, Chilukuri Mohan, Lisa Ann Osadciw</i>	

Revisiting Elitism in Ant Colony Optimization	122
<i>Tony White, Simon Kaegi, Terri Oda</i>	

A New Approach to Improve Particle Swarm Optimization	134
<i>Liping Zhang, Huanjun Yu, Shangxu Hu</i>	

A-Life, Adaptive Behavior, Agents, and Ant Colony Optimization – Posters

Clustering and Dynamic Data Visualization with Artificial Flying Insect	140
<i>S. Aupetit, N. Monmarché, M. Slimane, C. Guinot, G. Venturini</i>	

Ant Colony Programming for Approximation Problems	142
<i>Mariusz Boryczka, Zbigniew J. Czech, Wojciech Wiecezorek</i>	

Long-Term Competition for Light in Plant Simulation	144
<i>Claude Lattaud</i>	

Using Ants to Attack a Classical Cipher	146
<i>Matthew Russell, John A. Clark, Susan Stepney</i>	

Comparison of Genetic Algorithm and Particle Swarm Optimizer When Evolving a Recurrent Neural Network	148
<i>Matthew Settles, Brandon Rodebaugh, Terence Soule</i>	

Adaptation and Ruggedness in an Evolvability Landscape	150
<i>Terry Van Belle, David H. Ackley</i>	

Study Diploid System by a Hamiltonian Cycle Problem Algorithm	152
<i>Dong Xianghui, Dai Ruwei</i>	

A Possible Mechanism of Repressing Cheating Mutants in Myxobacteria	154
<i>Ying Xiao, Winfried Just</i>	

Tour Jeté, Pirouette: Dance Choreographing by Computers	156
<i>Tina Yu, Paul Johnson</i>	

Multiobjective Optimization Using Ideas from the Clonal Selection Principle	158
<i>Nareli Cruz Cortés, Carlos A. Coello Coello</i>	

Artificial Immune Systems

A Hybrid Immune Algorithm with Information Gain for the Graph Coloring Problem	171
<i>Vincenzo Cutello, Giuseppe Nicosia, Mario Pavone</i>	

MILA – Multilevel Immune Learning Algorithm	183
<i>Dipankar Dasgupta, Senhua Yu, Nivedita Sumi Majumdar</i>	
The Effect of Binary Matching Rules in Negative Selection	195
<i>Fabio González, Dipankar Dasgupta, Jonatan Gómez</i>	
Immune Inspired Somatic Contiguous Hypermutation for Function Optimisation	207
<i>Johnny Kelsey, Jon Timmis</i>	
A Scalable Artificial Immune System Model for Dynamic Unsupervised Learning	219
<i>Olfa Nasraoui, Fabio Gonzalez, Cesar Cardona, Carlos Rojas, Dipankar Dasgupta</i>	
Developing an Immunity to Spam	231
<i>Terri Oda, Tony White</i>	

Artificial Immune Systems – Posters

A Novel Immune Anomaly Detection Technique Based on Negative Selection	243
<i>F. Niño, D. Gómez, R. Vejar</i>	
Visualization of Topic Distribution Based on Immune Network Model ...	246
<i>Yasufumi Takama</i>	
Spatial Formal Immune Network	248
<i>Alexander O. Tarakanov</i>	

Coevolution

Focusing versus Intransitivity (Geometrical Aspects of Co-evolution)	250
<i>Anthony Bucci, Jordan B. Pollack</i>	
Representation Development from Pareto-Coevolution	262
<i>Edwin D. de Jong</i>	
Learning the Ideal Evaluation Function	274
<i>Edwin D. de Jong, Jordan B. Pollack</i>	
A Game-Theoretic Memory Mechanism for Coevolution	286
<i>Sevan G. Ficici, Jordan B. Pollack</i>	
The Paradox of the Plankton: Oscillations and Chaos in Multispecies Evolution	298
<i>Jeffrey Horn, James Cattron</i>	

Exploring the Explorative Advantage of the Cooperative Coevolutionary (1+1) EA	310
<i>Thomas Jansen, R. Paul Wiegand</i>	
PalmPrints: A Novel Co-evolutionary Algorithm for Clustering Finger Images	322
<i>Nawwaf Kharma, Ching Y. Suen, Pei F. Guo</i>	
Coevolution and Linear Genetic Programming for Visual Learning	332
<i>Krzysztof Krawiec and Bir Bhanu</i>	
Finite Population Models of Co-evolution and Their Application to Haploidy versus Diploidy	344
<i>Anthony M.L. Liekens, Huub M.M. ten Eikelder, Peter A.J. Hilbers</i>	
Evolving Keepaway Soccer Players through Task Decomposition	356
<i>Shimon Whiteson, Nate Kohl, Risto Mäkkiläinen, Peter Stone</i>	

Coevolution – Posters

A New Method of Multilayer Perceptron Encoding	369
<i>Emmanuel Blindauer, Jerzy Korczak</i>	
An Incremental and Non-generational Coevolutionary Algorithm	371
<i>Ramón Alfonso Palacios-Durazo, Manuel Valenzuela-Rendón</i>	
Coevolutionary Convergence to Global Optima	373
<i>Lothar M. Schmitt</i>	
Generalized Extremal Optimization for Solving Complex Optimal Design Problems	375
<i>Fabiano Luis de Sousa, Valeri Vlassov, Fernando Manuel Ramos</i>	
Coevolving Communication and Cooperation for Lattice Formation Tasks	377
<i>Jekanthan Thangavelautham, Timothy D. Barfoot, Gabriele M.T. D’Eleuterio</i>	

DNA, Molecular, and Quantum Computing

Efficiency and Reliability of DNA-Based Memories	379
<i>Max H. Garzon, Andrew Neel, Hui Chen</i>	
Evolving Hogg’s Quantum Algorithm Using Linear-Tree GP	390
<i>André Leier, Wolfgang Banzhaf</i>	
Hybrid Networks of Evolutionary Processors	401
<i>Carlos Martín-Vide, Victor Mitrana, Mario J. Pérez-Jiménez, Fernando Sancho-Caparrini</i>	

DNA-Like Genomes for Evolution <i>in silico</i>	413
<i>Michael West, Max H. Garzon, Derrel Blain</i>	

DNA, Molecular, and Quantum Computing – Posters

String Binding-Blocking Automata	425
<i>M. Sakthi Balan</i>	
On Setting the Parameters of QEA for Practical Applications: Some Guidelines Based on Empirical Evidence	427
<i>Kuk-Hyun Han, Jong-Hwan Kim</i>	
Evolutionary Two-Dimensional DNA Sequence Alignment	429
<i>Edgar E. Vallejo, Fernando Ramos</i>	

Evolvable Hardware

Active Control of Thermoacoustic Instability in a Model Combustor with Neuromorphic Evolvable Hardware	431
<i>John C. Gallagher, Saranyan Vignraham</i>	
Hardware Evolution of Analog Speed Controllers for a DC Motor	442
<i>David A. Gwaltney, Michael I. Ferguson</i>	

Evolvable Hardware – Posters

An Examination of Hypermutation and Random Immigrant Variants of mrCGA for Dynamic Environments	454
<i>Gregory R. Kramer, John C. Gallagher</i>	
Inherent Fault Tolerance in Evolved Sorting Networks	456
<i>Rob Shepherd and James Foster</i>	

Evolutionary Robotics

Co-evolving Task-Dependent Visual Morphologies in Predator-Prey Experiments	458
<i>Gunnar Buason, Tom Ziemke</i>	
Integration of Genetic Programming and Reinforcement Learning for Real Robots	470
<i>Shotaro Kamio, Hideyuki Mitsuhashi, Hitoshi Iba</i>	
Multi-objectivity as a Tool for Constructing Hierarchical Complexity	483
<i>Jason Teo, Minh Ha Nguyen, Hussein A. Abbass</i>	
Learning Biped Locomotion from First Principles on a Simulated Humanoid Robot Using Linear Genetic Programming	495
<i>Krister Wolff, Peter Nordin</i>	

Evolutionary Robotics – Posters

An Evolutionary Approach to Automatic Construction of the Structure in Hierarchical Reinforcement Learning	507
<i>Stefan Elfving, Eiji Uchibe, Kenji Doya</i>	

Fractional Order Dynamical Phenomena in a GA	510
<i>E.J. Solteiro Pires, J.A. Tenreiro Machado, P.B. de Moura Oliveira</i>	

Evolution Strategies/Evolutionary Programming

Dimension-Independent Convergence Rate for Non-isotropic (1, λ) – ES	512
<i>Anne Auger, Claude Le Bris, Marc Schoenauer</i>	

The Steady State Behavior of $(\mu/\mu_I, \lambda)$ -ES on Ellipsoidal Fitness Models Disturbed by Noise	525
<i>Hans-Georg Beyer, Dirk V. Arnold</i>	

Theoretical Analysis of Simple Evolution Strategies in Quickly Changing Environments	537
<i>Jürgen Branke, Wei Wang</i>	

Evolutionary Computing as a Tool for Grammar Development	549
<i>Guy De Pauw</i>	

Solving Distributed Asymmetric Constraint Satisfaction Problems Using an Evolutionary Society of Hill-Climbers	561
<i>Gerry Dozier</i>	

Use of Multiobjective Optimization Concepts to Handle Constraints in Single-Objective Optimization	573
<i>Arturo Hernández Aguirre, Salvador Botello Rionda, Carlos A. Coello Coello, Giovanni Lizárraga Lizárraga</i>	

Evolution Strategies with Exclusion-Based Selection Operators and a Fourier Series Auxiliary Function	585
<i>Kwong-Sak Leung, Yong Liang</i>	

Ruin and Recreate Principle Based Approach for the Quadratic Assignment Problem	598
<i>Alfonas Misevicius</i>	

Model-Assisted Steady-State Evolution Strategies	610
<i>Holger Ulmer, Felix Streichert, Andreas Zell</i>	

On the Optimization of Monotone Polynomials by the (1+1) EA and Randomized Local Search	622
<i>Ingo Wegener, Carsten Witt</i>	

Evolution Strategies/Evolutionary Programming – Posters

A Forest Representation for Evolutionary Algorithms Applied to Network Design	634
<i>A.C.B. Delbem, Andre de Carvalho</i>	
Solving Three-Objective Optimization Problems Using Evolutionary Dynamic Weighted Aggregation: Results and Analysis	636
<i>Yaochu Jin, Tatsuya Okabe, Bernhard Sendhoff</i>	
The Principle of Maximum Entropy-Based Two-Phase Optimization of Fuzzy Controller by Evolutionary Programming	638
<i>Chi-Ho Lee, Ming Yuchi, Hyun Myung, Jong-Hwan Kim</i>	
A Simple Evolution Strategy to Solve Constrained Optimization Problems	640
<i>Efrén Mezura-Montes, Carlos A. Coello Coello</i>	
Effective Search of the Energy Landscape for Protein Folding	642
<i>Eugene Santos Jr., Keum Joo Kim, Eunice E. Santos</i>	
A Clustering Based Niching Method for Evolutionary Algorithms	644
<i>Felix Streichert, Gunnar Stein, Holger Ulmer, Andreas Zell</i>	

Evolutionary Scheduling Routing

A Hybrid Genetic Algorithm for the Capacitated Vehicle Routing Problem	646
<i>Jean Berger, Mohamed Barkaoui</i>	
An Evolutionary Approach to Capacitated Resource Distribution by a Multiple-agent Team	657
<i>Mudassar Hussain, Bahram Kimiaghali, Abdollah Homaiyar, Albert Esterline, Bijan Sayyaroosari</i>	
A Hybrid Genetic Algorithm Based on Complete Graph Representation for the Sequential Ordering Problem	669
<i>Dong-Il Seo, Byung-Ro Moon</i>	
An Optimization Solution for Packet Scheduling: A Pipeline-Based Genetic Algorithm Accelerator	681
<i>Shiann-Tsong Sheu, Yue-Ru Chuang, Yu-Hung Chen, Eugene Lai</i>	

Evolutionary Scheduling Routing – Posters

Generation and Optimization of Train Timetables Using Coevolution	693
<i>Paavan Mistry, Raymond S.K. Kwan</i>	

Genetic Algorithms

Chromosome Reuse in Genetic Algorithms	695
<i>Adnan Acan, Yüce Tekol</i>	
Real-Parameter Genetic Algorithms for Finding Multiple Optimal Solutions in Multi-modal Optimization	706
<i>Pedro J. Ballester, Jonathan N. Carter</i>	
An Adaptive Penalty Scheme for Steady-State Genetic Algorithms	718
<i>Helio J.C. Barbosa, Afonso C.C. Lemonge</i>	
Asynchronous Genetic Algorithms for Heterogeneous Networks Using Coarse-Grained Dataflow	730
<i>John W. Baugh Jr., Sujay V. Kumar</i>	
A Generalized Feedforward Neural Network Architecture and Its Training Using Two Stochastic Search Methods	742
<i>Abdesselam Bouzerdoum, Rainer Mueller</i>	
Ant-Based Crossover for Permutation Problems	754
<i>Jürgen Branke, Christiane Barz, Ivesa Behrens</i>	
Selection in the Presence of Noise	766
<i>Jürgen Branke, Christian Schmidt</i>	
Effective Use of Directional Information in Multi-objective Evolutionary Computation	778
<i>Martin Brown, R.E. Smith</i>	
Pruning Neural Networks with Distribution Estimation Algorithms	790
<i>Erick Cantú-Paz</i>	
Are Multiple Runs of Genetic Algorithms Better than One?	801
<i>Erick Cantú-Paz, David E. Goldberg</i>	
Constrained Multi-objective Optimization Using Steady State Genetic Algorithms	813
<i>Deepti Chafekar, Jiang Xuan, Khaled Rasheed</i>	
An Analysis of a Reordering Operator with Tournament Selection on a GA-Hard Problem	825
<i>Ying-Ping Chen, David E. Goldberg</i>	
Tightness Time for the Linkage Learning Genetic Algorithm	837
<i>Ying-Ping Chen, David E. Goldberg</i>	
A Hybrid Genetic Algorithm for the Hexagonal Tortoise Problem	850
<i>Heemahn Choe, Sung-Soon Choi, Byung-Ro Moon</i>	

Normalization in Genetic Algorithms	862
<i>Sung-Soon Choi and Byung-Ro Moon</i>	
Coarse-Graining in Genetic Algorithms: Some Issues and Examples	874
<i>Andrés Aguilar Contreras, Jonathan E. Rowe, Christopher R. Stephens</i>	
Building a GA from Design Principles for Learning Bayesian Networks . . .	886
<i>Steven van Dijk, Dirk Thierens, Linda C. van der Gaag</i>	
A Method for Handling Numerical Attributes in GA-Based Inductive Concept Learners	898
<i>Federico Divina, Maarten Keijzer, Elena Marchiori</i>	
Analysis of the (1+1) EA for a Dynamically Bitwise Changing ONEMAX	909
<i>Stefan Droste</i>	
Performance Evaluation and Population Reduction for a Self Adaptive Hybrid Genetic Algorithm (SAHGA)	922
<i>Felipe P. Espinoza, Barbara S. Minsker, David E. Goldberg</i>	
Schema Analysis of Average Fitness in Multiplicative Landscape	934
<i>Hiroshi Furutani</i>	
On the Treewidth of NK Landscapes	948
<i>Yong Gao, Joseph Culberson</i>	
Selection Intensity in Asynchronous Cellular Evolutionary Algorithms . . .	955
<i>Mario Giacobini, Enrique Alba, Marco Tomassini</i>	
A Case for Codons in Evolutionary Algorithms	967
<i>Joshua Gilbert, Maggie Eppstein</i>	
Natural Coding: A More Efficient Representation for Evolutionary Learning	979
<i>Raúl Giráldez, Jesús S. Aguilar-Ruiz, José C. Riquelme</i>	
Hybridization of Estimation of Distribution Algorithms with a Repair Method for Solving Constraint Satisfaction Problems	991
<i>Hisashi Handa</i>	
Efficient Linkage Discovery by Limited Probing	1003
<i>Robert B. Heckendorn, Alden H. Wright</i>	
Distributed Probabilistic Model-Building Genetic Algorithm	1015
<i>Tomoyuki Hiroyasu, Mitsunori Miki, Masaki Sano, Hisashi Shimosaka, Shigeyoshi Tsutsui, Jack Dongarra</i>	

HEMO: A Sustainable Multi-objective Evolutionary Optimization Framework	1029
<i>Jianjun Hu, Kisung Seo, Zhun Fan, Ronald C. Rosenberg, Erik D. Goodman</i>	
Using an Immune System Model to Explore Mate Selection in Genetic Algorithms	1041
<i>Chien-Feng Huang</i>	
Designing A Hybrid Genetic Algorithm for the Linear Ordering Problem	1053
<i>Gaofeng Huang, Andrew Lim</i>	
A Similarity-Based Mating Scheme for Evolutionary Multiobjective Optimization	1065
<i>Hisao Ishibuchi, Youhei Shibata</i>	
Evolutionary Multiobjective Optimization for Generating an Ensemble of Fuzzy Rule-Based Classifiers	1077
<i>Hisao Ishibuchi, Takashi Yamamoto</i>	
Voronoi Diagrams Based Function Identification	1089
<i>Carlos Kavka, Marc Schoenauer</i>	
New Usage of SOM for Genetic Algorithms	1101
<i>Jung-Hwan Kim, Byung-Ro Moon</i>	
Problem-Independent Schema Synthesis for Genetic Algorithms	1112
<i>Yong-Hyuk Kim, Yung-Keun Kwon, Byung-Ro Moon</i>	
Investigation of the Fitness Landscapes and Multi-parent Crossover for Graph Bipartitioning	1123
<i>Yong-Hyuk Kim, Byung-Ro Moon</i>	
New Usage of Sammon's Mapping for Genetic Visualization	1136
<i>Yong-Hyuk Kim, Byung-Ro Moon</i>	
Exploring a Two-Population Genetic Algorithm	1148
<i>Steven Orla Kimbrough, Ming Lu, David Harlan Wood, D.J. Wu</i>	
Adaptive Elitist-Population Based Genetic Algorithm for Multimodal Function Optimization	1160
<i>Kwong-Sak Leung, Yong Liang</i>	
Wise Breeding GA via Machine Learning Techniques for Function Optimization	1172
<i>Xavier Llorà, David E. Goldberg</i>	

Facts and Fallacies in Using Genetic Algorithms for Learning Clauses in First-Order Logic	1184
<i>Flaviu Adrian Mărginean</i>	
Comparing Evolutionary Computation Techniques via Their Representation	1196
<i>Boris Mitavskiy</i>	
Dispersion-Based Population Initialization	1210
<i>Ronald W. Morrison</i>	
A Parallel Genetic Algorithm Based on Linkage Identification	1222
<i>Masaharu Munetomo, Naoya Murao, Kiyoshi Akama</i>	
Generalization of Dominance Relation-Based Replacement Rules for Memetic EMO Algorithms	1234
<i>Tadahiko Murata, Shiori Kaige, Hisao Ishibuchi</i>	

Author Index

Volume II

Genetic Algorithms (continued)

Design of Multithreaded Estimation of Distribution Algorithms	1247
<i>Jiri Ocenasek, Josef Schwarz, Martin Pelikan</i>	
Reinforcement Learning Estimation of Distribution Algorithm	1259
<i>Topon Kumar Paul, Hitoshi Iba</i>	
Hierarchical BOA Solves Ising Spin Glasses and MAXSAT	1271
<i>Martin Pelikan, David E. Goldberg</i>	
ERA: An Algorithm for Reducing the Epistasis of SAT Problems	1283
<i>Eduardo Rodriguez-Tello, Jose Torres-Jimenez</i>	
Learning a Procedure That Can Solve Hard Bin-Packing Problems: A New GA-Based Approach to Hyper-heuristics	1295
<i>Peter Ross, Javier G. Marín-Blázquez, Sonia Schulenburg, Emma Hart</i>	
Population Sizing for the Redundant Trivial Voting Mapping	1307
<i>Franz Rothlauf</i>	
Non-stationary Function Optimization Using Polygenic Inheritance	1320
<i>Conor Ryan, J.J. Collins, David Wallin</i>	

Scalability of Selectorecombinative Genetic Algorithms for Problems with Tight Linkage	1332
<i>Kumara Sastry, David E. Goldberg</i>	
New Entropy-Based Measures of Gene Significance and Epistasis	1345
<i>Dong-Il Seo, Yong-Hyuk Kim, Byung-Ro Moon</i>	
A Survey on Chromosomal Structures and Operators for Exploiting Topological Linkages of Genes	1357
<i>Dong-Il Seo, Byung-Ro Moon</i>	
Cellular Programming and Symmetric Key Cryptography Systems	1369
<i>Franciszek Seredyński, Pascal Bouvry, Albert Y. Zomaya</i>	
Mating Restriction and Niching Pressure: Results from Agents and Implications for General EC	1382
<i>R.E. Smith, Claudio Bonacina</i>	
EC Theory: A Unified Viewpoint	1394
<i>Christopher R. Stephens, Adolfo Zamora</i>	
Real Royal Road Functions for Constant Population Size	1406
<i>Tobias Storch, Ingo Wegener</i>	
Two Broad Classes of Functions for Which a No Free Lunch Result Does Not Hold	1418
<i>Matthew J. Streeter</i>	
Dimensionality Reduction via Genetic Value Clustering	1431
<i>Alexander Topchy, William Punch</i>	
The Structure of Evolutionary Exploration: On Crossover, Buildings Blocks, and Estimation-of-Distribution Algorithms	1444
<i>Marc Toussaint</i>	
The Virtual Gene Genetic Algorithm	1457
<i>Manuel Valenzuela-Rendón</i>	
Quad Search and Hybrid Genetic Algorithms	1469
<i>Darrell Whitley, Deon Garrett, Jean-Paul Watson</i>	
Distance between Populations	1481
<i>Mark Wineberg, Franz Oppacher</i>	
The Underlying Similarity of Diversity Measures Used in Evolutionary Computation	1493
<i>Mark Wineberg, Franz Oppacher</i>	
Implicit Parallelism	1505
<i>Alden H. Wright, Michael D. Vose, Jonathan E. Rowe</i>	

Finding Building Blocks through Eigenstructure Adaptation	1518
<i>Danica Wyatt, Hod Lipson</i>	
A Specialized Island Model and Its Application in Multiobjective Optimization	1530
<i>Ningchuan Xiao, Marc P. Armstrong</i>	
Adaptation of Length in a Nonstationary Environment	1541
<i>Han Yu, Annie S. Wu, Kuo-Chi Lin, Guy Schiavone</i>	
Optimal Sampling and Speed-Up for Genetic Algorithms on the Sampled OneMax Problem	1554
<i>Tian-Li Yu, David E. Goldberg, Kumara Sastry</i>	
Building-Block Identification by Simultaneity Matrix	1566
<i>Chatchawit Apornthewan, Prabhas Chongstitvatana</i>	
A Unified Framework for Metaheuristics	1568
<i>Jürgen Branke, Michael Stein, Hartmut Schmeck</i>	
The Hitting Set Problem and Evolutionary Algorithmic Techniques with ad-hoc Viruses (HEAT-V)	1570
<i>Vincenzo Cutello, Francesco Pappalardo</i>	
The Spatially-Dispersed Genetic Algorithm	1572
<i>Grant Dick</i>	
Non-universal Suffrage Selection Operators Favor Population Diversity in Genetic Algorithms	1574
<i>Federico Divina, Maarten Keijzer, Elena Marchiori</i>	
Uniform Crossover Revisited: Maximum Disruption in Real-Coded GAs	1576
<i>Stephen Drake</i>	
The Master-Slave Architecture for Evolutionary Computations Revisited	1578
<i>Christian Gagné, Marc Parizeau, Marc Dubreuil</i>	
Genetic Algorithms – Posters	
Using Adaptive Operators in Genetic Search	1580
<i>Jonatan Gómez, Dipankar Dasgupta, Fabio González</i>	
A Kernighan-Lin Local Improvement Heuristic That Solves Some Hard Problems in Genetic Algorithms	1582
<i>William A. Greene</i>	
GA-Hardness Revisited	1584
<i>Haipeng Guo, William H. Hsu</i>	

Barrier Trees For Search Analysis	1586
<i>Jonathan Hallam, Adam Prügel-Bennett</i>	
A Genetic Algorithm as a Learning Method Based on Geometric Representations	1588
<i>Gregory A. Holifield, Annie S. Wu</i>	
Solving Mastermind Using Genetic Algorithms	1590
<i>Tom Kalisker, Doug Camens</i>	
Evolutionary Multimodal Optimization Revisited	1592
<i>Rajeev Kumar, Peter Rockett</i>	
Integrated Genetic Algorithm with Hill Climbing for Bandwidth Minimization Problem	1594
<i>Andrew Lim, Brian Rodrigues, Fei Xiao</i>	
A Fixed-Length Subset Genetic Algorithm for the p-Median Problem	1596
<i>Andrew Lim, Zhou Xu</i>	
Performance Evaluation of a Parameter-Free Genetic Algorithm for Job-Shop Scheduling Problems	1598
<i>Shouichi Matsui, Isamu Watanabe, Ken-ichi Tokoro</i>	
SEPA: Structure Evolution and Parameter Adaptation in Feed-Forward Neural Networks	1600
<i>Paulito P. Palmes, Taichi Hayasaka, Shiro Usui</i>	
Real-Coded Genetic Algorithm to Reveal Biological Significant Sites of Remotely Homologous Proteins	1602
<i>Sung-Joon Park, Masayuki Yamamura</i>	
Understanding EA Dynamics via Population Fitness Distributions	1604
<i>Elena Popovici, Kenneth De Jong</i>	
Evolutionary Feature Space Transformation Using Type-Restricted Generators	1606
<i>Oliver Ritthoff, Ralf Klinkenberg</i>	
On the Locality of Representations	1608
<i>Franz Rothlauf</i>	
New Subtour-Based Crossover Operator for the TSP	1610
<i>Sang-Moon Soak, Byung-Ha Ahn</i>	
Is a Self-Adaptive Pareto Approach Beneficial for Controlling Embodied Virtual Robots?	1612
<i>Jason Teo, Hussein A. Abbass</i>	

A Genetic Algorithm for Energy Efficient Device Scheduling in Real-Time Systems	1614
<i>Lirong Tian, Tughrul Arslan</i>	
Metropolitan Area Network Design Using GA Based on Hierarchical Linkage Identification.....	1616
<i>Miwako Tsuji, Masaharu Munetomo, Kiyoshi Akama</i>	
Statistics-Based Adaptive Non-uniform Mutation for Genetic Algorithms	1618
<i>Shengxiang Yang</i>	
Genetic Algorithm Design Inspired by Organizational Theory: Pilot Study of a Dependency Structure Matrix Driven Genetic Algorithm	1620
<i>Tian-Li Yu, David E. Goldberg, Ali Yassine, Ying-Ping Chen</i>	
Are the “Best” Solutions to a Real Optimization Problem Always Found in the Noninferior Set? Evolutionary Algorithm for Generating Alternatives (EAGA)	1622
<i>Emily M. Zechman, S. Ranji Ranjithan</i>	
Population Sizing Based on Landscape Feature	1624
<i>Jian Zhang, Xiaohui Yuan, Bill P. Buckles</i>	
Genetic Programming	
Structural Emergence with Order Independent Representations	1626
<i>R. Muhammad Atif Azad, Conor Ryan</i>	
Identifying Structural Mechanisms in Standard Genetic Programming ...	1639
<i>Jason M. Daida, Adam M. Hilss</i>	
Visualizing Tree Structures in Genetic Programming	1652
<i>Jason M. Daida, Adam M. Hilss, David J. Ward, Stephen L. Long</i>	
What Makes a Problem GP-Hard? Validating a Hypothesis of Structural Causes	1665
<i>Jason M. Daida, Hsiao-Wei Li, Ricky Tang, Adam M. Hilss</i>	
Generative Representations for Evolving Families of Designs	1678
<i>Gregory S. Hornby</i>	
Evolutionary Computation Method for Promoter Site Prediction in DNA	1690
<i>Daniel Howard, Karl Benson</i>	
Convergence of Program Fitness Landscapes	1702
<i>W.B. Langdon</i>	

Multi-agent Learning of Heterogeneous Robots by Evolutionary Subsumption	1715
<i>Hongwei Liu, Hitoshi Iba</i>	
Population Implosion in Genetic Programming.....	1729
<i>Sean Luke, Gabriel Catalin Balan, Liviu Panait</i>	
Methods for Evolving Robust Programs.....	1740
<i>Liviu Panait, Sean Luke</i>	
On the Avoidance of Fruitless Wraps in Grammatical Evolution	1752
<i>Conor Ryan, Maarten Keijzer, Miguel Nicolau</i>	
Dense and Switched Modular Primitives for Bond Graph Model Design ..	1764
<i>Kisung Seo, Zhun Fan, Jianjun Hu, Erik D. Goodman, Ronald C. Rosenberg</i>	
Dynamic Maximum Tree Depth.....	1776
<i>Sara Silva, Jonas Almeida</i>	
Difficulty of Unimodal and Multimodal Landscapes in Genetic Programming.....	1788
<i>Leonardo Vanneschi, Marco Tomassini, Manuel Clergue, Philippe Collard</i>	
Genetic Programming – Posters	
Ramped Half-n-Half Initialisation Bias in GP.....	1800
<i>Edmund Burke, Steven Gustafson, Graham Kendall</i>	
Improving Evolvability of Genetic Parallel Programming Using Dynamic Sample Weighting	1802
<i>Sin Man Cheang, Kin Hong Lee, Kwong Sak Leung</i>	
Enhancing the Performance of GP Using an Ancestry-Based Mate Selection Scheme.....	1804
<i>Rodney Fry, Andy Tyrrell</i>	
A General Approach to Automatic Programming Using Occam’s Razor, Compression, and Self-Inspection	1806
<i>Peter Galos, Peter Nordin, Joel Olsén, Kristofer Sundén Ringnér</i>	
Building Decision Tree Software Quality Classification Models Using Genetic Programming.....	1808
<i>Yi Liu, Taghi M. Khoshgoftaar</i>	
Evolving Petri Nets with a Genetic Algorithm	1810
<i>Holger Mauch</i>	

Diversity in Multipopulation Genetic Programming	1812
<i>Marco Tomassini, Leonardo Vanneschi, Francisco Fernández, Germán Galeano</i>	
An Encoding Scheme for Generating λ -Expressions in Genetic Programming	1814
<i>Kazuto Tominaga, Tomoya Suzuki, Kazuhiro Oka</i>	
AVICE: Evolving Avatar's Movement	1816
<i>Hiromi Wakaki, Hitoshi Iba</i>	

Learning Classifier Systems

Evolving Multiple Discretizations with Adaptive Intervals for a Pittsburgh Rule-Based Learning Classifier System	1818
<i>Jaume Bacardit, Josep Maria Garrell</i>	
Limits in Long Path Learning with XCS	1832
<i>Alwyn Barry</i>	
Bounding the Population Size in XCS to Ensure Reproductive Opportunities	1844
<i>Martin V. Butz, David E. Goldberg</i>	
Tournament Selection: Stable Fitness Pressure in XCS	1857
<i>Martin V. Butz, Kumara Sastry, David E. Goldberg</i>	
Improving Performance in Size-Constrained Extended Classifier Systems	1870
<i>Devon Dawson</i>	
Designing Efficient Exploration with MACS: Modules and Function Approximation	1882
<i>Pierre Gérard, Olivier Sigaud</i>	
Estimating Classifier Generalization and Action's Effect: A Minimalist Approach	1894
<i>Pier Luca Lanzi</i>	
Towards Building Block Propagation in XCS: A Negative Result and Its Implications	1906
<i>Kurian K. Tharakunnel, Martin V. Butz, David E. Goldberg</i>	

Learning Classifier Systems – Posters

Data Classification Using Genetic Parallel Programming	1918
<i>Sin Man Cheang, Kin Hong Lee, Kwong Sak Leung</i>	
Dynamic Strategies in a Real-Time Strategy Game	1920
<i>William Joseph Falke II, Peter Ross</i>	

Using Raw Accuracy to Estimate Classifier Fitness in XCS	1922
<i>Pier Luca Lanzi</i>	

Towards Learning Classifier Systems for Continuous-Valued Online Environments	1924
<i>Christopher Stone, Larry Bull</i>	

Real World Applications

Artificial Immune System for Classification of Gene Expression Data	1926
<i>Shin Ando, Hitoshi Iba</i>	

Automatic Design Synthesis and Optimization of Component-Based Systems by Evolutionary Algorithms	1938
<i>P.P. Angelov, Y. Zhang, J.A. Wright, V.I. Hanby, R.A. Buswell</i>	

Studying the Advantages of a Messy Evolutionary Algorithm for Natural Language Tagging	1951
<i>Lourdes Araujo</i>	

Optimal Elevator Group Control by Evolution Strategies.....	1963
<i>Thomas Beielstein, Claus-Peter Ewald, Sandor Markon</i>	

A Methodology for Combining Symbolic Regression and Design of Experiments to Improve Empirical Model Building	1975
<i>Flor Castillo, Kenric Marshall, James Green, Arthur Kordon</i>	

The General Yard Allocation Problem	1986
<i>Ping Chen, Zhaohui Fu, Andrew Lim, Brian Rodrigues</i>	

Connection Network and Optimization of Interest Metric for One-to-One Marketing.....	1998
<i>Sung-Soon Choi, Byung-Ro Moon</i>	

Parameter Optimization by a Genetic Algorithm for a Pitch Tracking System	2010
<i>Yoon-Seok Choi, Byung-Ro Moon</i>	

Secret Agents Leave Big Footprints: How to Plant a Cryptographic Trapdoor, and Why You Might Not Get Away with It	2022
<i>John A. Clark, Jeremy L. Jacob, Susan Stepney</i>	

GenTree: An Interactive Genetic Algorithms System for Designing 3D Polygonal Tree Models	2034
<i>Clare Bates Congdon, Raymond H. Mazza</i>	

Optimisation of Reaction Mechanisms for Aviation Fuels Using a Multi-objective Genetic Algorithm	2046
<i>Lionel Elliott, Derek B. Ingham, Adrian G. Kyne, Nicolae S. Mera, Mohamed Pourkashanian, Chritopher W. Wilson</i>	

System-Level Synthesis of MEMS via Genetic Programming and Bond Graphs	2058
<i>Zhun Fan, Kisung Seo, Jianjun Hu, Ronald C. Rosenberg, Erik D. Goodman</i>	
Congressional Districting Using a TSP-Based Genetic Algorithm	2072
<i>Sean L. Forman, Yading Yue</i>	
Active Guidance for a Finless Rocket Using Neuroevolution	2084
<i>Faustino J. Gomez, Risto Mikkulainen</i>	
Simultaneous Assembly Planning and Assembly System Design Using Multi-objective Genetic Algorithms	2096
<i>Karim Hamza, Juan F. Reyes-Luna, Kazuhiro Saitou</i>	
Multi-FPGA Systems Synthesis by Means of Evolutionary Computation	2109
<i>J.I. Hidalgo, F. Fernández, J. Lanchares, J.M. Sánchez, R. Hermida, M. Tomassini, R. Baraglia, R. Perego, O. Garnica</i>	
Genetic Algorithm Optimized Feature Transformation – A Comparison with Different Classifiers	2121
<i>Zhijian Huang, Min Pei, Erik Goodman, Yong Huang, Gaoping Li</i>	
Web-Page Color Modification for Barrier-Free Color Vision with Genetic Algorithm	2134
<i>Manabu Ichikawa, Kiyoshi Tanaka, Shoji Kondo, Koji Hiroshima, Kazuo Ichikawa, Shoko Tanabe, Kiichiro Fukami</i>	
Quantum-Inspired Evolutionary Algorithm-Based Face Verification	2147
<i>Jun-Su Jang, Kuk-Hyun Han, Jong-Hwan Kim</i>	
Minimization of Sonic Boom on Supersonic Aircraft Using an Evolutionary Algorithm	2157
<i>Charles L. Karr, Rodney Bowersox, Vishnu Singh</i>	
Optimizing the Order of Taxon Addition in Phylogenetic Tree Construction Using Genetic Algorithm	2168
<i>Yong-Hyuk Kim, Seung-Kyu Lee, Byung-Ro Moon</i>	
Multicriteria Network Design Using Evolutionary Algorithm	2179
<i>Rajeev Kumar, Nilanjan Banerjee</i>	
Control of a Flexible Manipulator Using a Sliding Mode Controller with Genetic Algorithm Tuned Manipulator Dimension	2191
<i>N.M. Kwok, S. Kwong</i>	
Daily Stock Prediction Using Neuro-genetic Hybrids	2203
<i>Yung-Keun Kwon, Byung-Ro Moon</i>	

Finding the Optimal Gene Order in Displaying Microarray Data	2215
<i>Seung-Kyu Lee, Yong-Hyuk Kim, Byung-Ro Moon</i>	
Learning Features for Object Recognition	2227
<i>Yingqiang Lin, Bir Bhanu</i>	
An Efficient Hybrid Genetic Algorithm for a Fixed Channel Assignment Problem with Limited Bandwidth	2240
<i>Showichi Matsui, Isamu Watanabe, Ken-ichi Tokoro</i>	
Using Genetic Algorithms for Data Mining Optimization in an Educational Web-Based System	2252
<i>Behrouz Minaei-Bidgoli, William F. Punch</i>	
Improved Image Halftoning Technique Using GAs with Concurrent Inter-block Evaluation	2264
<i>Emi Myodo, Hernán Aguirre, Kiyoshi Tanaka</i>	
Complex Function Sets Improve Symbolic Discriminant Analysis of Microarray Data	2277
<i>David M. Reif, Bill C. White, Nancy Olsen, Thomas Aune, Jason H. Moore</i>	
GA-Based Inference of Euler Angles for Single Particle Analysis	2288
<i>Shusuke Saeki, Kiyoshi Asai, Katsutoshi Takahashi, Yutaka Ueno, Katsunori Isono, Hitoshi Iba</i>	
Mining Comprehensible Clustering Rules with an Evolutionary Algorithm	2301
<i>Ioannis Sarafis, Phil Trinder, Ali Zalzal</i>	
Evolving Consensus Sequence for Multiple Sequence Alignment with a Genetic Algorithm	2313
<i>Conrad Shyu, James A. Foster</i>	
A Linear Genetic Programming Approach to Intrusion Detection	2325
<i>Dong Song, Malcolm I. Heywood, A. Nur Zincir-Heywood</i>	
Genetic Algorithm for Supply Planning Optimization under Uncertain Demand	2337
<i>Tezuka Masaru, Hiji Masahiro</i>	
Genetic Algorithms: A Fundamental Component of an Optimization Toolkit for Improved Engineering Designs	2347
<i>Siu Tong, David J. Powell</i>	
Spatial Operators for Evolving Dynamic Bayesian Networks from Spatio-temporal Data	2360
<i>Allan Tucker, Xiaohui Liu, David Garway-Heath</i>	

An Evolutionary Approach for Molecular Docking	2372
<i>Jinn-Moon Yang</i>	
Evolving Sensor Suites for Enemy Radar Detection	2384
<i>Ayşe S. Yilmaz, Brian N. McQuay, Han Yu, Annie S. Wu, John C. Sciortino, Jr.</i>	

Real World Applications – Posters

Optimization of Spare Capacity in Survivable WDM Networks	2396
<i>H.W. Chong, Sam Kwong</i>	
Partner Selection in Virtual Enterprises by Using Ant Colony Optimization in Combination with the Analytical Hierarchy Process	2398
<i>Marco Fischer, Hendrik Jähn, Tobias Teich</i>	
Quadrilateral Mesh Smoothing Using a Steady State Genetic Algorithm	2400
<i>Mike Holder, Charles L. Karr</i>	
Evolutionary Algorithms for Two Problems from the Calculus of Variations	2402
<i>Bryant A. Julstrom</i>	
Genetic Algorithm Frequency Domain Optimization of an Anti-Resonant Electromechanical Controller	2404
<i>Charles L. Karr, Douglas A. Scott</i>	
Genetic Algorithm Optimization of a Filament Winding Process	2406
<i>Charles L. Karr, Eric Wilson, Sherri Messimer</i>	
Circuit Bipartitioning Using Genetic Algorithm	2408
<i>Jong-Pil Kim, Byung-Ro Moon</i>	
Multi-campaign Assignment Problem and Optimizing Lagrange Multipliers	2410
<i>Yong-Hyuk Kim, Byung-Ro Moon</i>	
Grammatical Evolution for the Discovery of Petri Net Models of Complex Genetic Systems	2412
<i>Jason H. Moore, Lance W. Hahn</i>	
Evaluation of Parameter Sensitivity for Portable Embedded Systems through Evolutionary Techniques	2414
<i>James Northern, Michael Shanblatt</i>	
An Evolutionary Algorithm for the Joint Replenishment of Inventory with Interdependent Ordering Costs	2416
<i>Anne Olsen</i>	

Benefits of Implicit Redundant Genetic Algorithms for Structural Damage Detection in Noisy Environments	2418
<i>Anne Raich, Tamás Liszkai</i>	
Multi-objective Traffic Signal Timing Optimization Using Non-dominated Sorting Genetic Algorithm II	2420
<i>Dazhi Sun, Rahim F. Benekohal, S. Travis Waller</i>	
Exploration of a Two Sided Rendezvous Search Problem Using Genetic Algorithms	2422
<i>T.Q.S. Truong, A. Stacey</i>	
Taming a Flood with a T-CUP – Designing Flood-Control Structures with a Genetic Algorithm	2424
<i>Jeff Wallace, Sushil J. Louis</i>	
Assignment Copy Detection Using Neuro-genetic Hybrids	2426
<i>Seung-Jin Yang, Yong-Geon Kim, Yung-Keun Kwon, Byung-Ro Moon</i>	
Search Based Software Engineering	
Structural and Functional Sequence Test of Dynamic and State-Based Software with Evolutionary Algorithms	2428
<i>André Baresel, Hartmut Pohlheim, Sadegh Sadeghipour</i>	
Evolutionary Testing of Flag Conditions	2442
<i>Andre Baresel, Harmen Sthamer</i>	
Predicate Expression Cost Functions to Guide Evolutionary Search for Test Data	2455
<i>Leonardo Bottaci</i>	
Extracting Test Sequences from a Markov Software Usage Model by ACO	2465
<i>Karl Doerner, Walter J. Gutjahr</i>	
Using Genetic Programming to Improve Software Effort Estimation Based on General Data Sets	2477
<i>Martin Lefley, Martin J. Shepperd</i>	
The State Problem for Evolutionary Testing	2488
<i>Phil McMinn, Mike Holcombe</i>	
Modeling the Search Landscape of Metaheuristic Software Clustering Algorithms	2499
<i>Brian S. Mitchell, Spiros Mancoridis</i>	

Search Based Software Engineering – Posters

Search Based Transformations	2511
<i>Deji Fatiregun, Mark Harman, Robert Hierons</i>	
Finding Building Blocks for Software Clustering	2513
<i>Kiarash Mahdavi, Mark Harman, Robert Hierons</i>	

Author Index

Swarms in Dynamic Environments

T.M. Blackwell

Department of Computer Science, University College London, Gower Street, London, UK
tim.blackwell@ieee.org

Abstract. Charged particle swarm optimization (CPSO) is well suited to the dynamic search problem since inter-particle repulsion maintains population diversity and good tracking can be achieved with a simple algorithm. This work extends the application of CPSO to the dynamic problem by considering a bi-modal parabolic environment of high spatial and temporal severity. Two types of charged swarms and an adapted neutral swarm are compared for a number of different dynamic environments which include extreme ‘needle-in-the-haystack’ cases. The results suggest that charged swarms perform best in the extreme cases, but neutral swarms are better optimizers in milder environments.

1 Introduction

Particle Swarm Optimization (PSO) is a population based optimization technique inspired by models of swarm and flock behavior [1]. Although PSO has much in common with evolutionary algorithms, it differs from other approaches by the inclusion of a solution (or particle) velocity. New potentially good solutions are generated by adding the velocity to the particle position. Particles are connected both temporally and spatially to other particles in the population (swarm) by two accelerations. These accelerations are spring-like: each particle is attracted to its previous best position, and to the global best position attained by the swarm, where ‘best’ is quantified by the value of a state function at that position. These swarms have proven to be very successful in finding global optima in various static contexts such as the optimization of certain benchmark functions [2].

The real world is rarely static, however, and many systems will require frequent re-optimization due to a dynamic environment. If the environment changes slowly in comparison to the computational time needed for optimization (i.e. to within a given error tolerance), then it may be hoped that the system can successfully re-optimize. In general, though, the environment may change on any time-scale (temporal severity), and the optimum position may change by any amount (spatial severity). In particular, the optimum solution may change discontinuously, and by a large amount, even if the dynamics are continuous [3]. Any optimization algorithm must therefore be able to both detect and respond to change.

Recently, evolutionary techniques have been applied to the dynamic problem [4, 5, 6]. The application of PSO techniques is a new area and results for environments of low spatial severity are encouraging [7, 8]. CPSO, which is an extension of PSO, has also been applied to more demanding environments, and found to outperform the conventional PSO [9, 10]. However, PSO can be improved or adapted by incorporating change detecting mechanisms [11]. In this paper we compare adaptive PSO with CPSO for various dynamic environments, some of which are severe both spatially and temporally. In order to do this, we use a model which enables simple testing for the three types of dynamism defined by Eberhart, Shi and Hu [7, 11].

2 Background

The problem of optimization within a general and unknown dynamic environment can be approached by a classification of the nature of the environment and a quantification of the difficulty of the problem. Eberhart, Shi and Hu [7, 11] have defined three types of dynamic environment. In type I environments, the optimum position $\mathbf{x}_{opt}$, defined with respect to a state function f , is subject to change. In type II environments, the value of f at $\mathbf{x}_{opt}$ varies and, in type III environments, both $\mathbf{x}_{opt}$ and $f(\mathbf{x}_{opt})$ may change. These changes may occur at any time, or they may occur at regular periods, corresponding, for example, to a periodic sensing of the environment. Type I problems have been quantified with a severity parameter s , which measures the jump in optimum location.

Previous work on PSO in dynamic environments has focused on periodic type I environments of small spatial severity. In these mild environments, the optimum position changes by an amount $s\mathbf{I}$, where $\mathbf{I}$ is the unit vector in the n -dimensional search space of the problem. Here, ‘small’ is defined by comparison with the dynamic range of the internal variables $\mathbf{x}$. Comparisons of CPSO and PSO have also been made for severe type I environments, where s is of the order of the dynamic range [9]. In this work, it was observed that the conventional PSO algorithm has difficulty adjusting in spatially severe environments due to over specialization. However, the PSO can be adapted by incorporating a change detection and response algorithm [11].

A different extension of PSO, which solves the problem of change detection and response, has been suggested by Blackwell and Bentley [10]. In this extension (CPSO), some or all of the particles have, in analogy with electrostatics, a ‘charge’. A third collision-avoiding acceleration is added to the particle dynamics, by incorporating electrostatic repulsion between charged particles. This repulsion maintains population diversity, enabling the swarm to automatically detect and respond to change, yet does not diminish greatly the quality of solution. In particular, it works well in certain spatially severe environments [9].

Three types of particle swarm can be defined: neutral, atomic and fully-charged. The neutral swarm has no charged particles and is identical with the conventional PSO. Typically, in PSO, there is a progressive collapse of the swarm towards the best position, with each particle moving with diminishing amplitude around the best posi-

tion. This ensures good exploitation, but diversity is lost. However, in a swarm of ‘charged’ particles, there is an additional collision avoiding acceleration. Animations for this swarm reveal that the swarm maintains an extended shape, with the swarm centre close to the optimum location [9, 10]. This is due to the repulsion which works against complete collapse. The diversity of this swarm is high, and response to environment change is quick. In an ‘atomic’ swarm, 50% of the particles are charged and 50% are neutral. Animations show that the charged particles orbit a collapsing nucleus of neutral particles, in a picture reminiscent of an atom. This type of swarm therefore balances exploration with exploitation. Blackwell and Bentley have compared neutral, fully charged and atomic swarms for a type-I time-dependent dynamic problem of high spatial severity [9]. No change detection mechanism is built into the algorithm. The atomic swarm performed best, with an average best values of f some six orders of magnitude less than the worst performer (the neutral swarm).

One problem with adaptive PSO [11], is the arbitrary nature of the algorithm (there are two detection methods and eight responses) which means that specification to a general dynamic environment is difficult. Swarms with charge do not need any adaptive mechanisms since they automatically maintain diversity. The purpose of this paper is to test charged swarms against a variety of environments, to see if they are indeed generally applicable without modification.

In the following experiments we extend the results obtained above by considering time-independent problems that are both spatially and *temporally* severe. A model of a general dynamic environment is introduced in the next section. Then, in section 4, we define the CPSO algorithm. The paper continues with sections on experimental design, results and analysis. The results are collecting together in a concluding section.

3 The General Dynamic Search Problem

The dynamic search problem is to find $\mathbf{x}_{opt}$ for a state function $f(\mathbf{x}, \mathbf{u}(t))$ so that $f(\mathbf{x}_{opt}, t) \equiv f_{opt}$ is the instantaneous global minimum of f . The state variables are denoted $\mathbf{x}$ and the influence of the environment is through a (small) number of control variables $\mathbf{u}$ which may vary in time. No assumptions are made about the continuity of $\mathbf{u}(t)$, but note that even smooth changes in $\mathbf{u}$ can lead to discontinuous change in $\mathbf{x}_{opt}$. (In practice a sufficient requirement may be to find a good enough approximation to $\mathbf{x}_{opt}$ i.e. to optimize f to within some tolerance δf in timescales δt . In this case, precise tracking of $\mathbf{x}_{opt}$ may not be necessary.)

This paper proposes a simple model of a dynamic function with moving local minima,

$$f = \min \{f_1(\mathbf{x}, \mathbf{u}_1), f_2(\mathbf{x}, \mathbf{u}_2), \dots, f_m(\mathbf{x}, \mathbf{u}_m)\} \quad (1)$$

where the control variables $\mathbf{u}_a = \{\mathbf{x}_a, h_a^2\}$ are defined so that f_a has a single minimum at $\mathbf{x}_a$, with an optimum value $h_a^2 \geq 0$ at $f_a(\mathbf{x}_a)$. If the functions f_a themselves have individual dynamics, f can be used to model a general dynamic environment.

A convenient choice for f_a , which allows comparison with other work on dynamic search with swarms [4, 7, 8, 9, 11], is the parabolic or sphere function in n dimensions,

$$f_a = \sum_{i=1}^n (x_i - x_{ai})^2 + h_a^2 \quad (2)$$

which differs from De Jong's $f1$ function [12] by the inclusion of a height offset h_a and a position offset x_{ai} . This model satisfies Branke's conditions for a benchmark problem (simple, easy to describe and analyze, and tunable) and is in many respects similar to his "moving peaks" benchmark problem, except that the widths of each optimum are not adjustable, and in this case we seek a *minimization* ("moving valleys") [6]. This simple function is easy to optimize with conventional methods in the static mono-modal case. However the problem becomes more acute as the number m of moving minima increases.

Our choice of f also suggests a simple interpretation. Suppose that all h_a are zero. Then f_a is the Euclidean 'squared distance' between vectors $\mathbf{x}$ and $\mathbf{x}_a$. Each local optimum position $\mathbf{x}_a$ can be regarded as a 'target'. Then, f is the squared distance of the nearest 'target' from the set $\{\mathbf{x}_a\}$ to $\mathbf{x}$. Suppose now that the vectors $\mathbf{x}$ are actually projections of vectors $\mathbf{y}$ in R^{n+1} , so that $\mathbf{y} = (\mathbf{x}, 0)$ and targets $\mathbf{y}_a$ have components $(\mathbf{x}_a, h_a)$ in this higher dimensional space. In other words, h_a are height offsets in the $n+1$ th dimension. From this perspective, f is still the squared distance to the nearest target, except that the system is restricted to R^n . For example, suppose that $\mathbf{x}$ is the 2-dimensional position vector of a ship, and $\{\mathbf{x}_a\}$ are a set of targets scattered on the sea bed at depths $\{h_a\}$. Then the square root of f at any time is the distance to the closest target and the depth of the shallowest object is $\sqrt{f(\mathbf{x}_{opt})}$. The task for the ship's navigator is to position the ship at $\mathbf{x}_{opt}$, directly over the shallowest target, given that all the targets are in independent motion along an uneven sea bed.

Since no assumptions have been made about the dynamics of the environment, the above model describes the situation where the change can occur at any time. In the periodic problem, we suppose that the control variables change simultaneously at times t_i and are held fixed at $\mathbf{u}_i$ for the corresponding intervals $[t_i, t_{i+1}]$:

$$\mathbf{u}(t) = \sum_i (\Theta(t_i) - \Theta(t_{i+1})) \mathbf{u}_i \quad (3)$$

where $\Theta(t)$ is the unit step function.

The PSO and CPSO experiments of [9] and [11] are time-dependent type I experiments with a single minimum at $\mathbf{x}_i$ and with $h_i = 0$. The generalization to more difficult type I environments is achieved by introducing more local minima at positions $\mathbf{x}_a$, but fixing the height offsets h_a . Type II environments are easily modeled by fixing the positions of the targets, but allowing h_a to change at the end of each period. Finally, a type III environment is produced by periodically changing both $\mathbf{x}_a$ and h_a .

Severity is a term that has been introduced to characterize problems where the optimum position changes by a fixed amount s at a given number of iterations [4, 7]. In [7, 11] the optimum position changes by small increments along a line. However,

Blackwell and Bentley have considered more severe dynamic systems whereby the optimum position can jump randomly within a target cube T which is of dimension equal to twice the dynamic range v_{max} [9]. Here severity is extended to include dynamic systems where the target jumps may be for periods of very short duration.

4 PSO and CPSO Algorithms

Table 1 shows the particle update algorithm. The PSO parameters g , g_2 and w govern convergence. The electrostatic acceleration a_i , parameterized by p_{core} , p and Q_i , is

$$a_i = \sum_{j \neq i} \frac{Q_i Q_j}{|r_{ij}|^3} r_{ij}, \quad p_{core} < r_{ij} < p, \quad r_{ij} = x_i - x_j \quad (4)$$

The PSO and CPSO search algorithm is summarized below in Table 2. To begin, a swarm of M particles, where each particle has n -dimensional position and velocity vectors $\{x_i, v_i\}$, is randomized in the box $T = D^n = [-v_{max}, v_{max}]^n$ where D is the ‘dynamic range’ and v_{max} is the clamping velocity. A set of period durations $\{t_i\}$ is chosen; these are either fixed to a common duration, or chosen from a uniform random distribution. A single iteration is a single pass through the loop in Table 2.

Denoting the best value position and value found by the swarm as x_{gb} and f_{gb} , change detection is simply invoked by comparing $f(x_{gb})$ with f_{gb} . If these are not equal, the inference is that f has changed since f_{gb} was last evaluated. The response is to re-randomize a fraction of the swarm in T , and to re-set f_{gb} to $f(x_{gb})$. The detection and response algorithm is only applied to neutral swarms.

The best position attained by a particle, $x_{pb,i}$, is updated by comparing $f(x_i)$ with $f(x_{pb,i})$: if $f(x_i) < f(x_{pb,i})$, then $x_{pb,i} \leftarrow x_i$. Any new $x_{pb,i}$ is then tested against x_{gb} , and a replacement is made, so that at each particle update $f(x_{gb}) = \min\{f(x_{pb,i})\}$. This specifies *update best(i)*.

Table 1. The particle update algorithm

<i>update particle(i)</i>
$v_i \leftarrow wv_i + g_1(x_{pb,i} - x_i) + g_2(x_{gb} - x_i) + a_i$
if $ v_i > v_{max}$
$v_i \leftarrow (v_{max} / v_i) v_i$
$x_i \leftarrow x_i + v_i$

Table 2. Search algorithm for charged and neutral particle swarm optimization

<i>(C)PSO search</i>
initialize swarm $\{ \mathbf{x}_p, \mathbf{v}_i \}$ and periods $\{t_j\}$ loop: if $t = t_j$ <i>update function</i> if (neutral swarm) <i>detect</i> and respond to change for $i = 1$ to M <i>update best</i> (i) <i>update particle</i> (i) endfor $t \leftarrow t + 1$ until stopping criterion is met

5 Experiment Design

Twelve experiments of varying severity were conceived, for convenience arranged in three groups. The parameters and specifications for these experiments are summarized in Tables 3 and 4. In each experiment, the dynamic function has two local minima at $\mathbf{x}_a$, $a = 1, 2$; the global minimum is at $\mathbf{x}_2$. The value of f at $\mathbf{x}_1$ is fixed at 100 in all experiments. The duration of the function update periods, denoted Δ , is either fixed at 100 iterations, or is a random integer between 1 and 100. (For simplicity, random variables drawn from uniform distribution with limits a, b will be denoted $x \sim [a, b]$ (continuous distribution) and $x \sim [a \dots b]$ (discrete distribution).

In the first group (A) of experiments, numbers 1 – 4, $\mathbf{x}_2$ is moved randomly in T (‘spatially severe’) or is moved randomly in a smaller box $0.1T$. The optimum value, $f(\mathbf{x}_2)$, is fixed at 0. These are all type I experiments, since the optimum location moves, but the optimum value is fixed. Experiments 3 and 4 repeat the conditions of 1 and 2 except that $\mathbf{x}_2$ moves at random intervals $\sim [1 \dots 100]$ (temporally severe).

Experiments 5 – 8 (Group B) are type II environments. In this case, $\mathbf{x}_1$ and $\mathbf{x}_2$ are fixed at $\pm \mathbf{r}$, along the body diagonal of T , where $\mathbf{r} = (\mathbf{v}_{\max}/3) (1, 1, 1)$. However, $f(\mathbf{x}_2)$ varies, with $h_2 \sim [0, 1]$, or $h_2 \sim [0, 100]$. Experiments 7 and 8 repeat the conditions of 5 and 6 but for high temporal severity.

In the last group (C) of experiments (9 – 12), both $\mathbf{x}_1$ and $\mathbf{x}_2$ jump randomly in T . In the type III case, experiments 11 and 12, $f(\mathbf{x}_2)$ varies. For comparison, experiments 9

and 10 duplicate the conditions of 11 and 12, but with fixed $f(\mathbf{x}_2)$. Experiments 10 and 12 are temporally severe versions of 9 and 11.

Each experiment, of 500 periods, was performed with neutral, atomic (i.e. half the swarm is charged) and fully charged swarms (all particles are charged) of 20 particles ($M = 20$). In addition, the experiments were repeated with a random search algorithm, which simply, at each iteration, randomizes the particles within T . A spatial dimension of $n = 3$ was chosen. In each run, whenever random numbers are required for target positions, height offsets and period durations, the same sequence of pseudo-random numbers is used, produced by separately seeded generators. The initial swarm configuration is random in T , and the same configuration is used for each run.

Table 3. Spatial, electrostatic and PSO Parameters

Spatial				Electrostatic			PSO	
v_{max}	n	M	T	p_{core}	p	Q_i	g_1, g_2	w
32	3	20	$[-32, 32]^3$	1	$2\sqrt{3}v_{max}$	16	$\sim[0, 1.49]$	$\sim[0.5, 1]$

Table 4. Experiment Specifications

Group	Expt	Targets $\{x_i, x_j\}$	Local Opt $\{f(x_i), f(x_j)\}$	Period Δ
A	1	$\{O, \sim 0.1T\}$	$\{100, 0\}$	100
	2	$\{O, \sim T\}$		$\sim[1, 100]$
	3	$\{O, \sim 0.1T\}$		
	4	$\{O, \sim T\}$		
B	5	$\{O-r, O+r\}$	$\{100, \sim[0, 1]\}$	100
	6		$\{100, \sim[0, 100]\}$	
	7		$\{100, \sim[0, 1]\}$	$\sim[1, 100]$
	8		$\{100, \sim[0, 100]\}$	
C	9	$\{\sim T, \sim T\}$	$\{100, 0\}$	100
	10			$\sim[1, 100]$
	11		$\{100, \sim[0, 100]\}$	100
	12			$\sim[1, 100]$

The search (C)PSO algorithm has a number of parameters (Table 3) which have been chosen to correspond to the values used in previous experiments [5, 9, 11]. These choices agree with Clerc's analysis for convergence [13]. The spatial and electrostatic parameters are once more chosen for comparison with previous work on charged particle swarms [9]. An analysis that explains the choice of the electrostatic parameters is

given in [14]. Since we are concerned with very severe environments, the response strategy chosen here is to randomize the positions of 50% of the swarm [11]. This also allows for comparisons with the atomic swarm which maintains a diverse population of 50% of the swarm.

6 Results and Analysis

The chief statistic is the ensemble average best value, $\langle f(\mathbf{x}_2) - f_{gb} \rangle$; this is positive and bounded by zero. A further statistic, the number of ‘successes’, $n_{\text{successes}}$, was also collected to aid analysis. Here, the search is deemed a success if $\mathbf{x}_{gb}$ is closer, at the end of each period, to target 2 (which always has the lower value of f) than it is to target 1. The results for the three swarms and for random search are shown in Figs 1 and 2. The light grey boxes in Figure 1, experiment 6, indicate an upper bound to the ensemble average due to the precision of the floating-point representation: for these runs, $f(\mathbf{x}_2) - f_{gb} = 0$ at the end of each period, but this is an artifact of the finite-precision arithmetic.

Group A. Figure 1 shows that all swarms perform better than random search except for the neutral swarm in spatially severe environments (2 and 4) and the atomic swarm in a spatially and temporally severe environment (4). In the least severe environment (1), the neutral swarm performs very well, confirming previous results. This swarm has the least diversity and the best exploitation. The order of performance for this experiment reflects the amount of diversity; neutral (least diversity, best), atomic, fully charged, and random (most diversity, worst). When environment 1 is made temporally severe (3), all swarms have similar performance and are better than random search. The implication here is that on average the environment changes too quickly for the better exploitation properties of the neutral swarm to become noticeable. Experiments 2 and 4 repeat the conditions of 1 and 2, except for higher spatial severity. Here the order of performance amongst the swarms is in increasing order of diversity (fully charged best and neutral worst). The reason for the poor performance of the neutral swarm in environments 2 and 4 can be inferred from the success data. The success rate of just 5% and ensemble average close to 100 ($= f(\mathbf{x}_1)$) suggests that the neutral swarm often gets stuck in the false minimum at $\mathbf{x}_1$. Since f_{gb} does not change at $\mathbf{x}_1$, the adapted swarm cannot register change, does not randomize, and so is unlikely to move away from $\mathbf{x}_1$ until $\mathbf{x}_2$ jumps to a nearby location. In fact the neutral swarm is worse than random search by an order of magnitude. Only the fully charged swarm out-performs random search appreciably for the spatially severe type I environments (2 and 4) and this margin diminishes when the environment is temporally severe too.

Group B. Throughout this group, all swarms are better than random and the number of successes shows that there are no problems with the false minimum. The swarm with the least diversity and best exploitation (neutral) does best since the optimum location

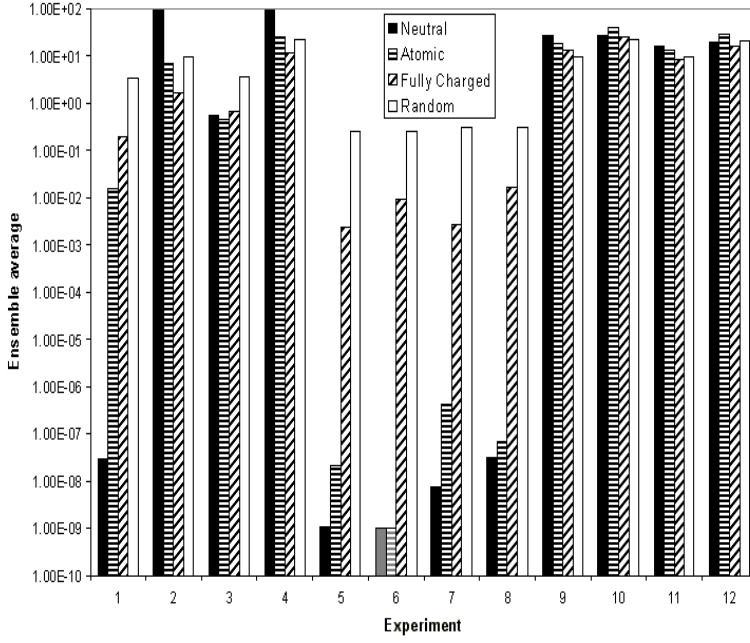


Fig. 1. Ensemble average $\langle f(x_s) - f_{ob} \rangle$ for all experiments

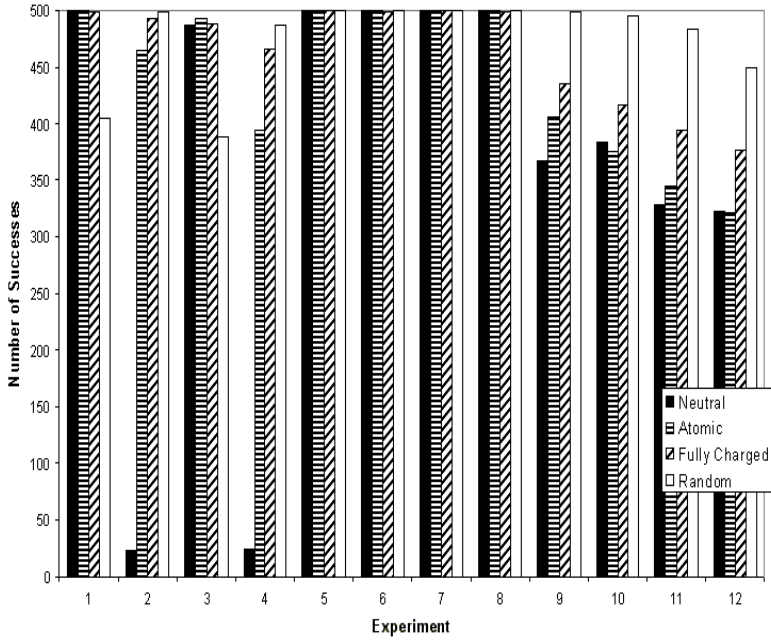


Fig. 2. Number of successes $n_{successes}$ for all experiments

does not change from period to period. The effect of increasing temporal severity can be seen by comparing 7 to 5 and 8 to 6. Fully charged and random are almost unaffected by temporal severity in these type II environments, but the performance of the neutral and atomic swarms worsens. Once more the explanation for this is that these are the only two algorithms which can significantly improve their best position over time because only these two contain neutral particles which can converge unimpeded on the minimum. This advantage is lessened when the average time between jumps is decreased. The near equality of ensemble averages for random search in 5 and 6, and again in 7 and 8, is due to the fact that random search is not trying to improve on a previous value – it just depends on the closest randomly generated points to x_2 during any period. Since x_1 and x_2 are fixed, this can only depend on the period size and not on $f(x_2)$.

Group C. The ensemble averages for the four experiments in this group (9-12) are broadly similar but the algorithm with the most successes in each experiment is random search. However random search is not able to exploit any good solution, so although the swarms have more failures, they are able to improve on their successes producing ensemble averages close to random search. In experiments 9 and 10, which are type I cases, all swarms perform less well than random search. These two experiments differ from environments 2 and 4, which are also spatially severe, by allowing the false minimum at x_i to jump as well. The result is that the performance of the neutral swarm improves since it is no longer caught by the false minimum at x_i ; the number of successes improves from less than 25 in 2 and 4, to over 350 in 9 and 10. In experiments 11 and 12 (type III) when f_{opt} changes in each period, the fully charged swarm marginally out-performs random search. It is worth noting that 12 is a very extreme environment: either minimum can jump by arbitrary amounts, on any time scale, and with the minimum value varying over a wide range. One explanation for the poor performance of all swarms in 9 and 10 is that there is a higher penalty ($\langle f(x_i) - f_{opt} \rangle = 100$) for getting stuck on the false minimum at x_i , than the corresponding penalty in 11 and 12 ($\langle f(x_i) - f_{opt} \rangle = 50$). The lower success rate for all swarms compared to random search supports this explanation.

7 Conclusions

A dynamic environment can present numerous challenges for optimization. This paper has presented a simple mathematical model which can represent dynamic environments of various types and severity. The neutral particle swarm is a promising algorithm for these problems since it performs well in the static case, and can be adapted to respond to change. However, one draw back is the arbitrary nature of the detection and response algorithms. Particle swarms with charge need no further adaptation to cope with the dynamic scenario due to the extended swarm shape. The neutral and two charged particle swarms have been tested, and compared with random search, with twelve environments which are classified by type. Some of these environments are extreme, both in the spatial as well as the temporal domain.

The results support the intuitive idea that type II environments (those in which the optimum location is fixed, but the optimum value may vary) present few problems to evolutionary methods since a population diversity is not important. In fact the algorithm with the lowest diversity performed best. Increasing temporal severity diminishes the performance of the two swarms with neutral particles, but does not affect the fully charged swarm.

However, environments where the optimum location can change (types I and III) are much harder to deal with, especially when the optimum jumps can be to an arbitrary point within the search space, and can happen at very short notice. This is the dynamic equivalent of the needle in a haystack problem. A type I environment has been identified which poses considerable problems for the adapted PSO algorithm: a stationary false minimum and a mobile true minimum with large spatial severity. There is a tendency for the neutral swarm to become trapped by the false minimum. In this case, the fully charged swarm is the better option.

Finally, the group C environments proved to be very challenging for all swarms. These environments are distinguished by two spatially severe minima with a large difference in function value at these minima. In other words, there is a large penalty for finding the false minimum rather than the true minimum. All swarms struggled to improve upon random search because of this trap.

Despite this, all swarms have been shown, for dynamic parabolic functions, to offer results comparable to random search in the worst cases, and considerably better than random in the more benign situations. As with static search problems, if some prior knowledge of the dynamics is known, a preferable algorithm can be chosen. According to the classification of Eberhart and Wu [7, 11], and for the examples studied here, the adapted neutral swarm is the best performer for mild type I and II environments. However, it can be easily fooled in type I and III environments where a false minimum is also dynamic. In this case, the charged swarms are better choices. As the environment becomes more extreme, charge, which is a diversity increasing parameter, becomes more useful. In short, if nothing is known about an environment, the fully charged swarm has the best average performance.

It is possible that different adaptations to the neutral swarm can lead to better performance in certain environments, but it remains to be seen if there is a single adaptation which works well over a range of environments. On the other hand, the charged swarm needs no further modification since the collision avoiding accelerations ensure exploration the space around a solution.

References

1. Kennedy J. and Eberhart, R.C.: Particle Swarm Optimization. Proc of the IEEE International Conference on Neural Networks IV (1995) 1942–1948
2. Eberhart R.C. and Shi Y.: Particle swarm optimization: Developments, applications and resources. Proc Congress on Evolutionary Computation (2001) 81–86
3. Saunders P.T.: An Introduction to Catastrophe Theory. Cambridge University Press (1980)

4. Angeline P.J.: Tracking extrema in dynamic environments. *Proc Evolutionary Programming IV*. (1998) 335–345
5. Bäck T.: On the behaviour of evolutionary algorithms in dynamic environments. *Proc Int. Conf. on Evolutionary Computation*. (1998) 446–451
6. Branke J.: Evolutionary algorithms for changing optimization problems. *Proc Congress on Evolutionary Computation*. (1999) 1875–1882
7. Eberhart R.C. and Shi Y.: Tracking and optimizing dynamic systems with particle swarms. *Proc Congress on Evolutionary Computation*. (2001) 94–97
8. Carlisle A. and Dozier G.: Adapting particle swarm optimization to dynamic environments. *Proc of Int Conference on Artificial Intelligence*. (2000) 429–434
9. Blackwell and Bentley P.J.: Dynamic search with charged swarms. *Proc Genetic and Evolutionary Computation Conference*. (2002) 19–26
10. Blackwell and Bentley P.J.: Don't push me! Collision avoiding swarms. *Proc Congress on Evolutionary Computation*. (2002) 1691–1696
11. Hu X. and Eberhart R.C.: Adaptive particle swarm optimization: detection and response to dynamic systems. *Proc Congress on Evolutionary Computation*. (2002) 1666–1670
12. De Jong K: An analysis of the behavior of a class of genetic adaptive systems. PhD thesis, University of Michigan (1975)
13. Clerc M.: The swarm and the queen: towards a deterministic and adaptive particle swarm optimization. *Proc Congress on Evolutionary Computation*. (1999) 1951–1957
14. Blackwell and Bentley P.J.: Improvised Music with Swarms, *Proc Congress on Evolutionary Computation*. (2002) 1462–1467

The Effect of Natural Selection on Phylogeny Reconstruction Algorithms

Dehua Hang¹, Charles Ofria¹, Thomas M. Schmidt², and Eric Tornø¹

¹Department of Computer Science & Engineering
Michigan State University, East Lansing, MI 48824 USA

²Department of Microbiology and Molecular Genetics
Michigan State University, East Lansing, MI 48824 USA
{hangdehu, ofria, tschmidt, tornø}@msu.edu

Abstract. We study the effect of natural selection on the performance of phylogeny reconstruction algorithms using Avida, a software platform that maintains a population of digital organisms (self-replicating computer programs) that evolve subject to natural selection, mutation, and drift. We compare the performance of neighbor-joining and maximum parsimony algorithms on these Avida populations to the performance of the same algorithms on randomly generated data that evolve subject only to mutation and drift. Our results show that natural selection has several specific effects on the sequences of the resulting populations, and that these effects lead to improved performance for neighbor-joining and maximum parsimony in some settings. We then show that the effects of natural selection can be partially achieved by using a non-uniform probability distribution for the location of mutations in randomly generated genomes.

1 Introduction

As researchers try to understand the biological world, it has become clear that knowledge of the evolutionary relationships and histories of species would be an invaluable asset. Unfortunately, nature does not directly track such changes, and so such information must be inferred by studying extant organisms. Many algorithms have been crafted to reconstruct phylogenetic trees - dendrograms in which species are arranged at the tips of branches, which are then linked successively according to common evolutionary ancestors. The input to these algorithms are typically traits of extant organisms such as gene sequences. Often, however, the phylogenetic trees produced by distinct reconstruction algorithms are different, and there is no way of knowing which, if any, is correct.

In order to determine which reconstruction algorithms work best, methods for evaluating these algorithms need to be developed. As documented by Hillis [1], four principal methods have been used for assessing phylogenetic accuracy: working with real lineages with known phylogenies, generating artificial data using computer simulations, statistical analyses, and congruence studies. These last two methods tend to focus on specific phylogenetic estimates; that is, they attempt to provide independent confirmations or probabilistic assurances for a specific result rather than

evaluate the general effectiveness of an algorithm. We focus on the first two methods, which are typically used to evaluate the general effectiveness of a reconstruction algorithm: computer simulations [2] and working with lineages with known phylogenies [3].

In computer simulations, data is generated according to a specific model of nucleotide or amino acid evolution. The primary advantages of the computer simulation technique are that the correct phylogeny is known, data can be collected with complete accuracy and precision, and vast amounts of data can be generated quickly. One commonly used computer simulation program is *seq-gen* [4]. Roughly speaking, *seq-gen* takes as input an ancestral organism, a model phylogeny, and a nucleotide substitution model and outputs a set of taxa that conforms to the inputs. Because the substitution model and the model phylogeny can be easily changed, computer simulations can generate data to test the effectiveness of reconstruction algorithms under a wide range of conditions.

Despite the many advantages of computer simulations, this technique suffers from a “credibility gap” due to the fact that the data is generated by an artificial process. That is, the sequences are never expressed and thus have no associated function. All genomic changes in such a model are the result of mutation and genetic drift; natural selection does not determine which position changes are accepted and which changes are rejected. Natural selection is only present via secondary relationships such as the use of a model phylogeny that corresponds to real data. For this reason, many biologists disregard computer simulation results.

Another commonly used evaluation method is to use lineages with known phylogenies. These are typically agricultural or laboratory lineages for which records have been kept or experimental phylogenies generated specifically to test phylogenetic methods. Known phylogenies overcome the limitation of computer simulations in that all sequences are real and do have a relation to function. However, working with known phylogenies also has its limitations. As Hillis states, “Historic records of cultivated organisms are severely limited, and such organisms typically have undergone many reticulations and relatively little genetic divergence.” [1]. Thus, working with these lineages only allows the testing of reconstructions of phylogenies of closely related organisms. Experimentally generated phylogenies were created to overcome this difficulty by utilizing organisms such as viruses and bacteria that reproduce very rapidly.

However, even research with experimentally generated lineages has its shortcomings. First, while the organisms are natural and evolving, several artificial manipulations are required in order to gather interesting data. For example, the mutation rate must be artificially increased to produce divergence and branches are forced by explicit artificial events such as taking organisms out of one petri dish and placing them into two others. Second, while the overall phylogeny may be known, the data captured is neither as precise nor complete as that with computer simulations. That is, in computer simulations, every single mutation can be recorded whereas with experimental phylogenies, only the major, artificially induced phylogenetic branch events can be recorded. Finally, even when working with rapidly reproducing organisms, significant time is required to generate a large amount of test data; far more time than when working with computer simulations.

Because of the limitations of previous evaluation methods, important questions about the effectiveness of phylogeny reconstruction algorithms have been ignored in

the past. One important question is the following: What is the effect of natural selection on the accuracy of phylogeny reconstruction algorithms?

Here, we initiate a systematic study of this question. We begin by generating two related data sets. In the first, we use a computer program that has the accuracy and speed of previous models, but also incorporates natural selection. In this system, a mutation only has the possibility of persisting if natural selection does not reject it. The second data set is generated with the same known phylogenetic tree structure as was found in the first, but this time all mutations are accepted regardless of the effect on the fitness of the resulting sequence (to mimic the more traditional evaluation methodologies). We then apply phylogeny reconstruction algorithms to the final genetic sequences in both data sets and compare the results to determine the effect of natural selection.

To generate our first data set, we use Avida, a digital life platform that maintains a population of digital organisms (i.e. programs) that evolve subject to mutation, drift, and natural selection. The true phylogeny is known because the evolution occurs in a computer in which all mutation events are recorded. On the other hand, even though Avida populations exist in a computer rather than in a petri dish or in nature, they are not simulations but rather are experiments with digital organisms that are analogous to experiments with biological organisms. We describe the Avida system in more detail in our methods section.

2 Methods

2.1 The Avida Platform [5]

The major difficulty in our proposed study is generating sequences under a variety of conditions where we know the complete history of all changes and the sequences evolve subject to natural selection, not just mutation and drift. We use the Avida system, an auto-adaptive genetic system designed for use as a platform in digital/artificial life research, for this purpose.

A typical Avida experiment proceeds as follows. A population of digital organisms (self-replicating computer programs with a Turing-complete genetic basis) is placed into a computational environment. As each organism executes, it can interact with the environment by reading inputs and writing outputs. The organisms reproduce by allocating memory to double their size, explicitly copying their genome (program) into the new space, and then executing a divide command that places the new copy onto one of the CPU's in the environment "killing" the organism that used to occupy that CPU. Mutations are introduced in a variety of ways. Here, we make the copy command probabilistic; that is, we can set a probability that the copy command fails by writing an arbitrary instruction rather than the intended instruction.

The crucial point is that during an Avida experiment, the population evolves subject to selective pressures. For example, in every Avida experiment, there is a selective pressure to reproduce quickly in order to propagate before being overwritten by another organism. We also introduce other selective pressures into the environment by rewarding organisms that perform specific computations by increasing the speed at which they can execute the instructions in their genome. For example, if the outputs produced by an organism demonstrate that the organism can

perform a Boolean logic operation such as “exclusive-or” on its inputs, then the organism and its immediate descendants will execute their genomes at twice their current rate. Thus there is selective pressure to adapt to perform environment-specific computations. Note that the rewards are not based on how the computation is performed; only the end product is examined. This leads to open-ended evolution where organisms evolve functionality in unanticipated ways.

2.2 Natural Selection and Avida

Digital organisms are used to study evolutionary biology as an independent form of life that shares no ancestry with carbon-based life. This approach allows general principles of evolution to be distinguished from historical accidents that are particular to biochemical life. As Wilke and Adami state, “In terms of the complexity of their evolutionary dynamics, digital organisms can be compared with biochemical viruses and bacteria”, and “Digital organisms have reached a level of sophistication that is comparable to that of experiments with bacteria or viruses” [6].

The limitation of working with digital organisms is that they live in an artificial world, so the conclusions from digital organism experiments are potentially an artifact of the particular choices of that digital world. But by comparing the results across wide ranges of parameter settings, as well as results from biochemical organisms and from mathematical theories, general principles can still be disentangled. Many important topics in evolutionary biology have been addressed by using digital organisms including the origins of biological complexity [7], and quasi-species dynamics and the importance of neutrality [8]. Some work has also compared biological systems with those of digital organisms, such as a study on the distribution of epistemic interactions among mutations [9], which was modeled on an earlier experiment with *E. coli* [10], and the similarity of the results were striking, supporting the theory that many aspects of evolving systems are governed by universal principles.

Avida is a well-developed digital organism platform. Avida organisms are self-replicating computer programs that live in, and adapt to, a controlled environment. Unlike other computational approaches to studying evolution (such as genetic algorithms or numerical simulations), Avida organisms must explicitly create a copy of their own genome to reproduce, and no particular genomic sequence is designated as the target or optimal sequence. Explicit and implicit mutations occur in Avida. Explicit mutations include point mutations incurred during the copy process and the random insertions and/or deletions of single instructions. Implicit mutations are the result of flawed copy algorithms. For example, an Avida organism might skip part of its genome during the replication, or replicate part of its genome more than once. The rates of explicit mutations can be controlled during the setup process, whereas implicit mutations cannot typically be controlled. Selection occurs because the environment in which the Avida organisms live is space limited. When a new organism is born, an older one is removed from the population.

2.3 Determining Correctness of a Phylogeny Reconstruction: The Four Taxa Case

Even when we know the correct phylogeny, it is not easy to measure the quality of a specific phylogeny reconstruction. A phylogeny can be thought of as an edge-weighted tree (or, more generally, an edge-weighted graph) where the edge weights correspond to evolutionary time or distance. Thus, a reconstruction algorithm should not only generate the correct topology or structure but also must generate the correct evolutionary distances.

Like many other studies, we simplify the problem by ignoring the edge weights and focus only on topology [11]. Even with this simplification, measuring correctness is not an easy problem. If the reconstructed topology is identical to the correct topology, then the reconstruction is correct. However, if the reconstructed topology is not identical, which will often be the case, it is not sufficient to say that the reconstruction is incorrect. There are gradations of correctness, and it is difficult to state that one topology is closer to the correct topology than a second one in many cases.

We simplify this problem so that there is an easy answer of right and wrong. We focus on reconstructing topologies based on populations with four taxa. With only four taxa, there really is only one decision to be made: Is A closest to B, C, or D? See the following diagram for an illustration of the three possibilities. Focusing on situations with only four taxa is a common technique used in the evaluation of phylogeny reconstruction algorithms [2,11,12].

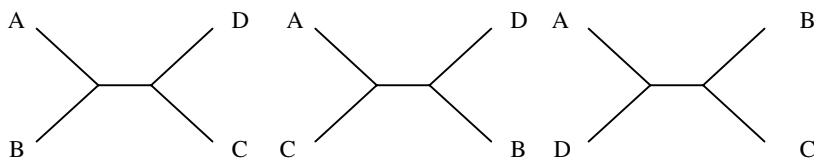


Fig. 1. Three possible topologies under four taxa model tree.

2.4 Generation of Avida Data

We generated Avida data in the following manner. First, we took a hand-made ancestor S1 and injected it into an environment E1 in which four simple computations were rewarded. The ancestor had a short copy loop and its genome was padded out to length 100 (from a simple 15-line self-replicator) with inert no-op instructions. The only mutations we allowed during the experiments were copy mutations and all size changes due to mis-copies were rejected; thus the lengths of all genome sequences throughout the execution are length 100. We chose to fix the length of sequences in order to eliminate the issue of aligning sequences. The specific length 100 is somewhat arbitrary. The key property is that it is enough to provide space for mutations and adaptations to occur given that we have disallowed insertions. All environments were limited to a population size of 3600. Previous work with avida (e.g. [16]) has shown that 3600 is large enough to allow for diversity while making large experiments practical.

After running for L1 updates, we chose the most abundant genotype S2 and placed S2 into a new environment E2 that rewarded more complex computations. Two computations overlapped with those rewarded by E1 so that S2 retained some of its fitness, but new computations were also rewarded to promote continued evolution. We executed two parallel experiments of S2 in E2 for 1.08×10^{10} cycles, which is approximately 10^4 generations. In each of the two experiments, we then sampled genotypes at a variety of times L2 along the line of descent from S2 to the most abundant genotype at the end of the execution. Let S3a-x denote the sampled descendant in the first experiment for $L2 = x$ while S3b-x denotes the same descendant in the second experiment.

Then, for each value x of L2, we took S3a-x and S3b-x and put them each into a new environment E3 that rewards five complex operations. Again, two rewarded computations overlapped with the computations rewarded by E2 (and there was no overlap with E1), and again, we executed two parallel experiments for each organism for a long time. In each of the four experiments, we then sampled genotypes at a variety of times L3 along the line of descent from S3a-x or S3b-x to the most abundant genotype at the end of the execution. For each value of L3, four taxa A, B, C and D were used for reconstruction. This experimental procedure is illustrated in the following diagram. Organisms A and B share the same ancestor S3a-x while organisms C and D share the same ancestor S3b-x.

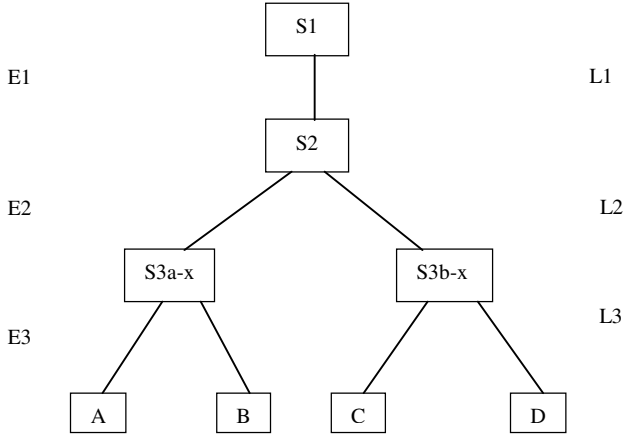


Fig. 2. Experimental procedure diagram.

We varied our data by varying the sizes of L2 and L3. For L2, we used values 3, 6, 10, 25, 50, and 100. For L3, we used values 3, 6, 10, 25, 100, 150, 200, 250, 300, 400, and 800. We repeated the experimental procedure 10 times.

The *tree structures* that we used for reconstruction were symmetric (they have the shape implied by Fig. 1). The *internal edge length* of any tree structure is twice the value of L2. The *external edge length* of any tree structure is simply L3. With six values of L2 and eleven values of L3, we used 66 different tree structures with 10 distinct copies of each tree structure.

2.5 Generation of Random Data

We developed a random data generator similar to seq-gen in order to produce data that had the same phylogenetic topology as the Avida data, but where the evolution occurred without any natural selection. Specifically, the generator took as input the known phylogeny of the corresponding Avida experiment, including how many mutations occurred along each branch of the phylogenetic tree, as well as the ancestral organism S2 (we ignored environment E1 as its sole purpose was to distance ourselves from the hand-written ancestral organism S1). The mutation process was then simulated starting from S2 and proceeding down the tree so that the number of mutations between each ancestor/descendant is identical to that in the corresponding *Avida* phylogenetic tree. The mutations, however, were random (no natural selection) as the position of the mutation was chosen according to a fixed probability distribution, henceforth referred to as the *location probability distribution*, and the replacement character was chosen uniformly at random from all different characters. In different experiments, we employed three distinct location probability distributions. We explain these three different distributions and our rationale for choosing them in Section 3.3. We generated 100 copies of each tree structure in our experiments.

2.6 Two Phylogeny Reconstruction Techniques (NJ, MP)

We consider two phylogeny reconstruction techniques in this study.

Neighbor-Joining. *Neighbor-joining (NJ)* [13,14] was first presented in 1987 and is popular primarily because it is a polynomial-time algorithm, which means it runs reasonably quickly even on large data sets. *NJ* is a distance-based method that implements a greedy strategy of repeatedly clustering the two closest clusters (at first, a pair of leaves; thereafter entire subtrees) with some optimizations designed to handle non-ultrametric data.

Maximum Parsimony. *Maximum parsimony (MP)* [15] is a character-based method for reconstructing evolutionary trees that is based on the following principle. Of all possible trees, the most parsimonious tree is the one that requires the fewest number of mutations. The problem of finding an *MP* tree for a collection of sequences is NP-hard and is a special case of the Steiner problem in graph theory. Fortunately, with only four taxa, computing the most parsimonious tree can be done rapidly.

2.7 Data Collection

We assess the performance of *NJ* and *MP* as follows. If *NJ* produces the same tree topology as the correct topology, it receives a score of 1 for that experiment. For each tree structure, we summed together the scores obtained by *NJ* on all copies (10 for Avida data, 100 for randomly generated data) to get *NJ*'s score for that tree structure.

Performance assessment was more complicated for *MP* because there are cases where multiple trees are equally parsimonious. In such cases, *MP* will output all of the most parsimonious trees. If *MP* outputs one of the three possible tree topologies (given that we are using four taxa for this evaluation) and it is correct, then *MP* gets a

score of 1 for that experiment. If *MP* outputs two tree topologies and one of them is correct, then *MP* gets a score of 1/2 for that experiment. If *MP* outputs all three topologies, then *MP* gets a score of 1/3 for that experiment. If *MP* fails to output the correct topology, then *MP* gets a score of 0 for that experiment. Again, we summed together the scores obtained by *MP* on all copies of the same tree structure (10 for Avida data, 100 for random data) to get *MP*'s score on that tree structure.

3 Results and Discussions

3.1. Natural Selection and Its Effect on Genome Sequences

Before we can assess the effect of natural selection on phylogeny reconstruction algorithms, we need to understand what kind of effect natural selection will have on the sequences themselves. We show two specific effects of natural selection.

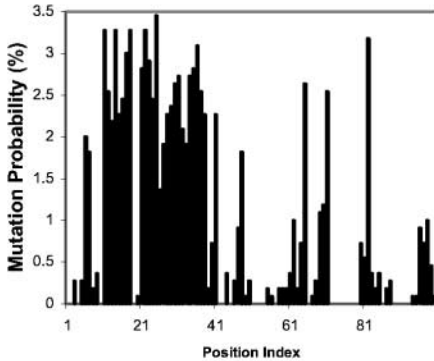


Fig. 3. Location probability distribution from one Avida run (length 100). Probability data are normalized to their percentage.

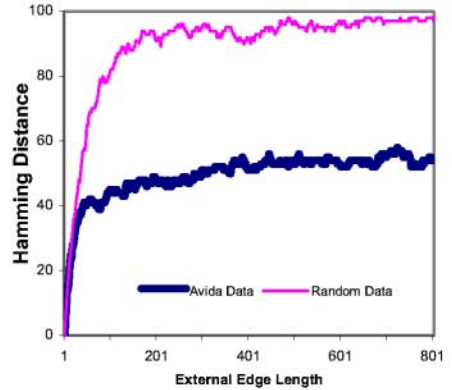


Fig. 4. Hamming distances between branch A and B from Avida data and randomly generated data. Internal edge length is 50.

We first show that the location probability distribution becomes non-uniform when the population evolves with natural selection. In a purely random model, each position is equally likely to mutate. However, with natural selection, some positions in the genome are less subject to accepted mutations than others. For example, mutations in positions involved in the copy loop of an Avida organism are typically detrimental and often lethal. Thus, accepted mutations in these positions are relatively rare compared to other positions. Fig. 2 shows the non-uniform position mutation probability distribution from a typical Avida experiment. This data captures the frequency of mutations by position in the line of descent from the ancestor to the most abundant genotype at the end of the experiment. While this is only one experiment, similar results apply for all of our experiments. In general, we found roughly three types of positions: fixed positions with no accepted mutations in the population

(accepted mutation rate = 0%); stable positions with a low rate of accepted mutations in the population (accepted mutation rate < 1%), and volatile positions with a high rate of accepted mutations (accepted mutation rate > 1%).

Because some positions are stable, we also see that the average hamming distance between sequences in populations is much smaller when the population evolves with natural selection. For example, in Fig. 4, we show that the hamming distance between two specific branches in our tree structure nears 96 (almost completely different) when there is no natural selection while the hamming distance asymptotes to approximately 57 when there is natural selection. While this is only data from one experiment, all our experiments show similar trends.

3.2 Natural Selection and Its Effect on Phylogeny Reconstruction

The question now is, will natural selection have any impact, harmful or beneficial, on the effectiveness of phylogeny reconstruction algorithms. Our hypothesis is that natural selection will improve the performance of phylogeny reconstruction algorithms. Specifically, for the symmetric tree structures that we study, we predict that phylogeny reconstruction algorithms will do better when at least one of the two intermediate ancestors will have incorporated some mutations that significantly improve its fitness. The resulting structures in the genome are likely to be preserved in some fashion in the two descendant organisms making their pairing more likely. Since the likelihood of this occurring increases as the internal edge length in our symmetric tree structure increases, we expect to see the performance difference of algorithms increase as the internal edge length increases.

The results from our experiments support our hypothesis. In Fig. 5, we show that *MP* does no better on the Avida data than the random data when the internal edge length is 6. *MP* does somewhat better on the Avida data than the random data when the internal edge length grows to 50. Finally *MP* does significantly better on the Avida data than the random data when the internal edge length grows to 200.

3.3 Natural Selection via Location Probability Distributions

Is it possible to simulate the effects of natural selection we have observed by the random data generator? In part 1, we observed that natural selection does have some effect on the genome sequences. For example, mutations are frequently observed only on part of the genome. If we tune the random data generator to use non-uniform location probability distributions, is it possible to simulate the effects of natural selection?

To answer this question, we collected data from 20 Avida experiments to determine what the location probability distribution looks like with natural selection. We first looked at how many positions typically are fixed (no mutations). Averaging the data from the 20 Avida experiments, we saw that 21 % are fixed in a typical run. We then looked further to see how many positions were stable (mutation rate $\leq 1\%$) and how many positions were volatile (mutation rate $> 1\%$) in a typical experiment. Our results show that 35% of the positions are stable, and 44% of the positions are volatile.

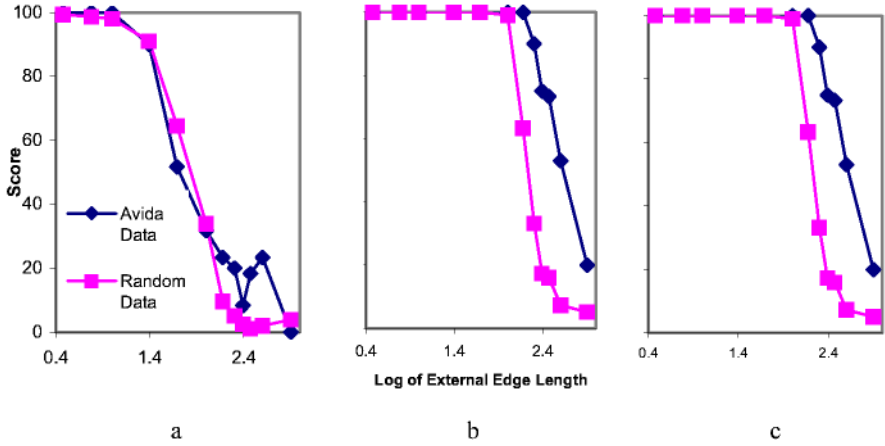


Fig. 5. *MP* scores vs log of external edge length. The internal edge lengths of a, b and c are 6, 50 and 200.

From these findings, we set up our random data generator with three different location probability distributions. The first is the uniform distribution. The second is a two-tiered distribution where 20 of the positions are fixed (no mutations) and the remaining 80 positions are equally likely. Finally, the third is a three-tiered distribution where 21 of the positions were fixed, 35 were stable (mutation rates of 0.296%), and 44 were volatile (mutation rates of 2.04%).

Results from using these three different location probability distributions are shown in Fig. 6. Random dataset A uses the three-tier location probability distribution. Random dataset B uses the uniform location probability distribution. Random dataset C uses the two-tier location probability distribution. We can see that *MP* exhibits similar performance on the Avida data and the random data with the three-tier location probability distribution.

Why does the three-tier location probability distribution seem to work so well? We believe it is because of the introduction of the stable positions (low mutation rates). Stable positions with a low probability are more likely to remain identical in the two final descendants that will make their final pairing more likely.

4 Future Work

While we feel that this preliminary work shows the effectiveness of using Avida to evaluate the effect of natural selection on phylogeny reconstruction, there are several important extensions that we plan to pursue in future work.

1. Our symmetric tree structure has only four taxa. Thus, there is only one internal edge and one bipartition. While this simplified the problem of determining if a reconstruction was correct or not, the scenario is not challenging and the full power

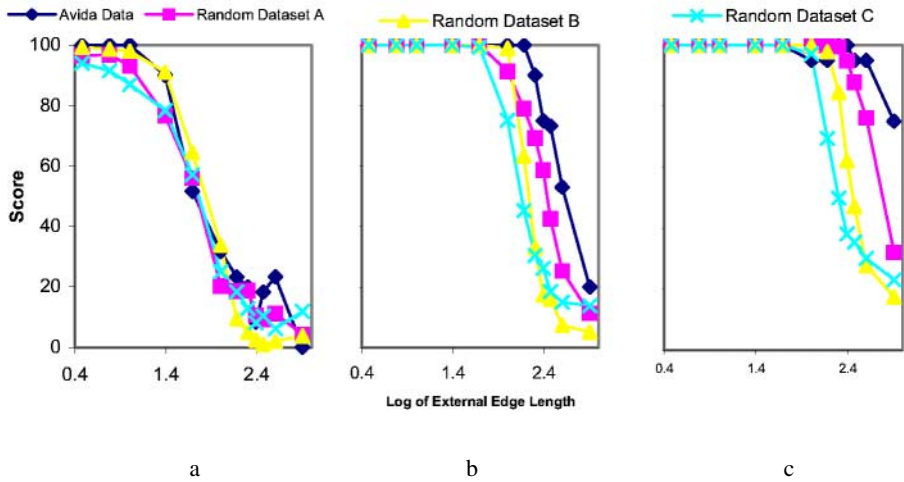


Fig. 6. MP scores from Avida data and 3 random datasets. The internal edge lengths of a, b and c are 6, 50 and 200.

of algorithms such as maximum parsimony could not be applied. In future work, we plan to examine larger data sets. To do so, we must determine a good method for evaluating partially correct reconstructions.

2. We artificially introduced branching events. We plan to avoid this in the future. To do so, we must determine a method for generating large data sets with similar characteristics in order to derive statistically significant results.

3. We used a fixed-length genome, which eliminates the need to align sequences before applying a phylogeny reconstruction algorithm. In our future work, we plan to perform experiments without fixed length, and we will then need to evaluate sequence alignment algorithms as well.

4. Finally, our environments were simple single niche environments. We plan to use more complex environments that can support multiple species that evolve independently.

Acknowledgements. The authors would like to thank James Vanderhyde for implementing some of the tools used in this work, and Dr. Richard Lenski for useful discussions. This work has been supported by National Science Foundation grant numbers EIA-0219229 and DEB-9981397 and the Center for Biological Modeling at Michigan State University.

References

1. Hillis D.M.: Approaches for Assessing Phylogenetic Accuracy, *Syst. Biol.* 44(1) (1995) 3–16
2. Huelsenbeck J.P.: Performance of Phylogenetic Methods in Simulation, *Syst. Biol.* 44(1) (1995) 17–48

3. Hillis D., Bull J.J., White M.E., Badgett M.R., Molineux L.J.: Experimental Phylogenetics: Generation of a Known Phylogeny. *Science* 255 (1992) 589–592
4. Ramnaut A. and Grassly N. C.: Seq-Gen: An application for the Monte Carlo simulation of DNA sequence evolution along phylogenetic trees. *Comput. Appl. Biosci.* 13 (1997) 235–238
5. Ofria C., Brown C.T., and Adami C.: The Avida User's Manual, 297–350 (1998)
6. Wilke C.O., Adami C.: The biology of digital organisms. *TRENDS in Ecology and Evolution*, 17:11 (2002) 528–532
7. Adami C., Ofria C., and Collier T.C.: Evolution of Biological Complexity. *Proc. Natl. Acad. Sci. USA* 97 (2000) 4463–4468
8. Wilke C.O., et. al.: Evolution of Digital Organisms at High Mutation Rates Leads to Survival of the Flattest. *Nature*, 412 (2001) 331–333
9. Lenski R.E., et. al.: Genome Complexity, Robustness, and Genetic Interactions in Digital Organisms. *Nature* 400 (1999) 661–664
10. Elena S.F. and Lenski, R.E.: Test of Synergistic Interactions Among Deleterious Mutations in Bacteria. *Nature* 390 (1997) 395–398
11. Gaut B.S. and Lewis P.O.: Success of Maximum Likelihood Phylogeny Inference in the Four-Taxon Case, *Mol. Biol. Evol* 12(1) (1995) 152–162
12. Tateno Y., Takezaki N., and Nei M.: Relative Efficiencies of the Maximum-Likelihood, Neighbor-joining, and Maximum Parsimony Methods When Substitution Rate Varies with Site, *Mol. Biol. Evol.* 11(2) (1994) 261–277
13. Saitou N. and Nei M.: The Neighbor-Joining Method: A New Method for Reconstructing Phylogenetic Trees, *Mol. Biol. Evol.* 4 (1987) 406–425
14. Studier J. and Keppler K.: A Note on the Neighbor-Joining Algorithm of Saitou and Nei, *Mol. Biol. Evol.* 5 (1988) 729–731
15. Fitch W.: Toward Defining the Course of Evolution: Minimum Change for a Specified Tree Topology, *Systematic Zoology*, 20 (1971) 406–416
16. Lenski E., Ofria C., Collier C. and Adami C.: Genome Complexity, Robustness and Genetic Interactions in Digital Organisms, *Nature*, 400 (1999) 661–664

AntClust: Ant Clustering and Web Usage Mining

Nicolas Labroche, Nicolas Monmarché, and Gilles Venturini

Laboratoire d'Informatique de l'Université de Tours,
École Polytechnique de l'Université de Tours-Département Informatique,
64, avenue Jean Portalis 37200 Tours, France
{labroche,monmarche,venturini}@univ-tours.fr
<http://www.antsearch.univ-tours.fr/>

Abstract. In this paper, we propose a new ant-based clustering algorithm called AntClust. It is inspired from the chemical recognition system of ants. In this system, the continuous interactions between the nestmates generate a “Gestalt” colonial odor. Similarly, our clustering algorithm associates an object of the data set to the odor of an ant and then simulates meetings between ants. At the end, artificial ants that share a similar odor are grouped in the same nest, which provides the expected partition. We compare AntClust to the K-Means method and to the AntClass algorithm. We present new results on artificial and real data sets. We show that AntClust performs well and can extract meaningful knowledge from real Web sessions.

1 Introduction

Numbers of computer scientists have proposed novel and successful approaches for solving problems by reproducing biological behaviors. For instance, genetic algorithms have been used in many research fields, such as clustering problems [1],[2] and optimization [3]. Other examples can be found in the modeling of collective behaviors of ants as in the well-known algorithmic approach Ant Colony Optimization (ACO)[4] in which pheromone trails are used. Similarly, ant-based clustering algorithms have been proposed ([5], [6], [7]). In these studies, researchers have modeled real ants abilities to sort their brood. Artificial ants may carry one or more objects and may drop them according to given probabilities. These agents do not communicate directly with each other's, but they may influence themselves through the configuration of objects on the floor. Thus, after a while, these artificial ants are able to construct groups of similar objects, a problem which is known as data clustering. We focus in this paper on another important collective behavior of the real ants, namely the construction of a colonial odor and its use to determine the ant nest membership.

Introduced in [8], the AntClust algorithm reproduces the main principles of this recognition system. It is able to find automatically a good partition over artificial and real data sets. Furthermore, it does not need the number of expected clusters to converge. It can also be easily adapted to any type of data

(from numerical vectors to character strings and multimedia), since a distance measure can be defined between the vectors of attributes that describe each object of the data set.

In this paper, we propose a new version of AntClust that does not need to be parameterized to produce the final partition. The paper is organized as follows: the section 2 gives a detailed description of the AntClust algorithm. The section 3 presents the experiments that have been conducted to set the parameters of AntClust regardless of the data sets. The section 4 compares the results of AntClust to those of the K-Means method (initialized with the expected number of clusters) and those of AntClass, an ant-based clustering algorithm. In the section 5, we present some of the clustering algorithms already used in the Web mining context and our very first results when we apply AntClust to real Web sessions. The last section concludes and discusses future evolutions of AntClust.

2 The AntClust Algorithm

The goal of AntClust is to solve the unsupervised clustering problem. It finds a partition, as close as possible to the natural partition of the data set, without any assumption concerning the definition of the objects or the number of expected clusters. The originality of AntClust is to model the chemical recognition system of ants to solve this problem. Real ants solve a similar problem in their every day life, when the individuals that wear the same cuticular odor gather in the same nest. AntClust associates an object of the data set to the genome of an artificial ant. Then, it simulates meetings between artificial ants to exchange their odor. We present hereafter the main principles of the chemical recognition system of ants. Then, we describe the representation and the coding of the parameters of an artificial ant and also the behavioral rules that allow the method to converge.

2.1 Principles of the Chemical Recognition System of Ants

AntClust is inspired from the chemical recognition system of ants. In this biological system, each ant possesses its own odor called label that is spread over its cuticle (its “skin”). The label is partially determined by the genome of the ant and by the substances extracted from its environment (mainly the nest materials and the food). When they meet other individuals, ants compare the perceived label to their template that they learned during their youth. This template is then updated during all their life by the mean of trophallaxies, allo-grooming and social contacts. The continuous chemical exchanges between the nestmates lead to the establishment of a colonial odor that is shared and recognized by every nestmates, according to the “Gestalt theory” [9,10].

2.2 The Artificial Ants Model

An artificial ant can be considered as a set of parameters that evolve according to behavioral rules. These rules reproduce the main principles of the recognition system and apply when two ants meet.

For one ant i , we define the parameters and properties listed hereafter.

The label $Label_i$ indicates the belonging nest of the ant and is simply coded by a number. At the beginning of the algorithm, the ant does not belong to a nest, so $Label_i = 0$. The label evolves until the ant finds the nest that best corresponds to its genome.

The genome $Genome_i$ corresponds to an object of the data set. It is not modified during the algorithm. When they meet, ants compare their genome to evaluate their similarity.

The template $Template_i$ or T_i is an acceptance threshold that is coded by a real value between 0 and 1. It is learned during an initialization period, similar to the ontogenesis period of the real ants, in which each artificial ant i meets other ants, and each time evaluates the similarity between their genomes. The resulting acceptance threshold T_i is a function of the maximal $Max(Sim(i, \cdot))$ and mean $\overline{Sim}(i, \cdot)$ similarities observed during this period. T_i is dynamic and is updated after each meeting realized by the ant i , as the similarities observed may have changed. The following equation shows how this threshold is learned and then updated:

$$T_i \leftarrow \frac{\overline{Sim}(i, \cdot) + Max(Sim(i, \cdot))}{2} \quad (1)$$

Once artificial ants have learned their template, they use it during their meetings to decide if they should accept the encountered ants. We define the acceptance mechanism between two ants i and j as a symmetric relation $A(i, j)$ in which the genomes similarity is compared to both templates as follows:

$$A(i, j) \Leftrightarrow (Sim(i, j) > T_i) \wedge (Sim(i, j) > T_j) \quad (2)$$

We state that there is “positive meeting” when there is acceptance between ants.

The estimator M_i indicates the proportion of meetings with nestmates. This estimator is set to 0 at the beginning of the algorithm. It is increased each time the ant i meets another ant with the same label (a nestmate) and decreased in the opposite case. M_i enables each ant to estimate the size of its nest.

The estimator M_i^+ reflects the proportion of positive meetings with nestmates of the ant i . In fact, this estimator measures how well accepted is the ant i in its own nest. It is roughly similar to M_i but add the “acceptance notion”. It is increased when ant i meets and accepts a nestmate and decreased when there is no acceptance with the encountered nestmate.

The age A_i is set to 0 and is increased each time the ant i meets another ant. It is used to update the maximal and mean similarities values and thus the value of the acceptance threshold of the ant $Template_i$.

At each iteration, AntClust randomly selects two ants, simulates meetings between them and applies a set of behavioral rules that enable the proper convergence of the method.

The 1st rule applies when two ants with no nest meet and accept each other. In this case, a new nest is created. This rule initiates the gathering of similar ants in the very first clusters. These clusters “seeds” are then used to generate the final clusters according to the other rules.

The 2nd rule applies when an ant with no nest meets and accepts an ant that already belongs to a nest. In this case, the ant that is alone joins the other in its nest. This rule enlarges the existing clusters by adding similar ants.

The 3rd rule increments the estimators M and M^+ in case of acceptance between two ants that belong to the same nest. Each ant, as it meets a nestmate and tolerates it, imagines that its nest is bigger and, as there is acceptance, feels more integrated in its nest.

The 4th rule applies when two nestmates meet and do not accept each other. In this case, the worst integrated ant is ejected from the nest. That rule permits to remove non-optimally clustered ants to change their nest and try to find a more appropriate one.

The 5th rule applies when two ants that belong to a distinct nest meet and accept each other. This rule is very important because it allows the gathering of similar clusters, the small one being progressively absorbed by the big one.

The AntClust algorithm can be summarized as follows:

Algorithm 1: ANTCLUST main algorithm

ANTCLUST()

- (1) Initialization of the ants:
 - (2) $\forall$ ants $i \in [1, N]$
 - (3) $Genome_i \leftarrow i^{th}$ object of the data set
 - (4) $Label_i \leftarrow 0$
 - (5) $Template_i$ is learned during N_{App} iterations
 - (6) $M_i \leftarrow 0, M_i^+ \leftarrow 0, A_i \leftarrow 0$
 - (7) $Nb_{Iter} \leftarrow 75 * N$
 - (8) Simulate Nb_{Iter} meetings between two randomly chosen ants
 - (9) Delete the nests that are not interesting with a probability P_{del}
 - (10) Re-assign each ant that has no more nest to the nest of the most similar ant.
-

3 AntClust Parameters Settings

It has been shown in [8] that the quality of the convergence of AntClust mainly depends on three major parameters, namely the number of iterations fixed to learn the template N_{App} , the number of iterations of the meeting step Nb_{Iter} and finally, the method that is used to filter the nests. We describe hereafter how we can fix the value of these parameters regardless of the structure of the data sets. First, we present our measure of the performance of the algorithm and the data sets used for evaluation.

3.1 Performance Measure

To express the performance of the method we define C_s as $1 - C_e$, where C_e is the clustering error. We choose an error measure adapted from the measure developed by Fowlkes and Mallows as used in [11]. The measure evaluates the differences between two partitions by comparing each pair of objects and by verifying each time if they are clustered similarly or not. Let P_i be the expected partition and P_a the output partition of AntClust. The clustering success $C_s(P_i, P_a)$ can be defined as follows:

$$C_s(P_i, P_a) = 1 - \frac{2}{N(N-1)} \times \sum_{(m,n) \in [1..N]^2, m < n} \epsilon_{mn} \quad (3)$$

where:

$$\epsilon_{mn} = \begin{cases} 0 & \text{if } (P_i(m) = P_i(n) \wedge P_a(m) = P_a(n)) \vee \\ & (P_i(m) \neq P_i(n) \wedge P_a(m) \neq P_a(n)) \\ 1 & \text{otherwise} \end{cases} \quad (4)$$

with N the number of objects in the original data set. $P_i(o_b)$ (resp. $P_a(o_b)$) is the cluster number of the object o_b in the partition P_i (resp. P_a).

We use artificial data sets named $Art_{1,2,3,4,5,6}$ for our evaluations. We generate them according to gaussian or uniform laws with distinct difficulties (irrelevant attributes, clusters overlap).

3.2 How Many Meetings?

For the time being, the user has to specify the number of iterations during which artificial ants meet, if she wants to adapt the algorithm to the size of the data set that is explored. AntClust has a default value that is set to 50000 iterations. Nevertheless, in the Web mining context, it is usual to work with larger data sets. In this case, AntClust cannot guarantee a proper convergence since some ants are assigned to a nest without having realized any meeting. Our idea is to consider that the number of iterations may be linearly linked to the number of ants (i.e. the number of objects in the data set). In the method, at each iteration, 2 ants are randomly and uniformly selected in the population. Let α denote the minimal number of iterations each ant has to perform to ensure the convergence of the algorithm. The following equation shows how we associate the total number of iterations Nb_{Iter} to the number of ants N :

$$Nb_{Iter} = \frac{1}{2} * \alpha * N \quad (5)$$

We evaluate the performance of the method for several values of α between 10 and 500 meetings per ant. The goal is to verify that α can be initialized regardless of the size of the data set. The figure 1 shows the results in term of mean clustering success. We have conducted 10 tests for each value of α and each artificial data set.

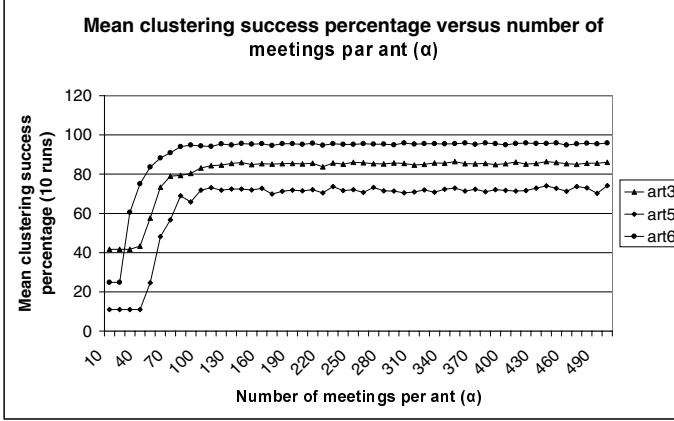


Fig. 1. Mean clustering success over 10 runs for each value of $\alpha \in [10, 500]$.

According to this figure, we can see that the values of the mean clustering success converge quite quickly regardless of the data set. Although the data sets size ranges from 200 to 1100 objects, the convergence seems to operate for the same value of α in every case. Thus, we may consider that there is a minimal number of meetings that each ant should realize to ensure the proper convergence of the method. We consider that we can set experimentally the value of α to 150.

As a consequence, the meetings step of AntClust can be solved in linear time with the number of objects in the data set.

3.3 How Many Iterations to Learn the Template?

We now focus on the number of iterations needed to learn the template. We think that the template learning process cannot be longer, in term of number of iterations, than the meetings step studied before. Let β be the number of meetings needed to learn the template per ant. As for the number of iterations of the meeting step, we test several values of β expressed as a percentage of the value of α ranging from 0% to 100%. The figure 2 presents the results that we obtained for each artificial data sets in term of mean clustering success according to the value of β .

As with the previous experiment, the results of mean clustering success converges even if the limit is not as clearly expressed in the plotted graph. It is important to notice that β is not necessarily positively linked to the performance. For example, the mean clustering success of *Art*₆ remains stable as β increases, whereas the performances of *Art*₄ decrease. This can be explained by the fact that the number of clusters is taken into account to compute C_s and that as β increases the error in the estimated number of clusters also increases

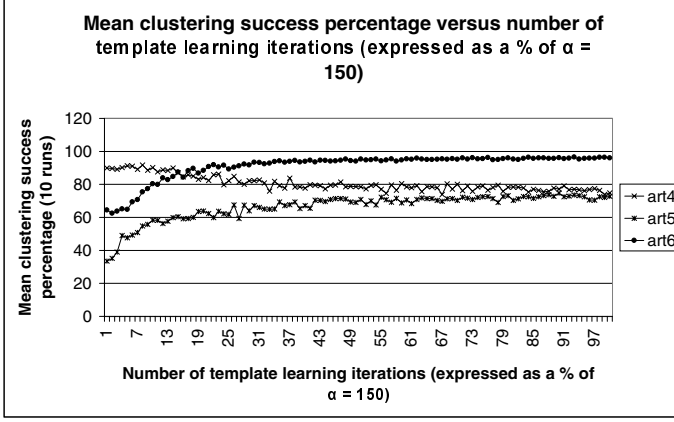


Fig. 2. Mean clustering success over 10 runs for each $\beta \in [1, 100]\%$ of α .

for *Art*₄. In fact, by increasing β , artificial ants become too sensitive and find too much clusters. As a conclusion, we set β as $0.5 * \alpha$, that is to say, $N_{App} \leftarrow 75$.

3.4 The Nest Deletion Method

At the end of AntClust, the nests that are not sufficiently “interesting” are deleted and the ants are reassigned to the nest of the most similar ant. This method allows suppressing noise in the final partition. For the time being, the nest deletion criterion is only based on the number of ants that belong to this nest and a threshold fixed by the user. The default value of this threshold is equal to a percentage of the total number of ants (generally 15%). This approach, although efficient, is limited, because the algorithm could not find more than a fixed number of clusters of the same size. Our optimization replaces this deterministic method by a probabilistic one that is more adaptive. We compute for each nest η a probability P_{del} to be deleted. This probability depends mainly on the mean integration of the ants in the nest η , $\overline{M}_{\eta}^{+}$, and the number of ants in the nest N_{η} as in the previous version. For the nest η , $P_{del}(\eta)$ is given as follows:

$$P_{del}(\eta) \leftarrow (1 - \nu) * \overline{M}_{\eta}^{+} + \nu * \frac{N_{\eta}}{N} \quad (6)$$

Several experimentations have been conducted and revealed that the value of $\nu = 0.2$ provides the best partitions for the set of data we tested. Hence, this method is more interesting than the previous one, because it allows to better appreciate the number of clusters thanks to its probabilistic nature.

4 Experiments and Results

In this section, we compare AntClust to the K-Means method and the AntClass algorithm to evaluate the performance of our method. First, we will briefly describe both methods. Second, we introduce the main properties of the data sets that are used for the comparison. Then, we present our results over artificial and real data sets.

4.1 K-Means and AntClass

We use two clustering algorithms to evaluate AntClust. First, we choose to apply a traditional K-Means approach because, it can perform well in a short time. This method needs an initial K-partition of the data set that is refined gradually. At each iteration, the objects are assigned to the most similar center of the clusters. The algorithm stops when the intra class inertia becomes stable. In our test, we generate randomly the initial partitions with the number of clusters that is expected in the data set, to get the best results that a K-Means approach can give.

Second, we compare AntClust to AntClass, an other ant-based clustering algorithm. This is an hybrid algorithm in which artificial ants create a first partition of the data set that is used by a K-Means algorithm as initial partition. This scheme is repeated twice in order to get the final partition. The ant-based part of AntClass relies on the establishment of a 2 dimensional grid. The objects and the artificial ants are randomly placed on the grid. At each iteration, ants move and have a probability to pick-up or to drop an object depending if they have already one or not. Artificial ants generate heaps of similar objects that define the partition.

4.2 Data Sets and Experimental Protocol

We evaluate the clustering methods over several data sets that represent distinct clustering difficulties in the same experimental conditions in order to better appreciate the performance of each of them. First, we test K-Means, AntClass and AntClust over the artificial data sets that have been previously introduced in the section 3.1, namely $Art_{1,2,3,4,5,6}$.

Second, we test the algorithms over real data sets such as Iris, Pima, Soybean, Glass and Thyroid. We expect these data sets to be more difficult to be cluster, since they may be noisier than artificial data sets. We introduce in the table 1 the parameters that characterize the artificial and real data sets used for our evaluations. The fields for each data set are: the number of objects (N), their associated number of attributes (M), and the number of clusters (K).

For each data set, we run 50 times each method and compute the mean clustering success (see section 3.1), the mean number of clusters found and their respective standard deviations. The next section presents our results.

Table 1. Main characteristics of the data sets.

	<i>Art₁</i>	<i>Art₂</i>	<i>Art₃</i>	<i>Art₄</i>	<i>Art₅</i>	<i>Art₆</i>
N	400	1000	1100	200	900	400
M	2	2	2	2	2	8
K	4	2	4	2	9	4
	<i>Iris</i>	<i>Glass</i>	<i>Pima</i>	<i>Soybean</i>	<i>Thyroid</i>	
N	150	214	798	47	215	
M	4	9	8	35	5	
K	3	7	2	4	3	

4.3 Results with Artificial and Real Data Sets

The table 2 shows the results obtained over the artificial and real data sets. We can see in this table that K-Means gives, in general, the best results in term of clustering success and obviously in term of number of clusters found as they are provided to the algorithm. AntClust has the best clustering results only twice for the data sets *Soybean* and *Thyroid* that are little data sets (with respectively 47 and 215 objects). Nevertheless, AntClust shows its ability to treat large data sets with *Art₂* and *Art₃*. AntClass performs better than AntClust three times, for *Art₁*, *Art₅* and *Iris*. For *Art₁* and *Art₅*, AntClass better estimates the number of expected clusters, but for *Iris*, the situation is reversed. This means that in this case, AntClust creates poorer quality clusters than those of AntClass. In all the other experiments K-Means and AntClust are more efficient than AntClass. One thing to point out is that when K-Means fails (for example in the data sets *Glass* and *Pima*), AntClass (which is partially based on K-Means method) and AntClust also behaves poorly. Nevertheless, AntClust performs well in general and can be even more efficient than K-Means for which the number of clusters is provided.

To complete the evaluation of AntClust, we compare its complexity to those of K-Means and AntClass. Let C_x be the complexity of the algorithm x . AntClust can be splitted in several steps: the template learning process ($\theta(N \times N_{App})$), the creation of the nests ($\theta(N \times Nb_{Iter})$) and finally the nests deletion process and the re-assignment of the ants to a nest. This step runs in quadratic time since each ant that has no more nest has to find the the most similar ant that belongs to a nest. The complexity of AntClust is $C_{AntClust} = \theta(N^2)$ in the worst case. The K-Means algorithm's complexity is known to be $C_{KMeans} = \theta(N)$ and AntClass has also a linear complexity $C_{AntClass} = \theta(N)$. Experimentally, our tests revealed that K-Means is the quickest method and that AntClust runs faster than AntClass.

Table 2. Mean number of clusters (# clusters) and mean success (Success) and their standard deviation for each data set and each method computed over 50 runs.

Data sets	# clusters						Success					
	K-Means		AntClass		AntClust		K-Means		AntClass		AntClust	
	mean	[std]	mean	[std]	mean	[std]	mean	[std]	mean	[std]	mean	[std]
<i>Art₁</i>	3.98	[0.14]	4.22	[1.15]	4.70	[0.95]	0.89	[0.00]	0.85	[0.05]	0.78	[0.03]
<i>Art₂</i>	2.00	[0.00]	12.32	[2.01]	2.30	[0.51]	0.96	[0.00]	0.59	[0.01]	0.93	[0.02]
<i>Art₃</i>	3.84	[0.37]	14.66	[2.68]	2.72	[0.88]	0.78	[0.02]	0.65	[0.01]	0.85	[0.02]
<i>Art₄</i>	2.00	[0.00]	1.68	[0.84]	4.18	[0.83]	1.00	[0.00]	0.71	[0.23]	0.77	[0.05]
<i>Art₅</i>	8.10	[0.75]	11.36	[1.94]	6.74	[1.66]	0.91	[0.02]	0.92	[0.01]	0.74	[0.02]
<i>Art₆</i>	4.00	[0.00]	3.74	[1.38]	4.06	[0.24]	0.99	[0.04]	0.89	[0.13]	0.95	[0.01]
<i>Iris</i>	2.96	[0.20]	3.52	[1.39]	2.82	[0.75]	0.86	[0.03]	0.81	[0.08]	0.78	[0.01]
<i>Glass</i>	6.88	[0.32]	5.60	[2.01]	5.90	[1.23]	0.68	[0.01]	0.60	[0.06]	0.64	[0.02]
<i>Pima</i>	2.00	[0.00]	6.10	[1.84]	10.66	[2.33]	0.56	[0.00]	0.53	[0.02]	0.54	[0.01]
<i>Soybean</i>	3.96	[0.20]	1.60	[0.49]	4.16	[0.55]	0.91	[0.08]	0.46	[0.17]	0.93	[0.04]
<i>Thyroid</i>	3.00	[0.00]	5.84	[1.33]	4.62	[0.90]	0.82	[0.00]	0.78	[0.09]	0.84	[0.03]

5 AntClust for Web Usage Mining

For the time being, a lot of research efforts have been conducted to cluster user sessions extracted from Web servers log files. The recurrent problem in this area is that clustering algorithms must be able to treat large data sets with an affordable computing time. Actually, a single Web server log file may contain several hundred thousand requests for Web pages. As a consequence, researchers try to use algorithms that run fast in their first approaches.

In [12], Yan et al. use a “First leader clustering algorithm”. The sessions are expressed as numerical vectors containing for each Web page, the number of recorded impacts. Although results are very promising and quickly computed, this method is limited since the final partition depends on the order of the sessions in the data set.

Heer and Chi introduce in [13] the Wavefront algorithm that improves the initialization step of the K-Means method. The cluster seeds are randomly generated according to an estimated center of gravity of the data set. According to the authors, this method allows a quicker convergence. In their work, the Web sessions are expressed as multi-modal vectors that take into account the navigation of the users (the time spent for each page) and model a page as a combination of structure and content information.

Finally, Estivill-Castro et al. propose in [14] a robust clustering algorithm that is mainly a K-Means algorithm in which the median estimator is used instead of the mean estimator. The algorithm is then more resistant to the noise in the data sets.

There are two major limitations to use “K-Means like” algorithms in the Web usage mining context. First, the number of clusters K has to be provided to ensure a good convergence of the method, but it can’t be easily set, unless

the user is able to guess how people navigate on her Web site, which is exactly the goal of the clustering process. Second, mean values have to be computed and limit the coding of the Web sessions to numerical expressions (difficulties for keywords or multimedia contents coding). Finally, mean values may not have any meaning in the Web sessions context.

5.1 Web Session Data

The last data set that we explore with AntClust is a Web server log file. We sort and filter this raw file to get a data set composed of users sessions. A session captures the activity of a user on a Web site during a specified period of time. The sessions have been recorded for one month in October 2001, on a web site of the University of Tours that contains computer sciences courses. The 1064 reconstructed sessions contain 667 unique sessions, that is to say, sessions that come from an IP number that has been seen only once in the requests log file. Consequently, our sessions may reflect a lot of distinct behaviors from the users, and thus may be very noisy. Nevertheless, as there are few hyperlinks between the online courses, we expect the clusters to be representative of a minimum of courses in order to be valuable and understandable. Similarly to other works, we coded the Web sessions as vectors where each component corresponds to the number of hits recorded for each page.

5.2 Results

When applied on Web sessions, AntClust finds 17 clusters. The 3 largest clusters contain the half of the sessions. Although the two largest clusters refer to 2 or 3 online computer sciences courses, the others generally reflect interests for only one course. This probably means that a majority of users went to the Web site with no specific goal and look at several courses to evaluate the content of the diploma. The other users were certainly already students and were searching for lecture notes relative to one topic. This little experiment proves that AntClust is able to generate a non-noisy partition of Web sessions that can help understanding the interests of the Web users. AntClust spends approximately 1.33 minutes at 650 MHz to cluster the user sessions, which is an affordable time.

6 Conclusion and Perspectives

AntClust is a new clustering algorithm that models the chemical recognition system of real ants. It associates one object of the data set to one ant and defines the expected partition as a set of nests. In this paper, we show how the internal parameters of the method can be set, regardless of the structure of the data set to be explored. Furthermore, we develop a non-deterministic approach to delete the uninteresting nest and to re-affect the ants which nest has been deleted. We evaluate its performance against K-Means and AntClass with artificial and real data sets. We show that AntClust can even do better than K-Means for two data

sets. When applied to Web sessions, AntClust finds meaningful clusters that can help understanding the interests of the users of the Web site. Our future works will try to apply AntClust to larger Web data sets to evaluate its robustness. In these experiments we will compare AntClust to other clustering algorithms in the Web context. We also plan to develop an incremental version of AntClust and our first results seem to be promising.

References

1. Y. Chiou and L. W. Lan, "Genetic clustering algorithms," *European journal of Operational Research*, no. 135, pp. 413–427, 2001.
2. L. Y. Tseng and S. B. Yang, "Genetic clustering algorithms," *European journal of Operational Research*, no. 135, pp. 413–427, 2001.
3. N. Monmarché, G. Venturini, and M. Slimane, "On how *Pachycondyla apicalis* ants suggest a new search algorithm," *Future Generation Computer Systems*, vol. 16, no. 8, pp. 937–946, 2000.
4. A. Coloni, M. Dorigo, and V. Maniezzo, "Distributed optimization by ant colonies," in *Proceedings of the First European Conference on Artificial Life* (F. Varela and P. Bourguin, eds.), pp. 134–142, MIT Press, Cambridge, Massachusetts, 1991.
5. E. Lumer and B. Faieta, "Diversity and adaptation in populations of clustering ants," in Cliff *et al.* [15], pp. 501–508.
6. P. Kuntz and D. Snyers, "Emergent colonization and graph partitioning," in Cliff *et al.* [15], pp. 494–500.
7. N. Monmarché, M. Slimane, and G. Venturini, "On improving clustering in numerical databases with artificial ants," in *Lecture Notes in Artificial Intelligence* (D. Floreano, J. Nicoud, and F. Mondala, eds.), (Swiss Federal Institute of Technology, Lausanne, Switzerland), pp. 626–635, Springer-Verlag, 13–17 September 1999.
8. N. Labroche, N. Monmarché, and G. Venturini, "A new clustering algorithm based on the chemical recognition system of ants," in *Proc. of 15th European Conference on Artificial Intelligence (ECAI 2002)*, Lyon FRANCE, pp. 345–349, 2002.
9. B. Hölldobler and E. Wilson, *The Ants*. Springer Verlag, Berlin, Germany, 1990.
10. N. Carlin and B. Hölldobler, "The kin recognition system of carpenter ants(*camponotus* spp.). i. hierarchical cues in small colonies," *Behav Ecol Sociobiol*, vol. 19, pp. 123–134, 1986.
11. J. Heer and E. Chi, "Mining the structure of user activity using cluster stability," in *Proceedings of the Workshop on Web Analytics, SIAM Conference on Data Mining (Arlington VA, April 2002)*., 2002.
12. T. Yan, M. Jacobsen, H. Garcia-Molina, and U. Dayal, "From user access patterns to dynamic hypertext linking," in *Proc. of 5th WWW*, pp. 1007–1014, 1996.
13. J. Heer and E. Chi, "Identification of web user traffic composition using multi-modal clustering and information scent," 2001.
14. V. Estivill-Castro and J. Yang, "Categorizing visitors dynamically by fast and robust clustering of access logs," *Lecture Notes in Computer Science*, vol. 2198, pp. 498–509, 2001.
15. D. Cliff, P. Husbands, J. Meyer, and S. W., eds., *Third International Conference on Simulation of Adaptive Behavior: From Animals to Animats 3*, MIT Press, Cambridge, Massachusetts, 1994.

A Non-dominated Sorting Particle Swarm Optimizer for Multiobjective Optimization

Xiaodong Li

School of Computer Science and Information Technology
RMIT University, VIC 3001, Melbourne, Australia
xiaodong@cs.rmit.edu.au
<http://www.cs.rmit.edu.au/~xiaodong>

Abstract. This paper introduces a modified PSO, Non-dominated Sorting Particle Swarm Optimizer (NSPSO), for better multiobjective optimization. NSPSO extends the basic form of PSO by making a better use of particles' personal bests and offspring for more effective non-domination comparisons. Instead of a single comparison between a particle's personal best and its offspring, NSPSO compares all particles' personal bests and their offspring in the entire population. This proves to be effective in providing an appropriate selection pressure to propel the swarm population towards the Pareto-optimal front. By using the non-dominated sorting concept and two parameter-free niching methods, NSPSO and its variants have shown remarkable performance against a set of well-known difficult test functions (ZDT series). Our results and comparison with NSGA II show that NSPSO is highly competitive with existing evolutionary and PSO multiobjective algorithms.

1 Introduction

Multiobjective optimization problems represent an important class of real-world problems. Typically such problems involve trade-offs. For example, a car manufacturer may wish to maximize its profit, but meanwhile also to minimize its production cost. These objectives are typically conflicting to each other. A higher profit would increase the production cost. There is no single optimal solution. Often the manufacturer needs to consider many possible “trade-off” solutions before choosing the one that suits its need. The curve or surface (for more than 2 objectives) describing the optimal trade-off solutions between objectives is known as the Pareto front. One of the major goals in multiobjective optimization is to find a set of well distributed optimal solutions along the Pareto front.

In recent years, population-based optimization methods such as Evolutionary Algorithms have become increasingly popular for solving multiobjective optimization problems [1][2]. EA's success is due to its generic ability to handle large complex real-world problems. Since EAs maintain a population of solutions, this allows exploration of different parts of the Pareto front simultaneously. However not until recently, another population-based optimization

technique Particle Swarm Optimization (PSO) has been only applied to single objective optimization tasks. PSO technique is inspired by studies of social behavior of insects and animals [3]. The social behavior is modeled in a PSO to guide a population of particles (so-called swarm) moving towards the most promising area of the search space. In PSO, Each particle represents a candidate solution, $X_i = (x_{i1}, x_{i2}, \dots, x_{id})$. d is the dimension of the search space. The i -th particle of the swarm population knows: a) its *personal best position* $P_i = (p_{i1}, p_{i2}, \dots, p_{id})$, i.e., the best position this particle has visited so far that yields the highest fitness value; and b) the *global best position*, $P_g = (p_{g1}, p_{g2}, \dots, p_{gd})$, i.e., the position of the best particle that gives the best fitness value in the entire population; and c) its current *velocity*, $V_i = (v_{i1}, v_{i2}, \dots, v_{id})$, which represents its position change; The following equation (1) uses the above information to calculate the new updated velocity for each particle in the next iteration step. Equation (2) updates the each particle's position in the search space.

$$v_{id} = wv_{id} + c_1r_1(p_{id} - x_{id}) + c_2r_2(p_{gd} - x_{id}) \quad (1)$$

$$x_{id} = x_{id} + \chi v_{id} , \quad (2)$$

where $d = 1, 2, \dots, D$; $i = 1, 2, \dots, N$; N is the size of the swarm population; χ is a constriction factor which controls and constricts the velocity's magnitude; w is the inertia weight, which is often used as a parameter to control exploration/exploitation in the search space; c_1 and c_2 are two coefficients (positive constants); r_1 and r_2 are two random numbers within the range $[0, 1]$. There is also a V_{MAX} , which sets the upper and lower bound for velocity values.

PSO has proved to be an efficient optimization method for single objective optimization, and more recently has also shown promising results for solving multiobjective optimization problems [4][5] [6][7]. What is in common among these works is the use of a basic form of PSO first introduced by Kennedy and Eberhart [3]. However the basic form of PSO has some serious limitations in particular when dealing with multiobjective optimization problems. In PSO, a particle is modified only through its personal best and global best to produce its offspring. At each iteration step, if the fitness of the offspring is better than the parent's personal best, then the personal best is updated with this offspring, however, there is no sharing of information with other particles in the population, except that each particle can access the global best. For multiobjective optimization, we argue that such sharing of information among all the individuals in a population is crucial in order to introduce the necessary selection pressure to propel the population moving towards the true Pareto-optimal front.

This paper introduces a modified PSO, Non-dominated Sorting Particle Swarm Optimizer (NSPSO), which is able to increase such "sharing" among all particles in a swarm population especially concerning how to allow the population as a whole to progress towards the true Pareto-optimal front. For clarity, in this paper, we use P^* to denote the true Pareto-optimal front, and Q the found non-dominated solution set.

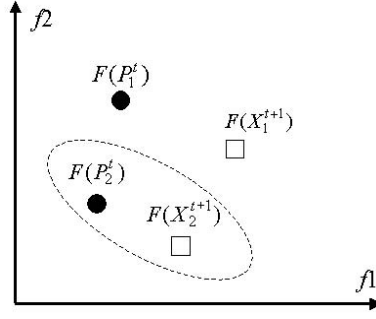


Fig. 1. Dominance relationships among 4 particles, including the personal best P_1^t of a particle X_1^t , and its potential offspring X_1^{t+1} , plus P_2^t and X_2^{t+1} for a second particle X_2^t , assuming minimization of f_1 and f_2 .

2 Modifying PSO for Better Dominance Comparison

One problem that can be identified with the basic form PSO is that dominance comparisons are not fully utilized in the process of updating the personal best P_i of each particle. This can be illustrated via the following example shown in Fig. 1. Note that $F(\cdot)$ denotes the evaluation of a particle in the objective space.

Fig.1 shows that the personal best P_1^t is mutually non-dominating with X_1^{t+1} , and P_2^t is non-dominating with X_2^{t+1} , however, X_1^{t+1} is dominated by X_2^{t+1} and P_2^t , and furthermore, P_1^t is also dominated by P_2^t . In a standard PSO, the i -th particle only has a single P_i^t , which is used to compare with its potential offspring X_i^{t+1} at time step $t+1$. If P_i^t is non-dominating with its potential offspring such as the situation shown in Fig. 1, then P_i^t will remain the same. The consequence of this kind of comparison is that the useful non-domination relationships among all the four particles will not be captured. Fig. 1 shows that if we allow all 4 particles to be compared, then we would have found P_2^t and X_2^{t+1} to be the better two to retain. This illustration shows that in a standard PSO, valuable non-domination comparisons are not effectively used. In other words, as a result of having only a single comparison between a particle's P_i^t and X_i^{t+1} , there is only a very weak selection pressure with respect to the non-dominated front that exists in the current population.

This would naturally lead to the next two questions - what if we allow all the personal bests of all particles and as well as these particles' offspring to be compared for non-domination relationships? How are we going to choose the global best for each particle in order to propel the population to move towards P^* , while also maintaining a diverse set of solutions?

3 Non-dominated Sorting PSO

Two major goals in multiobjective optimization are to obtain a set of non-dominated solutions as closely as possible to the true Pareto front P^* , and to

maintain a well-distributed solution set along the Pareto front. To achieve this using PSO, a Non-dominated Sorting Particle Swarm Optimizer (NSPSO) is proposed. In NSPSO, we adopt the non-dominated sorting concept used in NSGA II [2], where the entire population is sorted into various non-domination levels. This provides the means for selecting the individuals in the better fronts, hence providing the necessary selection pressure to push the population towards P^* . To maintain population diversity, we use a widely-used niching method [8], and also the crowding distance assignments adopted by NSGA II [9]. The following two sections describe these methods.

3.1 Selection Pressure towards P^*

Instead of comparing solely on a particle's personal best with its potential offspring, the entire population of N particles' personal bests and N of these particles' offspring are first combined to form a temporary population of $2N$ particles. After this, domination comparisons among all the $2N$ individuals in this temporary population are carried out. This "combine-then-compare" approach will ensure more non-dominated solutions can be discovered through the domination comparison operations. Since updating the personal bests of the N particles represents "collectively" a step towards a better region in the search space, retaining them for domination comparison provides the needed selection pressure towards P^* . By comparing the combined $2N$ particles for non-domination relationships, we will be able to sort the entire population in different non-domination levels as used in NSGA II. This type of sorting can then be used to introduce the selection bias to the individuals in the populations, in favour of individuals closer to the true Pareto front P^* . At each iteration step, we choose only N individuals out of the $2N$ to the next iteration step, based on the non-domination levels. This process can be illustrated in Fig. 2. First the entire population is sorted into

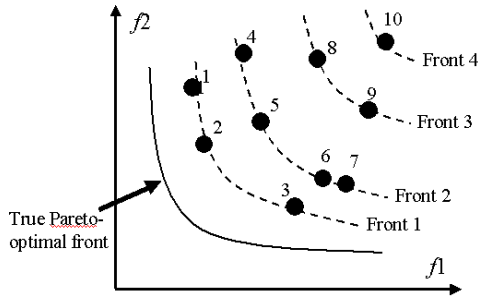


Fig. 2. Particles of a swarm population of 10 are classified into 4 successive non-dominated fronts

two sets, the non-dominated set and the remaining dominated set. In Fig.2, the non-dominated set is Front 1, which contains 3 particles labeled as 1, 2 and 3. Front 1 is the best non-dominated set, since all particles in Front 1 are not dominated by any other particles in the entire population. To obtain the next front,

Front 2, we temporarily remove Front 1 from the population, then find the non-dominated solutions of the remaining population, which is Front 2. Then again we remove Front 2 as well, in order to identify the non-dominated solutions of the next level. This procedure continues until all particles in the population are classified into different non-dominated front levels. For each particle, there are $O(mN)$ comparisons are required to find out if it is dominated by other particles in a population of size N (m is the number of objectives). The complexity to find the first non-dominated front (i.e., Front 1) for the whole population is $O(mN^2)$. In the worst case where there is only one particle in each front, the complexity of the above procedure for classifying different non-dominated fronts is $O(mN^3)$.

Now we create the new particle population for the next iteration step, by selecting particles from fronts in ascending order, e.g., first from Front 1, then Front 2, etc, until N particles (or a specified threshold) are selected. Since the particles in the first few fronts get chosen first, this selection pressure will effectively drive the particle population towards the best front over many iteration steps. Note that Front 1 could have more than N particles (since the combined $2N$ particles are sorted), especially after a number of steps in a run. Setting a threshold may be necessary as it would allow opportunities of particles from other fronts to be selected as well, i.e., maintaining “lateral diversity” [2].

3.2 Parameter-Free Niching Methods to Maintain a Diverse Q

Niching methods have been extensively studied and used as means of maintaining population diversity in Genetic Algorithms. One commonly used niching method is a sharing function model introduced by Goldberg and Richardson [10], where a niche is treated as a resource shared among other individuals in the niche. In NSPSO, to achieve the second goal of maintaining a diverse non-dominated solution set, two niching methods are tried, in order to see how effective each method is in maintaining solution diversity. The first method requires calculating a niche count for each particle, whereas the second method requires calculating a crowding distance value for each particle.

Using Niche Count. In NSPSO, the niche count m_i of a particle i is simply calculated as the number of other particles within a σ_{share} distance (i.e., Euclidean distance) from i . Note that σ_{share} can be calculated dynamically at each iteration step. Fig.3 shows how niche counts are calculated for two candidate solution A and B. Both A and B are on the current non-dominated front. However since A has a smaller niche count than B, A will be preferred over B. This choice has the effect of emphasizing a more diverse non-dominated front.

One of the undesirable features of the niching method employing σ_{share} is that σ_{share} has to be specified by a user, and the model’s performance is highly dependant on the choice of value for this parameter. We adopted the dynamic update of σ_{share} proposed by Fonseca and Fleming [11] so that we do not have to specify σ_{share} . For two objective functions, the following equation is used to determine the σ_{share} dynamically [2]:

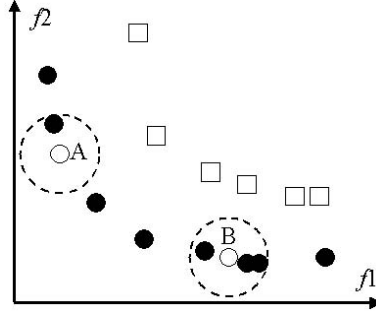


Fig. 3. Niche counts are calculated for particle A and B on the non-dominated front Q (indicated by circles), assuming minimization of f_1 and f_2 .

$$\sigma_{share} = \frac{u_2 - l_2 + u_1 - l_1}{N - 1}, \quad (3)$$

where u_i and l_i are the upper and lower bounds for each of the two objective values for the entire population. Note that as population size increases, the σ_{share} is reduced to accommodate more niches. At each iteration step, we select from the current non-dominated solution set Q those particles with smallest niche counts. Then P_g for each particle is randomly chosen among these “less crowded” non-dominated particles.

Using Crowding Distance Assignments. The 2nd niching method employing crowding distance [9] is also free of choosing such a parameter. Deb et al. [9] in their NSGA II introduced this niching method that makes use of the density of solutions around a particular point on the non-dominated front. The density is estimated by calculating the so-called crowding distance of a point i , which is the average distance of the two point $i-1$ and $i+1$ on either side of this point i along each of the objectives. When we use crowding distance values for niching, we simply sort all the particles of the current Q in descending order, and then choose a particle randomly from the top part of the sorted list (i.e., a particle in the least crowded areas in the Pareto region) as P_g for each particle. This process is repeated for each particle in the population, and over many iteration steps. The complexity of this procedure in the worst case is $O(mN \log N)$, when all particles are in one front. For more information on crowding distance assignment, the readers can refer to [9].

Replacement of “Overcrowded” Particles with New Particles. Another method that can further promote diversity is to remove particles from overcrowded areas on the current non-dominated front Q , and replace them with new particles. We implement this in NSPSO by removing the particle with the largest niche count (or the smallest crowding distance value), and replace it with a randomly generated new particle, at each iteration step. Since the new particle chooses the particle with the smallest niche count, i.e., least crowded (or

the largest crowding distance value), from the current Q as its P_g , this particle should have a better chance to land somewhere “less crowded” on the current front Q .

3.3 NSPSO Algorithm

NSPSO can be summarized in the following steps:

1. Initialize the population and store the population in a list *PSOList*:
 - a) The current position of the i -th particle, X_i and its current velocity V_i , are initialized with random real numbers within the specified decision variable range; V_i has a probability of 0.5 being specified in a different direction; The personal best position P_i , is set to X_i ; V_{MAX} is set to the upper and lower bounds of the decision variable range.
 - b) Evaluate each particle in the population; iteration counter $t := 0$.
2. $t := t + 1$.
3. Identify particles that give non-dominated solutions in the population and store them in a list *nonDomPSOList*.
4. Calculate - a) niche count, or b) crowding distance value, for each particle.
5. Resort the *nonDomPSOList* according to a) niche counts, or b) crowding distance values.
6. *For*($i := 0; i < numParticles; i++$) (step through *PSOList*):
 - a) Select randomly a global best P_g for the i -th particle from a specified top part (e.g. top 5%) of the sorted *nonDomPSOList*.
 - b) Calculate the new velocity V_i , based on the equation (1), and the new X_i by equation (2).
 - c) Add the i -th particle's P_i and the new X_i to a temporary population, stored in *nextPopList*. Note that P_i and X_i now coexist. Also note that *nextPopList* now has a size of $2N$.
 - d) Go to a) if $i < numParticles$.
7. Identify particles that give non-dominated solutions from *nextPopList*, and store them in *nonDomPSOList*. Particles other than non-dominated ones from *nextPopList* are stored in a list *nextPopListRest*.
8. Empty *PSOList* for the next iteration step.
9. Select randomly members of *nonDomPSOList* and add them to *PSOList* (not to exceed *numParticles*).
10. Loop if *PSOList* size $< numParticles$:
 - a) Identify non-dominated particles from *nonDomListRest* and store them in *nextNonDomList*.
 - b) Add members of *nextNonDomList* to *PSOList*, if still the *PSOList* size $< numParticles$.
 - c) Copy *nextPopListRest* to *nextPopListRestCopy*, then empty *nextPopListRest*.
 - d) Assign the vacant *nextPopListRest* with the remaining particles other than non-dominated ones from *nextPopListRestCopy*.
 - e) Go back to a), if still the *PSOList* size $< numParticles$.
11. If $t < maxIterations$, go to 2.
12. Obtain Q from the final population, and calculate the performance metric values (see section 4).

4 Performance Metrics

Diversity of Q . We measure the diversity of solutions along the Pareto front in the final population by comparing the uniform distribution and the deviation of solutions as described by Deb [2]:

$$\Delta = \frac{\sum_{m=1}^M d_m^e + \sum_{i=1}^{|Q|} |d_i - \bar{d}|}{\sum_{m=1}^M d_m^e + |Q| \bar{d}}, \quad (4)$$

where d_i is the distance between two neighbouring solutions in the non-dominated solution set Q , $\bar{d}$ is the mean value of all the d_i . d_m^e is the distance between the extreme solutions of the true Pareto-optimal set P^* and Q on the

m -th objective. d_m^e is calculated by using Schott's difference distance measure [2]. By using d_m^e , equation (4) also takes into account of the extent of the spread. For an ideal distribution of solutions (uniform and $d_m^e = 0.0$), Δ is 0.0. For functions with disconnected sectors on the Pareto front, e.g., ZDT3, Δ is calculated within each continuous sector and then averaged.

Number of non-dominated solutions found. The above diversity metric does not take into account the number of optimal solutions found. An ideal uniform distribution of Q with $\Delta = 0.0$ could have very few solutions. Obviously we prefer a uniform distribution with a larger number of solutions found in the final step of a run.

Closeness to P^* . Generational distance (GD) metric is used to measure the closeness of solutions in Q to P^* . The GD metric finds the average distance of the solutions of Q from P^* [2]:

$$GD = \frac{(\sum_{i=1}^{|Q|} d_i^p)^{1/p}}{|Q|}. \quad (5)$$

For a two objective problem ($p = 2$), d_i is the Euclidean distance between the solution $i \in Q$ and the nearest member of P^* .

5 Experiments

Four test functions ZDT1, ZDT2, ZDT3 and ZDT4 were used [2]. These functions are considered to be difficult because of the large number of decision variables, disconnectedness of P^* , and multiple local fronts. The initial population of the NSPSO was set to 200. c_1 and c_2 were set to 2.0. w was gradually decreased from 1.0 to 0.4. V_{MAX} was set to be the bounds of decision variable ranges, and χ simply to 1.0. NSPSO was run for 100 iteration steps. At the final iteration, the diversity metric Δ and closeness metric GD values were calculated according to equation (4) and (5), and also the number of non-dominated solutions found. A set of $|P^*| = 500$ uniformly distributed Pareto-optimal solutions is used to calculate the GD values. The results of NSPSO were compared with the real-parameter NSGA II. NSGA II also had an initial population of 200, and it was run for 100 generations. As suggested in [2], a crossover probability of 0.9 and a mutation probability of $1/n$ (n is the number of real-variables) were used. The SBX and real-parameter mutation operators, η_c and η_m , were set to 20 respectively. NSPSO and NSGA II were both run 10 times. The results are averaged and summarised in Table 1 - 3. Four NSPSO variants were used: **NC**: using niche count only; **NC-R**: using niche count with replacement; **CD**: using crowding distance only; and **CD-R**: using crowding distance with replacement. Refer to section 3.2 about these four niching variants.

Table 1. Mean and variance values of the GD metric measuring convergence.

Algorithm	ZDT1		ZDT2		ZDT3		ZDT4	
	GD	δ^2	GD	δ^2	GD	δ^2	GD	δ^2
NC	7.97E-04	8.13E-05	8.05E-04	3.05E-05	3.40E-03	2.54E-04	7.82E-04	6.91E-05
NC-R	7.53E-04	4.18E-05	2.93E-02	9.03E-02	3.22E-03	5.31E-04	7.36E-04	5.16E-05
CD	1.05E-03	1.71E-04	8.48E-04	4.57E-05	3.54E-03	4.98E-04	9.03E-04	1.66E-04
CD-R	8.61E-04	1.55E-04	2.93E-02	9.02E-02	3.53E-03	3.79E-04	8.42E-04	1.50E-04
NSGA II	1.14E-03	7.64E-05	8.25E-04	3.26E-05	N/A	N/A	2.92E-02	4.67E-02

Table 2. Mean and variance values of the Δ metric measuring diversity.

Algorithm	ZDT1		ZDT2		ZDT3		ZDT4	
	Δ	δ^2	Δ	δ^2	Δ	δ^2	Δ	δ^2
NC	7.67E-01	3.00E-02	7.58E-01	2.77E-02	9.14E-01	3.93E-02	7.68E-01	3.57E-02
NC-R	7.72E-01	3.58E-02	7.67E-01	4.36E-02	9.04E-01	6.89E-02	8.06E-01	5.05E-02
CD	7.62E-01	3.83E-02	7.48E-01	4.66E-02	8.69E-01	5.81E-02	7.36E-01	1.84E-02
CD-R	7.62E-01	3.17E-02	7.60E-01	4.86E-02	8.65E-01	5.94E-02	7.89E-01	3.98E-02
NSGA II	3.86E-01	1.63E-02	3.90E-01	2.01E-02	N/A	N/A	6.55E-01	1.98E-01

6 Results and Discussion

Table 1 shows that for all 4 functions, NSPSO has no trouble reaching P^* for almost all of the 10 runs within 100 iterations. However an “outlier” was identified for NC-R (ZDT2) and CD-R (ZDT2) each. Closely examining the data, it was found that the poor GD value was the result of a single poor run, but 9 others were in fact very good. Note that NSGA II failed to converge in 100 iteration steps on ZDT3. NSGA II (ZDT4) has the worst variance and GD values, which are due to two runs reaching only local fronts (Fig. 4 (last on the bottom row)). Table 2 shows that NSGA II has a better Δ value overall, whereas NSPSO’s Δ values are higher. However from a typical run as shown in Fig. 4, we can see that NSPSO has a coverage of P^* just as good as NSGA II. The higher Δ values can be attributed to a larger number of different Q found by NSPSO than NSGA II, as shown in Table 3. Since we can obtain the best front (Front 1) with possibly

Table 3. Number of non-dominated solutions found in the final iteration step.

Algorithm	ZDT1	ZDT2	ZDT3	ZDT4
NC	274.3	297	201	276.4
NC-R	277.2	282.6	186.9	286.4
CD	187.9	239.7	134.1	159.6
CD-R	247.8	290.1	194	254.5
NSGA II	200	200	N/A	175.5

more than N non-dominated solutions in the combined population of $2N$ personal bests and offspring, NSPSO is often able to obtain more than 200 different non-dominated solutions in the final step. In contrast, NSGA II generally has a constant number of non-dominated solutions, which is its initial population size (except for ZDT4). To our surprise, NC-R and CD-R did not seem to make much difference in terms of Δ and GD values. However in Table 3, we can note that CD-R managed to find many more different non-dominated solutions than CD. For NC and NC-R, however, there is not much difference. Fig. 4 presents

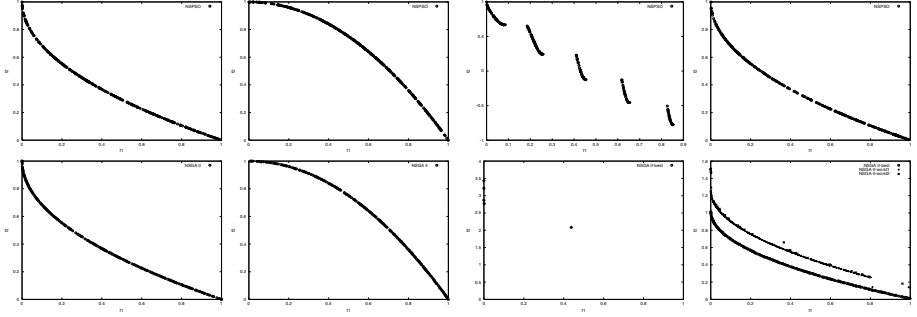


Fig. 4. Non-dominated solutions found by NSPSO (top row) and NSGA II (bottom row) for ZDT1, 2, 3, and 4 (from left to right).

that for a typical NSPSO and NSGA II run, the non-dominated solutions found in the final iteration step for all 4 test functions. Overall, the results show that NSPSO is very competitive in terms of solution spread, coverage and closeness to P^* . Fig. 4 (3rd on the top row) shows that NSPSO has no trouble converging on the different disconnected sectors of P^* for ZDT3, whereas in this case, NSGA II failed completely (3rd on the bottom row). For ZDT4, NSPSO's 10 runs have all converged to P^* with a good spread and coverage. In contrast, NSGA II had 2 out of 10 runs reaching only a local front (Fig. 4 (last on the bottom row)).

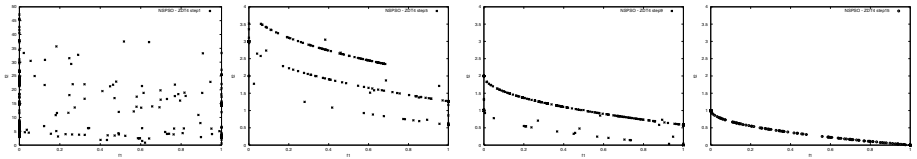


Fig. 5. Snapshots of a NSPSO run showing the entire particle population at step 1, 3, 9 and 15, for solving ZDT4.

Fast and better convergence towards the global front. Fig. 5 presents 4 snapshots of the first few steps in a NSPSO's run for solving ZDT4. It can be

noted that even in step 3, there were no particles that had found P^* , but by step 9 there were quite a few, though the majority of the particles were stuck at the 2nd best front. By step 15, all particles had reached P^* without difficulty. Considering the majority of current multiobjective evolutionary algorithms (with the exception of NSGA II) are unable to converge to P^* for ZDT4, this is a remarkable result. NSPSO also demonstrates its consistency as 10 out of 10 runs have all converged to P^* , better than the NSGA II counterpart. NSPSO's impressive performance is due to the fact that we allow all particles' personal bests and offspring to be combined for non-domination comparison. Updating of the particles is then based on their mutual non-domination relationships with respect to the current Q in the population, not with respect to just another single particle (like the basic PSO). Each time, the best N particles in terms of non-domination levels of the entire population are selected for the next iteration step. Even if a single particle has found P^* , the particle's personal best and its offspring will be emphasised favourably in the next step. Since this personal best and the offspring coexist and they are most likely to be located near P^* , it is likely they will be selected again for the next step. This is more like a "self-proliferation" of good particles. The aggregated effect of this over many steps would produce a large number of fitter particles along P^* . Since at each step we only select the best N particles, particles on those local fronts are gradually phased out, replaced by the fitter ones on the global front.

Using a larger population size. It was found that a reasonably large population is necessary for a good convergence. If population size was too small (e.g., 20), NSPSO could converge to sub-optimal fronts, or to P^* but with a limited number of non-dominated solutions, which is insufficient in terms of solution spread and coverage. This is because a smaller number of particles do not sufficiently sample the search space, and as a result, certain existing particles could quickly become too dominant early on, and they would prevent other potentially good particles (in terms of different non-dominated solutions) from being produced. A large initial population size would allow for a better sampling of the search space, and from there onwards allow NSPSO to better use domination comparison operations to find a wide spread of solutions along P^* .

7 Conclusion

With an aim to improve PSO's effectiveness in utilising the domination comparison operations for solving multiobjective optimization problems, this paper has proposed a modified PSO, NSPSO. The model is able to discover more non-dominated relations by comparing the personal bests and offspring of all particles in a combined swarm population, thereby providing a more appropriate selection pressure for the population to approach the true Pareto-optimal front. NSPSO adopts the non-dominated sorting concept, and uses two parameter-free niching techniques to promote solution diversity. It has been shown that NSPSO and its variants are able to perform remarkably well against some difficult test

functions (functions with high-dimensional decision variables and multiple local fronts) found in the literature. NSPSO is also fast, more reliable, and often converging to the true Pareto-optimal front with a good solution spread and coverage within just a few steps. This once again proves that PSO is a powerful optimization technique that can be used efficiently not only for single objective, but also for multiobjective optimization. Future work will look into application of NSPSO to problems with more than 2 objectives, and also real-world multi-objective optimization problems.

References

1. Zitzler, E., Deb, K. and Thiele, L.: Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary Computation*, **8**(2):173–195, April (2000).
2. Deb, K.: *Multi-Objective Optimization using Evolutionary Algorithms*, John Wiley & Sons, Chichester, UK (2001).
3. Kennedy, J. and Eberhart, R.: Particle Swarm Optimization. In *Proceedings of the Fourth IEEE International Conference on Neural Networks*, Perth, Australia. IEEE Service Center(1995) 1942–1948.
4. Coello, C.A.C. and Lechuga, M.S.: MOPSO: A Proposal for Multiple Objective Particle Swarm Optimization, in *Proceedings of Congress on Evolutionary Computation (CEC'2002)*, Vol. 2, IEEE Press (2002) 1051–1056.
5. Hu, X. and Eberhart, R.: Multiobjective Optimization Using Dynamic Neighbourhood Particle Swarm Optimization. In *Proceedings of the IEEE World Congress on Computational Intelligence*, Hawaii, May 12–17, 2002. IEEE Press (2002).
6. Parsopoulos, K.E. and Vrahatis, M.N.: Particle Swarm Optimization Method in Multiobjective Problems, in *Proceedings of the 2002 ACM Symposium on Applied Computing (SAC'2002)* (2002) 603–607.
7. Fieldsend, E. and Singh, S.: A Multi-Objective Algorithm based upon Particle Swarm Optimisation, an Efficient Data Structure and Turbulence, *Proceedings of the 2002 U.K. Workshop on Computational Intelligence*, Birmingham, UK(2002) 37–44.
8. Horn, J., Nafpliotis, N., and Goldberg, D.E.: A Niche Pareto Genetic Algorithm for Multiobjective Optimization. In *Proceedings of the First IEEE Conference on Evolutionary Computation*, IEEE World Congress on Computational Intelligence, vol: **1**, Piscataway, New Jersey. IEEE Service Center.(1994) 82–87.
9. Deb, K., Agrawal, S. Pratap, A. and Meyarivan, T.: A Fast Elitist NonDominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II. In *Proceedings of Parallel Problem Solving from Nature - PPSN VI*, Springer(2000) 849–858.
10. Goldberg, D.E., and Richardson, J.J.: Genetic Algorithms with sharing for multimodal function optimization. *Genetic Algorithms and Their Applications: Proceedings of the Second ICGA*, Lawrence Erlbaum Associates, Hillsdale, NJ, (1987) 41–49.
11. Fonseca, C.M. and Fleming, P.J.: Genetic algorithms for multiobjective optimization: Formulation, discussion, and generalization. In *Proceedings of the Fifth International Conference on Genetic Algorithms* (1993) 355–365.

The Influence of Run-Time Limits on Choosing Ant System Parameters

Krzysztof Socha

IRIDIA, Université Libre de Bruxelles, CP 194/6,
Av. Franklin D. Roosevelt 50, 1050 Bruxelles, Belgium

ksocha@ulb.ac.be

<http://iridia.ulb.ac.be>

Abstract. The influence of the allowed running time on the choice of the parameters of an ant system is investigated. It is shown that different parameter values appear to be optimal depending on the algorithm run-time. The performance of the *MAX-MIN* Ant System (*MMAS*) on the University Course Timetabling Problem (UCTP) – a type of constraint satisfaction problem – is used as an example. The parameters taken into consideration include the type of the local search used, and some typical parameters for *MMAS* – the τ_{min} and ρ . It is shown that the optimal parameters depend significantly on the time limits set. Conclusions summarizing the influence of time limits on parameter choice, and possible methods of making the parameter choice more independent from the time limits, are presented.

1 Introduction

Ant Colony Optimization (ACO) is a metaheuristic proposed by Dorigo et al. [1]. The inspiration of ACO is the foraging behavior of real ants. The basic ingredient of ACO is the use of a probabilistic solution construction mechanism based on stigmergy. ACO has been applied successfully to numerous combinatorial optimization problems including the traveling salesman problem [2], quadratic assignment problem [3], scheduling problems [4], and others. In this paper we focus on the *MAX-MIN* Ant System [5] – a version of ACO.

ACO, similarly to any other metaheuristic, may be parameterized. This means that in order to be able to deliver optimal performance, the ant algorithm uses a number of parameters that precisely define its operation. The parameters are usually chosen specifically for the class of problems the ant algorithm is to tackle. Usually, the optimal (or near optimal) parameters are chosen by the trial-and-error procedure. As the number of trials necessary to fine-tune the parameters is usually quite high, it is often the case that limited time is used for each algorithm run. The best parameters found are then used for tackling actual problems. In turn, the actual problem solving runs tend to be much longer.

We do not focus on the actual results obtained by the algorithm, but only on choosing the best set of parameters optimizing algorithm performance within allowed run-time limit. This paper attempts to show that optimal parameters

for the ant algorithm depend significantly on the time given it to run. Hence, the parameter fine-tuning done with run-times significantly shorter than actual problem solving runs may cause choosing suboptimal parameters. It is assumed that for the time limits investigated, the algorithm does not reach optimality – it does not find the optimal solution.

The remaining part of the paper is organized as follows: Section 2 briefly presents the example problem used for evaluating the performance of the ant algorithm. Section 3 presents the ant system used to solve the problem – the *MAX-MIN* Ant System. The main design considerations and parameters used are highlighted. Section 4 presents the local search used by the algorithm, and discusses how time limits imposed determine the choice of the local search type. Section 5 discusses some other parameters used by the algorithm, and presents the relationship between their optimal values and the time limits imposed. Finally, Section 6 summarizes the findings and presents the conclusions drawn.

2 Example Problem

The problem used to illustrate the thesis of this paper, is the University Course Timetabling Problem (UCTP) [6,7,8]. It is a type of constraint satisfaction problem. It consists of a set of n events $E = \{e_1, \dots, e_n\}$ to be scheduled in a set of i timeslots $T = \{t_1, \dots, t_i\}$, and a set of j rooms $R = \{r_1, \dots, r_j\}$ in which events can take place. Additionally, there is defined a set of students S who attend the events, and a set of features F satisfied by rooms and required by events. Each student is already preassigned to a subset of events. A feasible timetable is one in which all events have been assigned a timeslot and a room, so that the following hard constraints are satisfied:

- no student attends more than one event at the same time;
- the room is big enough for all the attending students and satisfies all the features required by the event;
- only one event is taking place in each room at a given time.

In addition, a feasible candidate timetable is penalized equally for each occurrence of the following soft constraint violations:

- a student has a class in the last slot of the day;
- a student has more than two classes in a row (one penalty for each class above the first two);
- a student has exactly one class during a day.

The infeasible timetables are worthless and are considered equally bad regardless of the actual level of infeasibility. The objective is to minimize the number of soft constraint violations ($\#scv$) in a feasible timetable. The solution to the UCTP is a mapping of events into particular timeslots and rooms. Fig. 1 shows an example of a timetable.

	r_1	r_2	r_3	...	r_j
t_1	e_{23}	e_{55}	e_{32}		
t_2		e_{12}			e_{54}
t_3	e_{89}	e_{19}	e_i		e_{18}
...					
t_i	e_7	e_{76}			e_2

Fig. 1. Timetable of i timeslots and j rooms ($k = i \cdot j$ places). Some events from the set E have already been placed.

Two instances of the UCTP are used for illustrating the performance of the ant algorithm in this paper – **competition04** and **competition07**. These instances have been proposed as a part of the International Timetabling Competition¹. Note that these instances are known to have a perfect solution, i.e. a solution where no hard or soft constraints are violated.

3 Algorithm Description

The basic mode of operation of the *MAX-MIN* Ant System used for the experiments is as follows. At each iteration of the algorithm, each of the m ants constructs a complete assignment C of events into timeslots and rooms. Following a pre-ordered list of events, the ants choose the timeslot and room for the given event probabilistically, guided by stigmergic information. This information is in the form of a matrix of *pheromone* values $\tau : E \times T \times R \rightarrow \mathbf{R}^+$, where E is the set of events, T is the set of timeslots, and R is a set of rooms. In order to maintain simplicity of the notation, let us call the timeslot-room combination a *place*. The pheromone matrix becomes then of form: $\tau : E \times P \rightarrow \mathbf{R}^+$, where P is the set of k places, and $k = |P| = |T| \cdot |R| = i \cdot j$.

Also, some problem specific knowledge (heuristic information) is used by the algorithm. The place for an event (i.e. the timeslot-room combination) is chosen only from the ones that are suitable for the given event - placing the event there will not violate any hard constraint. If at some point of time during the construction of the assignment there is no such a place available, a list of timeslots is extended by one, and the event is placed in one of the rooms of this additional timeslot. This of course results in an infeasible solution² as the number of timeslots used from now on exceeds i . This also means that the pheromone matrix has to be extended as well. It is done by creating an extended set T' of i' timeslots, and consequently a new extended set P' of k' places. The new pheromone matrix is defined as $\tau : E \times P' \rightarrow \mathbf{R}^+$. Note that initially it is assumed that $i' = i$ and $k' = k$.

¹ <http://www.idsia.ch/Files/ttcomp2002/>

² Only for this particular ant, and only in this iteration.

Once all the ants have constructed their assignment of events into places, a local search routine is used to further improve the solutions. More details about local search routine are provided in Sec. 4. Finally the best solution of each iteration is compared to the global best solution found so far. If the iteration best solution is better than the global best, it is replaced. Only the global best solution is used for the pheromone update.

If the differences between extreme pheromone values were too large, all ants would almost always generate the same solutions, which would mean algorithm stagnation. The $\mathcal{M}\mathcal{A}\mathcal{X}$ - $\mathcal{M}\mathcal{I}\mathcal{N}$ Ant System introduces upper and lower limits on the pheromone values – τ_{max} and τ_{min} respectively [5] – that prevent this. The maximal difference between the extreme levels of pheromone may be controlled, and thus the search intensification versus diversification may be balanced. The pheromone update rule is as follows (for the particular case of assigning events e into places p):

$$\tau_{(e,p)} \leftarrow \begin{cases} (1 - \rho) \cdot \tau_{(e,p)} + \tau_{fixed} & \text{if } (e, p) \text{ is in } C_{global\_best} \\ (1 - \rho) \cdot \tau_{(e,p)} & \text{otherwise,} \end{cases} \quad (1)$$

where $\rho \in [0, 1]$ is the evaporation rate, and τ_{fixed} is the pheromone update value. Pheromone update is completed using the following:

$$\tau_{(e,p)} \leftarrow \begin{cases} \tau_{min} & \text{if } \tau_{(e,p)} < \tau_{min}, \\ \tau_{max} & \text{if } \tau_{(e,p)} > \tau_{max}, \\ \tau_{(e,p)} & \text{otherwise.} \end{cases} \quad (2)$$

The pheromone update value τ_{fixed} is a constant that has been established after some experiments with the values calculated based on the actual quality of the solution. The function q measures the quality of a candidate solution C by counting the number of constraint violations. According to the definition of $\mathcal{M}\mathcal{M}\mathcal{A}\mathcal{S}$, $\tau_{max} = \frac{1}{\rho} \cdot \frac{g}{1+q(C_{optimal})}$, where g is a scaling factor. Since it is known that $q(C_{optimal}) = 0$ for the considered test instances, we set τ_{max} to a fixed value $\tau_{max} = \frac{1}{\rho}$. We observed that the proper balance of the pheromone update and the evaporation rate was achieved with a constant value of $\tau_{fixed} = 1.0$, which was also more efficient than the calculation of exact value based on quality of the solution.

4 Influence of Local Search

It has been shown in the literature that ant algorithms perform particularly well, when supported by a local search (LS) routine [2,9,10]. There were also attempts to design the local search for the particular problem tackled here (the UCTP) [11]. Here, we try to show that although adding an LS to an algorithm improves the results obtained, it is important to carefully choose the type of such LS routine, especially with regard to algorithm running time limits imposed.

The LS used here by the $\mathcal{M}\mathcal{M}\mathcal{A}\mathcal{S}$ solving the UCTP consists of two major modules. The first module tries to improve an infeasible solution (i.e. a solution

that uses more than i timeslots), so that it becomes feasible. Since its main purpose is to produce a solution that does not contain any hard constraint violations and that fits into i timeslots, we call it **HardLS**. The second module of the LS is run only if a feasible solution is available (either generated by an ant directly, or obtained after running **HardLS**). This module tries to increase the quality of the solution by reducing number of the soft constraint violations ($\#scv$), and hence is called **SoftLS**. It does so by rearranging the events in the timetable, but any such rearrangement must never produce an infeasible solution.

The **HardLS** module is always called before calling the **SoftLS** module, if the solution found by an ant is infeasible. Also, it is not parameterized in any way, so in this paper we will not go into details of its operation.

SoftLS rearranges the events aiming at increasing the quality of the already feasible solution, without introducing infeasibility. This means that an event may only be placed in timeslot $t_{l:l \leq i}$. In the process of finding the most efficient LS, we developed the following three types of **SoftLS**:

- **type 0** – The simplest and the fastest version. It tries to move one event at a time to an empty place that is suitable for this event, so that after such a move the quality of the solution is improved. The starting place is chosen randomly, and then the algorithm loops through all the places trying to put the events in empty places until a perfect solution is found, or until in the last $k = |P|$ iterations there was no improvement.
- **type 1** – Version similar to the **SoftLS type 0**, but also enhanced by the ability to swap two events in one step. The algorithm not only checks, if an event may be moved to another empty suitable place to improve the solution, but also checks, if this event could perhaps be swapped with any other event. Only moves (or swaps) that do not violate any hard constraints and improve the overall solution are accepted. This version of **SoftLS** usually provides a greater solution improvement than the **SoftLS type 0**, but also a single run takes significantly more time.
- **type 2** – The most complex version. In this case, as a first step, the **SoftLS type 1** is run. After that, the second step is executed: the algorithm tries to further improve the solution by changing the order of timeslots. It attempts to swap any two timeslots (i.e. move all the events from one timeslot to the other without changing the room assignment), so the solution is improved. The operation continues until no swaps of any two timeslots may further improve the solution. The two steps are repeated until a perfect solution is found, or neither of them has produced any improvement. This version of **SoftLS** is the most time consuming.

4.1 Experimental Results

We ran several experiments in order to establish, which of the presented **SoftLS** types is best suited for the problem being solved. Fig. 2 presents the performance of our ant algorithm with different versions of **SoftLS**, as a function of time limit

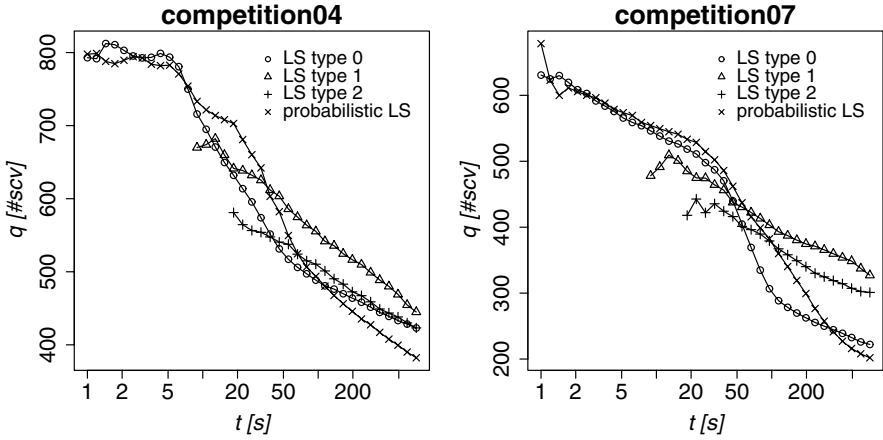


Fig. 2. Mean value of the quality of the solutions ($\#scv$) generated by the $\mathcal{MMAS}$ using different versions of local search on two instances of the UCTP – **competition04** and **competition07**.

imposed on the algorithm run-time. Note that we initially focus here on the three basic types of **SoftLS**. The additional **SoftLS** type – probabilistic LS – that is also presented on this figure, is described in more detail in Sec. 4.2.

We ran 100 trials for each of the **SoftLS** types. The time limit imposed on each run was 672 seconds (chosen with the use of benchmark program supplied by Ben Peachter as part of the International Timetabling Competition). We measured the quality of the solution throughout the duration of each run. All the experiments were conducted on the same computer (AMD Athlon 1100 MHz, 256 MB RAM) under a Linux operating system.

Fig. 2 clearly indicates the differences in performance of the $\mathcal{MMAS}$, when using different types of **SoftLS**. While the **SoftLS** type 0 produces first results already within the first second of the run, the other two types of **SoftLS** produce first results only after 10-20 seconds. However, the first results produced by either the **SoftLS** type 1 or type 2 are significantly better than the results obtained by the **SoftLS** type 0 within the same time. With the increase of allowed algorithm run-time, the **SoftLS** type 0 quickly outperforms **SoftLS** type 1, and then type 2. While in case of **competition07**, the **SoftLS** type 0 remains the best within the imposed time limit (i.e. 672 seconds), in case of **competition04**, the **SoftLS** type 2 apparently eventually catches up. This may indicate that if more time was allowed for each version of the algorithm to run, the best results may be obtained by **SoftLS** type 2, rather than type 0. It is also visible that towards the end of the search process, the **SoftLS** type 1 appears to converge faster than type 0 or type 2 for both test instances. Again, this may indicate that – if longer run-time was allowed – the best **SoftLS** type may be different yet again.

It is hence very clear that the best of the three presented types of local search for the UCTP may only be chosen after defining the time limit for a single algorithm run. The examples of time limits and appropriate best LS type are summarized in Tab. 1.

Table 1. Best type of the **SoftLS** depending on example time limits.

Time Limit [s]	Best SoftLS Type	
	competition04	competition07
5	type 0	type 0
10	type 1	type 1
20	type 2	type 2
50	type 0	type 2
200	type 0	type 0
672	type 0/2	type 0

4.2 Probabilistic Local Search

After experimenting with the basic types of **SoftLS** presented in Sec. 4, we realized that apparently different types of **SoftLS** work best during different stages of the search process. We wanted to find a way to take advantage of all of the types of **SoftLS**.

First, we thought of using a particular type of **SoftLS** depending on the time spent by the algorithm on searching. However this approach, apart from having an obvious disadvantage of the necessity of measuring time and being dependent on the hardware used, had some additional problems. We found that the solution (however good it was) generated with the use of any basic type of **SoftLS**, was not always easy to be further optimized by another type of **SoftLS**. When the type of **SoftLS** used changed, the algorithm spent some time recovering from the previously found local optimum. Also, the sheer necessity of defining the right moments, when the **SoftLS** type was to be changed was a problem. It had to be done for each problem instance separately, as those times differed significantly from instance to instance.

In order to overcome these difficulties, we came up with the idea of *probabilistic local search*. Such local search would probabilistically choose the basic type of the **SoftLS** to be used. Its behavior may be controlled by proper adjustment of the probabilities of running the different basic types of **SoftLS**. After some initial tests, we found that rather small probability of running the **SoftLS** type 1 and type 2 comparing to the probability of running the **SoftLS** type 0, produced best results within the time limit defined. Fig. 2 also presents the mean values obtained by 100 runs of this probabilistic local search. The probabilities of running each type of the basic **SoftLS** types that were used to obtain these results, are listed in Tab. 2.

The performance of the probabilistic **SoftLS** is apparently the worst for around first 50 seconds of the run-time for both test problem instances. After

Table 2. Probabilities of running different types of the **SoftLS**.

SoftLS Type	Probabilities	
	competition04	competition07
type 0	0.90	0.94
type 1	0.05	0.03
type 2	0.05	0.03

that, it improves faster than the performance of any other type of **SoftLS**, and eventually becomes the best. In case of the **competition04** problem instance, it becomes the best already after around 100 seconds of the run-time, and in case of the **competition07** problem instance, after around 300 seconds.

It is important to note that the probabilities of running the basic types of **SoftLS** have been chosen in such a way that this probabilistic **SoftLS** is in fact very close to the **SoftLS type 0**. Hence, its characteristics are also similar. However, by appropriately modifying the probability parameters, the behavior of this probabilistic **SoftLS** may be adjusted, and hence provide good results for any given time limits. In particular, the probabilistic **SoftLS** may be reduced to any of the basic versions of **SoftLS**.

5 ACO Specific Parameters

Having shown in Sec. 4 that choice of the best type of local search very much depends on the time the algorithm is run, we wanted to see if this also applies to other algorithm parameters. Another aspect of the **MAX-MIN** Ant System that we investigated with regard to the imposed time limits, was a subset of the typical **MMAS** parameters: evaporation rate ρ and pheromone lower bound τ_{min} . We chose these two parameters among others, as they have been shown in the literature [12,10,5] to have significant impact on the results obtained by a **MAX-MIN** Ant System.

We generated 110 different sets of these two parameters. We chose the evaporation rate $\rho \in [0.05, 0.50]$ with the step of 0.05, and the pheromone lower bound $\tau_{min} \in [6.25 \cdot 10^5, 6.4 \cdot 10^3]$ with the logarithmic step of 2. This gave 10 different values of ρ and 11 different values of τ_{min} – 110 possible pairs of values. For each such pair, we ran the algorithm 10 times with the time limit set to 672 seconds. We measured the quality of the solution throughout the duration of each run for all the 110 cases. Fig. 3 presents the gray-shade-coded grid of ranks of mean solution values obtained by the algorithm with different sets of the parameters for four different run-times allowed (respectively 8, 32, 128, and 672 seconds)³. The results presented, were obtained for the **competition04** instance.

The results indicate that the best solutions – those with higher ranks (darker) – are found for different sets of parameters, depending on the allowed run-time

³ The ranks were calculated independently for each time limit studied.

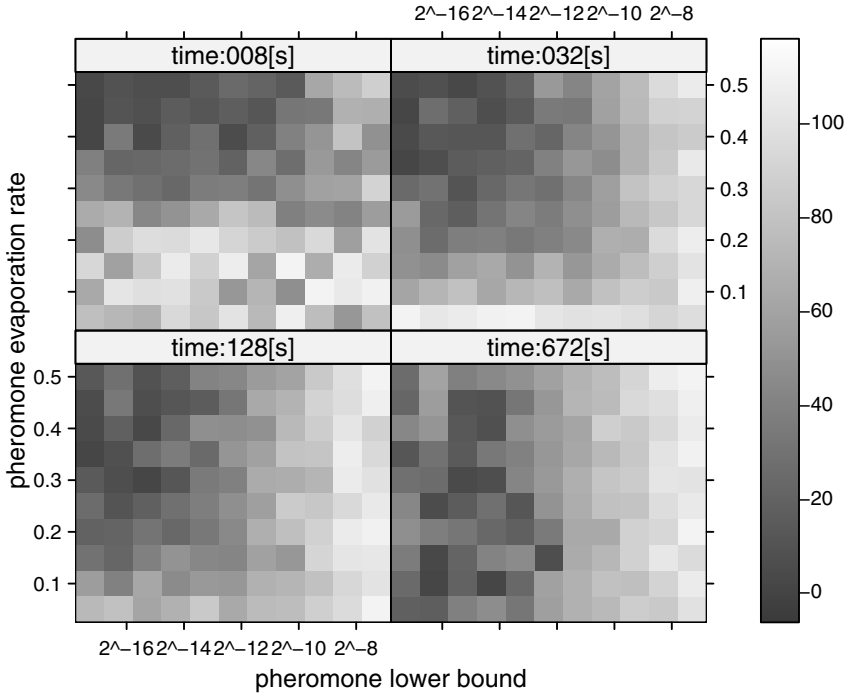


Fig. 3. The ranks of the solution means for the `competition04` instance with regard to the algorithm run-time. The ranks of the solutions are depicted (gray-shade-coded) as function of the pheromone lower bound τ_{min} , and pheromone evaporation rate ρ .

limit. In order to be able to analyse the relationship between the best solutions obtained and the algorithm run-time more closely, we calculated the mean value of the results for 16 best pairs of parameters, for several time limits between 1 and 672 seconds. The outcome of that analysis is presented on Fig. 4. The figure presents respectively: the average best evaporation rate as a function of algorithm run-time: $\rho(t)$, the average best pheromone lower bound as a function of run-time: $\tau_{min}(t)$, and also how the pair of the best average ρ and τ_{min} , changes with run-time. Additionally, it shows how the average best solution obtained with the current best parameters change with algorithm run-time: $q(t)$.

It is clearly visible that the average best parameters change with the change of run-time allowed. Hence, similarly as in case of the local search, the choice of parameters should be done with close attention to the imposed time limits. At the same time, it is important to mention that the probabilistic method of choosing the configuration that worked well in the case of the `SoftLS`, is rather difficult to implement in case of the `MMAS` specific parameters. Here, the change of parameters' values has its effect on algorithm behavior only after several iterations, rather than immediately as in case of LS. Hence, rapid changes

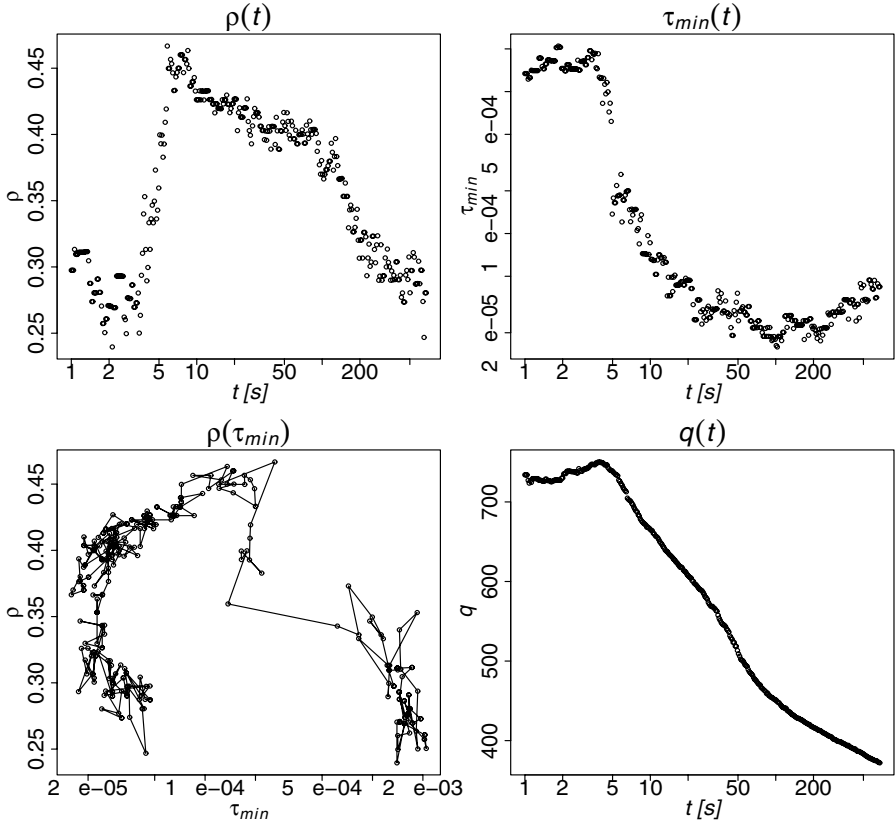


Fig. 4. Analysis of average best ρ and τ_{min} parameters as a function of time assigned for the algorithm run (the upper charts). Also, the relation between best values of ρ and τ_{min} , as changing with running time, and the average quality of the solutions obtained with the current best parameters as a function of run-time (lower charts).

of these parameters may only result in algorithm behavior that would be similar to simply using the average values of the probabilistically chosen ones.

More details about the experiments conducted, as well as the source code of the algorithm used, and also results for other test instances that could not be included in the text due to the limited length of this paper, may be found on the Internet⁴.

6 Conclusions and Future Work

Based on the examples presented, it is clear that the optimal parameters of the *MAX-MIN* Ant System may only be chosen with close attention to the run-

⁴ <http://iridia.ulb.ac.be/~ksocha/antparam03.html>

time limits. Hence, the time-limits have to be clearly defined before attempting to fine-tune the parameters. Also, the test runs used to adjust the parameter values should be conducted under the same conditions as the actual problem solving runs.

In case of some parameters, such as the type of the local search to be used, a probabilistic method may be used to obtain very good results. For some other types of parameters (τ_{min} and ρ in our example) such a method is not so good, and some other approach is needed. The possible solution is to make the parameter values variable throughout the run of the algorithm. The variable parameters may change according to a predefined sequence of values, or they may be adaptive – the changes may be a derivative of a certain algorithm state.

This last idea seems especially promising. The problem however is to define exactly how the state of the algorithm should influence the parameters. To make the performance of the algorithm independent from the time limits imposed on the run-time, several runs are needed. During those runs, the algorithm (or at least algorithm designer) may *learn* what is the relation between the algorithm state, and the optimal parameter values. It remains an open question how difficult it would be to design such a *self-fine-tuning* algorithm, or how much time such an algorithm would need in order to learn.

6.1 Future Work

In the future, we plan to investigate further the relationship between different ACO parameters and run-time limits. This should include the investigation of other test instances, and also other example problems. We will try to define a mechanism that would allow a dynamic adaptation of the parameters. Also, it is very interesting to see if the parameter-runtime relation is similar (or the same) regardless of the instance or problem studied (at least for some ACO parameters). If so, this could permit proposing a general framework of ACO parameter adaptation, rather than a case by case approach.

We believe that the results presented in this paper may also be applicable to other combinatorial optimization problems solved by ant algorithms. In fact it is very likely that they are also applicable to other metaheuristics as well⁵. The results presented in this paper do not yet allow to simply jump to such conclusions however. We plan to continue the research to show that it is in fact the case.

Acknowledgments. Our work was supported by the *Metaheuristics Network*, a Research Training Network funded by the Improving Human Potential Programme of the CEC, grant HPRN-CT-1999-00106. The information provided is the sole responsibility of the authors and does not reflect the Community's opinion. The Community is not responsible for any use that might be made of data appearing in this publication.

⁵ Of course with regard to their specific parameters.

References

1. Dorigo, M., Maniezzo, V., Coloni, A.: The ant system: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics* **26** (1996) 29–41
2. Stützle, T., Dorigo, M.: Aco algorithms for the traveling salesman problem. In Makela, M., Miettinen, K., Neittaanmäki, P., Périaux, J., eds.: *Proceedings of Evolutionary Algorithms in Engineering and Computer Science: Recent Advances in Genetic Algorithms, Evolution Strategies, Evolutionary Programming, Genetic Programming and Industrial Applications (EUROGEN 1999)*, John Wiley & Sons (1999)
3. Stützle, T., Dorigo, M. In: *ACO Algorithms for the Quadratic Assignment Problem*. McGraw-Hill (1999)
4. Merkle, D., Middendorf, M., Schneck, H.: Ant colony optimization for resource-constrained project scheduling. In: *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2000)*, Morgan Kaufmann Publishers (2000) 893–900
5. Stützle, T., Hoos, H.H.: *MAX-MIN* Ant System. *Future Generation Computer Systems* **16** (2000) 889–914
6. Rossi-Doria, O., Sampels, M., Chiarandini, M., Knowles, J., Manfrin, M., Mastrolilli, M., Paquete, L., Paechter, B.: A comparison of the performance of different metaheuristics on the timetabling problem. In: *Proceedings of the 4th International Conference on Practice and Theory of Automated Timetabling (PATAT 2002)* (to appear). (2002)
7. Socha, K., Knowles, J., Sampels, M.: A *MAX-MIN* Ant System for the University Timetabling Problem. In Dorigo, M., Di Caro, G., Sampels, M., eds.: *Proceedings of ANTS 2002 – Third International Workshop on Ant Algorithms*. *Lecture Notes in Computer Science*, Springer Verlag, Berlin, Germany (2002)
8. Socha, K., Sampels, M., Manfrin, M.: Ant Algorithms for the University Course Timetabling Problem with Regard to the State-of-the-Art. In: *Proceedings of EvoCOP 2003 – 3rd European Workshop on Evolutionary Computation in Combinatorial Optimization*, LNCS 2611. Volume 2611 of *Lecture Notes in Computer Science*, Springer, Berlin, Germany (2003)
9. Maniezzo, V., Carbonaro, A.: Ant Colony Optimization: an Overview. In Ribeiro, C., ed.: *Essays and Surveys in Metaheuristics*, Kluwer Academic Publishers (2001)
10. Stützle, T., Hoos, H. In: *The MAX-MIN Ant System and Local Search for Combinatorial Optimization Problems: Towards Adaptive Tools for Combinatorial Global Optimisation*. Kluwer Academic Publishers (1998) 313–329
11. Burke, E.K., Newall, J.P., Weare, R.F.: A memetic algorithm for university exam timetabling. In: *Proceedings of the 1st International Conference on Practice and Theory of Automated Timetabling (PATAT 1995)*, LNCS 1153, Springer-Verlag (1996) 241–251
12. Stützle, T., Hoos, H.: Improvements on the ant system: A detailed report on max-min ant system. Technical Report AIDA-96-12 – Revised version, Darmstadt University of Technology, Computer Science Department, Intellectics Group (1996)

Emergence of Collective Behavior in Evolving Populations of Flying Agents

Lee Spector¹, Jon Klein^{1,2}, Chris Perry¹, and Mark Feinstein¹

¹ School of Cognitive Science, Hampshire College
Amherst, MA 01002, USA

² Physical Resource Theory, Chalmers U. of Technology and Göteborg University
SE-412 96 Göteborg, Sweden

{lspector, jklein, perry, mfeinstein}@hampshire.edu
<http://hampshire.edu/lspector>

Abstract. We demonstrate the emergence of collective behavior in two evolutionary computation systems, one an evolutionary extension of a classic (highly constrained) flocking algorithm and the other a relatively un-constrained system in which the behavior of agents is governed by evolved computer programs. We describe the systems in detail, document the emergence of collective behavior, and argue that these systems present new opportunities for the study of group dynamics in an evolutionary context.

1 Introduction

The evolution of group behavior is a central concern in evolutionary biology and behavioral ecology. Ethologists have articulated many costs and benefits of group living and have attempted to understand the ways in which these factors interact in the context of evolving populations. For example, they have considered the thermal advantages that warm-blooded animals accrue by being close together, the hydrodynamic advantages for fish swimming in schools, the risk of increased incidence of disease in crowds, the risk of cuckoldry by neighbors, and many advantages and risks of group foraging [4]. Attempts have been made to understand the evolution of group behavior as an optimization process operating on these factors, and to understand the circumstances in which the resulting optima are stable or unstable [6], [10]. Similar questions arise at a smaller scale and at an earlier phase of evolutionary history with respect to the evolution of symbiosis, multicellularity, and other forms of aggregation that were required to produce the first large, complex life forms [5], [1].

Artificial life technologies provide new tools for the investigation of these issues. One well-known, early example was the use of the Tierra system to study the evolution of a simple form of parasitism [7]. Game theoretic simulations, often based on the Prisoner's Dilemma, have provided ample data and insights, although usually at a level of abstraction far removed from the physical risks and opportunities presented by real environments (see, e.g., [2], about which we say a bit more below). Other investigators have attempted to study the evolution of

collective behavior in populations of flying or swimming agents that are similar in some ways to those investigated here, with varying degrees of success [8], [13]. The latest wave of artificial life technology presents yet newer opportunities, however, as it is now possible to conduct much more elaborate simulations on modest hardware and in short time spans, to observe both evolution and behavior in real time in high-resolution 3D displays, and to interactively explore the ecology of evolving ecosystems.

In the present paper we describe two recent experiments in which the emergence of collective behavior was observed in evolving populations of flying agents. The first experiment used a system, called SWARMEVOLVE 1.0, that extends a classic flocking algorithm to allow for multiple species, goal orientation, and evolution of the constants in the hard-coded motion control equation. In this system we observed the emergence of a form of collective behavior in which species act similarly to multicellular organisms. The second experiment used a later and much-altered version of this system, called SWARMEVOLVE 2.0, in which the behavior of agents is controlled by evolved computer programs instead of a hard-coded motion control equation.¹ In this system we observed the emergence of altruistic food-sharing behaviors and investigated the link between this behavior and the stability of the environment.

Both SWARMEVOLVE 1.0 and SWARMEVOLVE 2.0 were developed within BREVE, a simulation package designed by Klein for realistic simulations of decentralized systems and artificial life in 3D worlds [3]. BREVE simulations are written by defining the behaviors and interactions of agents using a simple object-oriented programming language called STEVE. BREVE provides facilities for rigid body simulation, collision detection/response, and articulated body simulation. It simplifies the rapid construction of complex multi-agent simulations and includes a powerful OpenGL display engine that allows observers to manipulate the perspective in the 3D world and view the agents from any location and angle. The display engine also provides several “special effects” that can provide additional visual cues to observers, including shadows, reflections, lighting, semi-transparent bitmaps, lines connecting neighboring objects, texturing of objects and the ability to treat objects as light sources. More information about BREVE can be found in [3]. The BREVE system itself can be found on-line at <http://www.spiderland.org/breve>.

In the following sections we describe the two SWARMEVOLVE systems and the collective behavior phenomena that we observed within them. This is followed by some brief remarks about the potential for future investigations into the evolution of collective behavior using artificial life technology.

¹ A system that appears to be similar in some ways, though it is based on 2D cellular automata and the Santa Fe Institute SWARM system, is described at <http://omicrongroup.org/evo/>.

2 SWARMEVOLVE 1.0

One of the demonstration programs distributed with BREVE is SWARM, a simulation of flocking behavior modeled on the “boids” work of Craig W. Reynolds [9]. In the BREVE SWARM program the acceleration vector for each agent is determined at each time step via the following formulae:

$$\mathbf{V} = c_1 \mathbf{V}_1 + c_2 \mathbf{V}_2 + c_3 \mathbf{V}_3 + c_4 \mathbf{V}_4 + c_5 \mathbf{V}_5$$

$$\mathbf{A} = m \left(\frac{\mathbf{V}}{|\mathbf{V}|} \right)$$

The c_i are constants and the $\mathbf{V}_i$ are vectors determined from the state of the world (or in one case from the random number generator) and then normalized to length 1. $\mathbf{V}_1$ is a vector away from neighbors that are within a “crowding” radius, $\mathbf{V}_2$ is a vector toward the center of the world, $\mathbf{V}_3$ is the average of the agent’s neighbors’ velocity vectors, $\mathbf{V}_4$ is a vector toward the center of gravity of all agents, and $\mathbf{V}_5$ is a random vector. In the second formula we normalize the resulting velocity vector to length 1 (assuming its length is not zero) and set the agent’s acceleration to the product of this result and m , a constant that determines the agent’s maximum acceleration. The system also models a floor and hard-coded “land” and “take off” behaviors, but these are peripheral to the focus of this paper. By using different values for the c_i and m constants (along with the “crowding” distance, the number of agents, and other parameters) one can obtain a range of different flocking behaviors; many researchers have explored the space of these behaviors since Reynolds’s pioneering work [9].

SWARMEVOLVE 1.0 enhances the basic BREVE SWARM system in several ways. First, we created three distinct species² of agents, each designated by a different color. As part of this enhancement we added a new term, $c_6 \mathbf{V}_6$, to the motion formula, where $\mathbf{V}_6$ is a vector away from neighbors *of other species* that are within a “crowding” radius. Goal-orientation was introduced by adding a number of randomly moving “energy” sources to the environment and imposing energy dynamics. As part of this enhancement we added one more new term, $c_7 \mathbf{V}_7$, to the motion formula, where $\mathbf{V}_7$ is a vector toward the nearest energy source. Each time an agent collides with an energy source it receives an energy boost (up to a maximum), while each of the following bears an energy cost:

- Survival for a simulation time step (a small “cost of living”).
- Collision with another agent.
- Being in a neighborhood (bounded by a pre-set radius) in which representatives of the agent’s species are outnumbered by representatives of other species.
- Giving birth (see below).

² “Species” here are simply imposed, hard-coded distinctions between groups of agents, implemented by filling “species” slots in the agent data structures with integers ranging from 0 to 2. This bears only superficial resemblance to biological notions of “species.”

The numerical values for the energy costs and other parameters can be adjusted arbitrarily and the effects of these adjustments can be observed visually and/or via statistics printed to the log file; values typical of those that we used can be found in the source code for SWARMEVOLVE 1.0.³

As a final enhancement we leveraged the energy dynamics to provide a fitness function and used a genetic encoding of the control constants to allow for evolution. Each individual has its own set of c_i constants; this set of constants controls the agent's behavior (via the enhanced motion formula) and also serves as the agent's genotype. When an agent's energy falls to zero the agent "dies" and is "reborn" (in the same location) by receiving a new genotype and an infusion of energy. The genotype is taken, with possible mutation (small perturbation of each constant) from the "best" current individual of the agent's species (which may be at a distant location).⁴ We define "best" here as the *product* of energy and age (in simulation time steps). The genotype of the "dead" agent is lost, and the agent that provided the genotype for the new agent pays a small energy penalty for giving birth. Note that reproduction is asexual in this system (although it may be sexual in SWARMEVOLVE 2.0).

The visualization system presents a 3D view (automatically scaled and targeted) of the geometry of the world and all of the agents in real time. Commonly available hardware is sufficient for fluid action and animation. Each agent is a cone with a pentagonal base and a hue determined by the agent's species (red, blue, or purple). The color of an agent is dimmed in inverse proportion to its energy — agents with nearly maximal energy glow brightly while those with nearly zero energy are almost black. "Rebirth" events are visible as agents flash from black to bright colors.⁵ Agent cones are oriented to point in the direction of their velocity vectors. This often produces an appearance akin to swimming or to "swooping" birds, particularly when agents are moving quickly. Energy sources are flat, bright yellow pentagonal disks that hover at a fixed distance above the floor and occasionally glide to new, random positions within a fixed distance from the center of the world. An automatic camera control algorithm adjusts camera zoom and targeting continuously in an attempt to keep most of the action in view.

Figure 1 shows a snapshot of a typical view of the SWARMEVOLVE world. An animation showing a typical action sequence can be found on-line.⁶

SWARMEVOLVE 1.0 is simple in many respects but it nonetheless exhibits rich evolutionary behavior. One can often observe the species adopting different strategies; for example, one species often evolves to be better at tracking quickly moving energy sources, while another evolves to be better at capturing static en-

³ <http://hampshire.edu/ljspector/swarmevolve-1.0.tz>

⁴ The choice to have death and rebirth happen in the same location facilitated, as an unanticipated side effect, the evolution of the form of collective behavior described below. In SWARMEVOLVE 2.0, among many other changes, births occur near parents.

⁵ Birth energies are typically chosen to be random numbers in the vicinity of half of the maximum.

⁶ <http://hampshire.edu/ljspector/swarmevolve-ex1.mov>

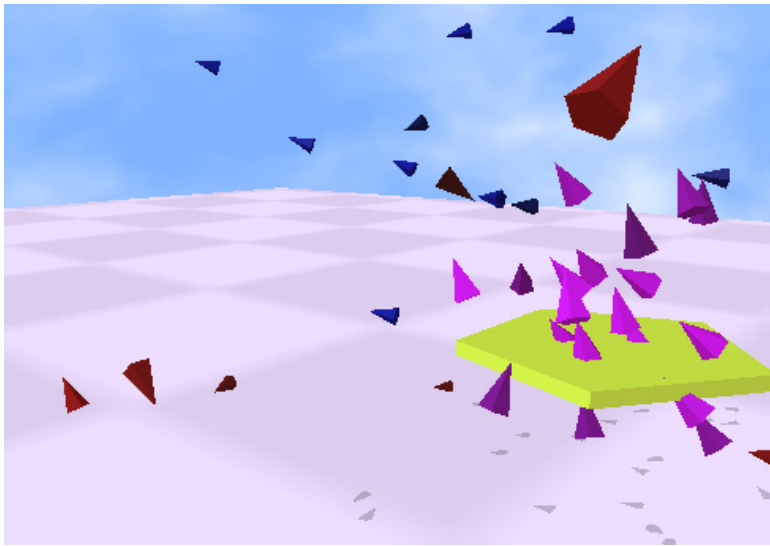


Fig. 1. A view of SWARMEVOLVE 1.0 (which is in color but will print black and white in the proceedings). The agents in control of the pentagonal energy source are of the purple species, those in the distance in the upper center of the image are blue, and a few strays (including those on the left of the image) are red. All agents are the same size, so relative size on screen indicates distance from the camera.

ergy sources from other species. An animation demonstrating evolved strategies such as these can be found on-line.⁷

3 Emergence of Collective Behavior in SWARMEVOLVE 1.0

Many SWARMEVOLVE runs produce at least some species that tend to form static clouds around energy sources. In such a species, a small number of individuals will typically hover within the energy source, feeding continuously, while all of the other individuals will hover in a spherical area surrounding the energy source, maintaining approximately equal distances between themselves and their neighbors. Figure 2 shows a snapshot of such a situation, as does the animation at <http://hampshire.edu/lsector/swarmevolve-ex2.mov>; note the behavior of the purple agents.

We initially found this behavior puzzling as the individuals that are not actually feeding quickly die. On first glance this does not appear to be adaptive behavior, and yet this behavior emerges frequently and appears to be relatively stable. Upon reflection, however, it was clear that we were actually observing the emergence of a higher level of organization.

When an agent dies it is reborn, in place, with a (possibly mutated) version of the genotype of the “best” current individual of the agent’s species, where

⁷ <http://hampshire.edu/lsector/swarmevolve-ex2.mov>



Fig. 2. A view of SWARMEVOLVE 1.0 in which a cloud of agents (the blue species) is hovering around the energy source on the right. Only the central agents are feeding; the others are continually dying and being reborn. As described in the text this can be viewed as a form of emergent collective organization or multicellularity. In this image the agents controlling the energy source on the left are red and most of those between the energy sources and on the floor are purple.

quality is determined from the product of age and energy. This means that the new children that replace the dying individuals on the periphery of the cloud will be near-clones of the feeding individuals within the energy source. Since the cloud generally serves to repel members of other species, the formation of a cloud is a good strategy for keeping control of the energy source. In addition, by remaining sufficiently spread out, the species limits the possibility of collisions between its members (which have energy costs). The high level of genetic redundancy in the cloud is also adaptive insofar as it increases the chances that the genotype will survive after a disruption (which will occur, for example, when the energy source moves).

The entire feeding cloud can therefore be thought of as a genetically coupled collective, or even as a multicellular organism in which the peripheral agents act as defensive organs and the central agents act as digestive and reproductive organs.

4 SWARMEVOLVE 2.0

Although SWARMEVOLVE 2.0 was derived from SWARMEVOLVE 1.0 and is superficially similar in appearance, it is really a fundamentally different system.

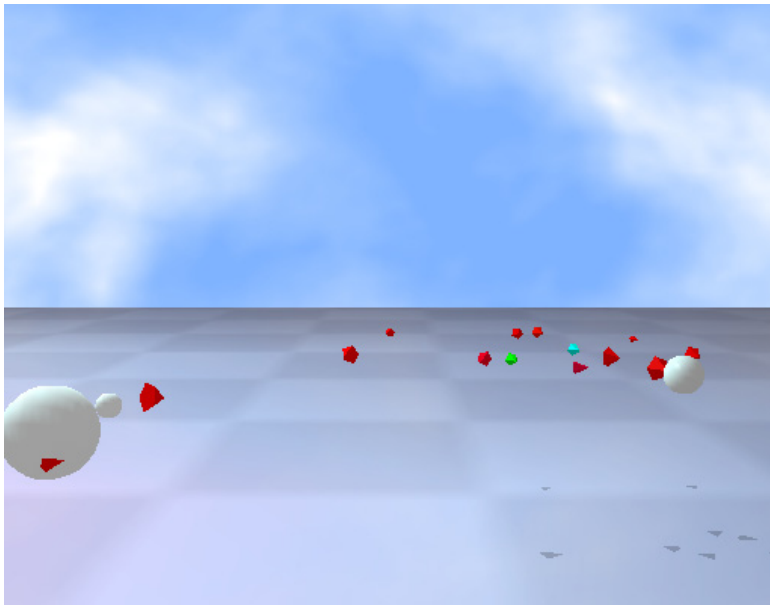


Fig. 3. A view of SWARMEVOLVE 2.0 in which energy sources shrink as they are consumed and agents are “fatter” when they have more energy.

The energy sources in SWARMEVOLVE 2.0 are spheres that are depleted (and shrink) when eaten; they re-grow their energy over time, and their signals (sensed by agents) depend on their energy content and decay over distance according to an inverse square law. Births occur near mothers and dead agents leave corpses that fall to the ground and decompose. A form of energy conservation is maintained, with energy entering the system only through the growth of the energy sources. All agent actions are either energy neutral or energy consuming, and the initial energy allotment of a child is taken from the mother. Agents get “fatter” (the sizes of their bases increase) when they have more energy, although their lengths remain constant so that length still provides the appropriate cues for relative distance judgement in the visual display. A graphical user interface has also been added to facilitate the experimental manipulation of system parameters and monitoring of system behavior.

The most significant change, however, was the elimination of hard-coded species distinctions and the elimination of the hard-coded motion control formula (within which, in SWARMEVOLVE 1.0, only the constants were subject to variation and evolution). In SWARMEVOLVE 2.0 each agent contains a computer program that is executed at each time step. This program produces two values that control the activity of the agent:

1. a vector that determines the agent’s acceleration,
2. a floating-point number that determines the agent’s color.

Agent programs are expressed in Push, a programming language designed by Spector to support the evolution of programs that manipulate multiple data types, including code; the explicit manipulation of code supports the evolution of modules and control structures, while also simplifying the evolution of agents that produce their own offspring rather than relying on the automatic application of hand-coded crossover and mutation operators [11], [12].

Table 1. Push instructions available for use in SWARMEVOLVE 2.0 agent programs

Instruction(s)	Description
DUP, POP, SWAP, REP, =, NOOP, PULL, PULLDUP, CONVERT, CAR, CDR, QUOTE, ATOM, NULL, NTH, +, *, /, >, <, NOT, AND, NAND OR, NOR, DO*, IF	Standard Push instructions (See [11])
VectorX, VectorY, VectorZ, VPlus, VMinus, VTimes, VDivide, VectorLength, Make-Vector	Vector access, construction, and manipulation
RandI, RandF, RandV, RandC	Random number, vector, and code generators
SetServoSetpoint, SetServoGain, Servo	Servo-based persistent memory
Mutate, Crossover	Stochastic list manipulation (parameters from stacks)
Spawn	Produce a child with code from code stack
ToFood	Vector to energy source
FoodIntensity	Energy of energy source
MyAge, MyEnergy, MyHue, MyVelocity, MyLocation, MyProgram	Information about self
ToFriend, FriendAge, FriendEnergy, FriendHue, FriendVelocity, FriendLocation, FriendProgram	Information about closest agent of similar hue
ToOther, OtherAge, OtherEnergy, OtherHue, OtherVelocity, OtherLocation, OtherProgram	Information about closest agent of non-similar hue
FeedFriend, FeedOther	Transfer energy to closest agent of indicated category

The Push instructions available for use in agent programs are shown in Table 1. In addition to the standard Push instructions, operating on integers, floating point numbers, Boolean values, and code expressions, instructions were added for the manipulation of vectors and for SWARMEVOLVE-specific sensors and actions. Note that two sets of instructions are provided for getting information about

other agents in the world, the “friend” instructions and the “other” instructions. Each “friend” instruction operates on the closest agent having a color similar to the acting agent (currently defined as having a hue within 0.1 in a hue scale that ranges from 0.0 to 1.0). Each “other” instruction operates on the closest agent having a color that is *not* similar to the acting agent.⁸ In some cases, in particular when an agent sets its color once and never changes it, the “friend” instructions will be likely to operate on relatives, since presumably these relatives would set their colors similarly. But since agents can change their colors dynamically each time-step, a “friend” is not necessarily a relative and a relative is not necessarily a “friend.” The term “friend” here should be taken with a grain of salt; the friend/other distinction provides a way for agents to distinguish among each other based on color, but they may use this capability in a variety of ways.

SwarmEvolve 2.0 is an “autoconstructive evolution” system, in which agents are responsible for producing their own offspring and arbitrary reproductive mechanisms may evolve [11]. Whenever an agent attempts to produce a child (by executing the Spawn instruction), the top of its code stack is examined. If the expression is empty (which happens rarely once the system has been running for some time) then a newly generated, random program is used for the child. If the expression is not empty then it is used as the child’s program, after a possible mutation. The probability of mutation is also determined by the parent. A random number is chosen from a uniform distribution from zero to the absolute value of the number on top of the Integer stack; if the chosen number is zero then a mutation is performed. The mutation operation is similar to that used in traditional genetic programming: a random sub-expression is replaced with a newly generated random expression. Note that the program access instructions provide the means for agents to produce their children asexually or sexually, potentially using code from many “mates.”

At the beginning of a SWARMEVOLVE 2.0 run most of the agents, which will have been generated randomly, will not have programs that cause them to seek food and produce offspring; they will therefore die rather quickly and the population will plummet. Whenever the population drops below a user-defined threshold the system injects new random agents into the world. With the parameters used here, however, it usually takes only a few hundred time-steps before “reproductive competence” is achieved — at this point the population is self-sustaining as there are a large number of agents capable of reproducing.

SWARMEVOLVE 2.0 is a complex program with many parameters, not all of which can be addressed in the scope of this short paper. However, the source code for the system (including the parameters used in the experiments described below) is available on-line.⁹ Figure 3 shows a typical scene from SWARMEVOLVE 2.0; an animation of a typical action sequence can be found on-line.¹⁰

⁸ If there are no other agents meeting the relevant criterion then each of these instructions operates on the acting agent itself.

⁹ <http://hampshire.edu/lsector/swarmevolve-2.0.tz>

¹⁰ <http://hampshire.edu/lsector/swarmevolve2-ex1.mov>

5 Emergence of Collective Behavior in SWARMEVOLVE 2.0

The last two instructions listed in Table 1, `FeedFriend` and `FeedOther`, provide a means for agents to transfer energy to one another (to share food). Each of these instructions transfers a small increment of energy (0.01 out of a possible total of 1.0), but only under certain conditions which we varied experimentally (see below). Ordinarily, the use of these instructions would seem to be maladaptive, as they decrease the energy of the acting agents. The use of a `Feed` instruction thereby makes the feeding agent both more vulnerable and less likely to produce children.

Might there nonetheless be some circumstances in which it is adaptive for agents to feed one another? We set out to investigate this question by conducting runs of SWARMEVOLVE 2.0 and monitoring the proportion of agents that feed or attempt to feed other agents.¹¹ Because the `Feed` instructions will occasionally occur in randomly generated code and in mutations, we expect every run to produce some number of calls to these instructions. We expect, however, that the proportion of food sharing agents, when averaged over a large number of runs, will reflect the extent to which food sharing is adaptive.

We hypothesized, for example, that dynamic, unstable environments might provide a breeding ground for altruistic feeding behavior. We reasoned as follows, from the perspective of a hypothetical agent in the system: “If the world is stable, and everyone who’s smart enough to find food can reliably get it, then I should get it when I can and keep it to myself. If the world is unstable, however, so that I’ll sometimes miss the food despite my best efforts, then it’d be better for me if everyone shared. Food from others would help to buffer the effects of chance events, and I’d be willing to share food when I have it in order to secure this kind of insurance.” Of course one shouldn’t put too much faith in such “just so stories,” but they can sometimes be a guide for intuitions. In the present case they led us to conduct a simple experiment in which we varied the stability of the energy sources and the sharing conditions in SWARMEVOLVE 2.0 and measured the proportion of food-sharing agents that resulted.

We conducted a total of 1,625 runs under a variety of stability and sharing conditions. We used values of the “stability” parameter ranging from 20 (unstable) to 2,000 (highly stable). The stability parameter governs the frequency with which energy sources begin to drift to new, random locations; the probability that a particular energy source will begin to drift to a new location in any particular time step is $\frac{1}{\text{stability}}$. We collected data on four different sharing conditions. In all of the conditions the potential recipient is the closest agent of similar or dissimilar color, depending on whether the agent is executing the `FeedFriend` or `FeedOther` instruction respectively. In all cases the feeding is conditional on the recipient having less energy than the provider. In “waste” sharing the energy is all lost in the transfer, and the recipient receives nothing; we included this

¹¹ For the analyses presented below we did not distinguish between `FeedFriend` and `FeedOther` executions; we explored the distinction briefly but there were no obvious patterns in the data.

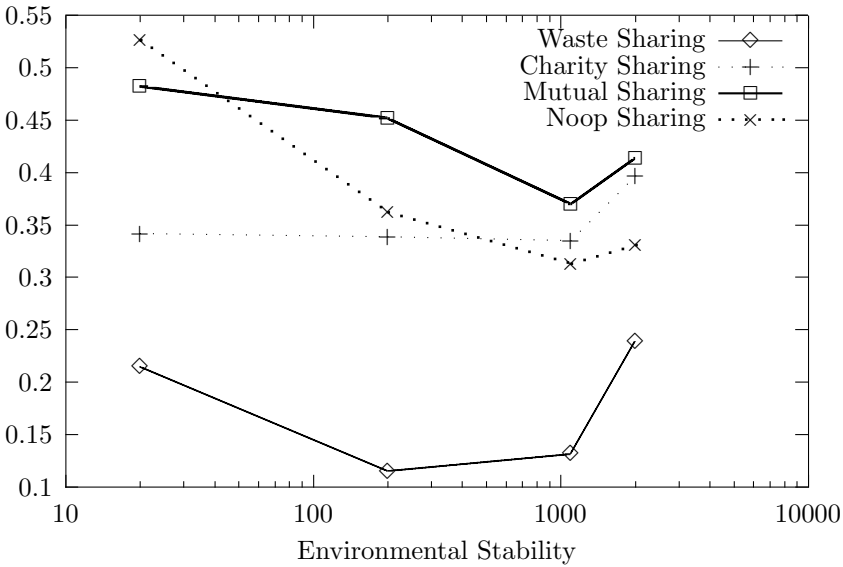


Fig. 4. Proportion of agents that share food (on the y axis) graphed vs. environmental (energy source) stability (on the x axis) for four sharing conditions (see text).

sharing condition as a control. In “charity” sharing the recipient receives all of the energy, regardless of whether or not the recipient itself shares energy. In “mutual” sharing the recipient receives all of the energy, but only if it has itself shared energy at least once in its life. Finally, in “noop” sharing no energy is transferred or lost; this is another control.

We collected data only from runs with at least 5,300 consecutive time-steps of reproductive competence — there were 936 runs meeting this condition. For qualifying runs we then collected data over the last 5,000 of time-steps, divided into 100-time-step “epochs.” At each epoch boundary we took a census, recording the proportion of living agents that have attempted to share energy with another agent on at least one occasion.

Our results are graphed in Figure 4. Our hypothesis that dynamic, unstable environments might provide a breeding ground for altruistic feeding behavior was only partially confirmed; indeed, the most stable environments also appear to be conducive to food sharing. To the extent that the hypothesis is confirmed it is interesting to note that a similar effect, involving the preference for cooperation in unpredictable environments, has been observed in a radically different, game-theoretic context [2].

The “waste” sharing control produced less food sharing than all of the other sharing conditions; this is what one would expect, as “waste” sharing has costs but no possible benefits. The “noop” sharing control, on the other hand, which has no costs and no benefits, produced slightly more sharing than all other conditions at low stability. Note, however, that both “charity” sharing and “mutual”

sharing, which have both costs and potential benefits, produced more sharing than both of the controls at several stability settings. There is substantial variance in the data, and the statistical significance of some of the differences visible in the graph is questionable. In any event we can say that the amount of sharing in the “charity” and “mutual” conditions was, under several stability settings, either greater than or at least not significantly less than the amount of sharing in the “noop” control. This by itself is evidence that altruistic feeding behavior is adaptive in the environment under study. Many of the other differences in the data are clearly significant, and the trends indicate that collective feeding behaviors do arise in some circumstances and not in others. These simulations provide a rich framework for investigating the relations between collective behavior and evolution, which we have only begun to explore.

6 Conclusions and Future Work

The emergence of collective behavior is an intriguing and at times counter-intuitive phenomenon, an understanding of which will have significant impacts on the study of living systems at all levels, from symbiotic microbes to human societies. The work presented in this paper demonstrates that new artificial life technologies provide new tools for the synthetic investigation of these phenomena, complementing the well-established analytic methods of evolutionary biology and behavioral ecology.

In particular we demonstrated the emergence of a simple form of multicellular organization in evolving populations of agents based on a traditional flocking algorithm. We also demonstrated the emergence of altruistic feeding behavior in a system that is considerably less constrained, as the agents are controlled by evolved computer programs. We believe that this latter system provides significant new avenues of study by allowing for agents of arbitrary complexity to evolve within complex, dynamic worlds.

Our own plans for this work in the near future include a systematic exploration of the effects of various parameter changes on the emergence of collective behavior. We are making the source code for SWARMEVOLVE 1.0 and 2.0 freely available in the hopes that others will also contribute to this process; see <http://hampshire.edu/lspector/swarmevolve-1.0.tz> and <http://hampshire.edu/lspector/swarmevolve-2.0.tz>.

Acknowledgments. Raymond Coppinger, Rebecca Neimark, Wallace Feurzeig, Oliver Selfridge, and three anonymous reviewers provided comments that helped to improve this work. This effort was supported by the Defense Advanced Research Projects Agency (DARPA) and Air Force Research Laboratory, Air Force Materiel Command, USAF, under agreement number F30502-00-2-0611, and by NSF grant EIA-0216344.

References

1. Bonner, J.T.: The evolution of complexity by means of natural selection. Princeton University Press, Princeton, NJ (1988)
2. Eriksson, A., Lindgren, K.: Cooperation in an Unpredictable Environment. Proc. Eighth Intl. Conf. on Artificial Life. The MIT Press, Cambridge, MA (2002) 394–399.
3. Klein, J.: breve: a 3D Environment for the Simulation of Decentralized Systems and Artificial Life. Proc. Eighth Intl. Conf. on Artificial Life. The MIT Press, Cambridge, MA (2002) 329–334. <http://www.spiderland.org/breve/breve-klein-alife2002.pdf>
4. Krebs, J.R., Davies, N.B.: An Introduction to Behavioural Ecology. 3rd edn. Blackwell Scientific Publications LTD, Oxford London Edinburgh Boston Melbourne Paris Berlin Vienna (1981)
5. Maynard Smith, J., Szathmáry, E.: The origins of life. Oxford University Press (1999)
6. Pulliam, H.R., Caraci, T.: Living in groups: is there an optimal group size? In J.R. Krebs and N.B. Davies (eds), Behavioral Ecology: An Evolutionary Approach, 2nd edn. Blackwell Scientific Publications LTD, Oxford London Edinburgh Boston Melbourne Paris Berlin Vienna (1984) 122–147
7. Ray, T.S.: Is it Alive or is it GA. Proc. Fourth Intl. Conf. on Genetic Algorithms. Morgan Kaufmann, San Mateo, CA (1991) 527–534
8. Reynolds, C.W.: An Evolved, Vision-Based Behavioral Model of Coordinated Group Motion. In From Animals to Animats 2: Proc. Second Intl. Conf. on Simulation of Adaptive Behavior. The MIT Press, Cambridge, MA (1993) 384–392
9. Reynolds, C.W.: Flocks, Herds, and Schools: A Distributed Behavioral Model. Computer Graphics **24**:4 (1987) 25–34
10. Silby, R.M.: Optimal group size is unstable. Anim. Behav. **31** (1983) 947–948
11. Spector, L., Robinson, A.: Genetic Programming and Autoconstructive Evolution with the Push Programming Language. Genetic Programming and Evolvable Machines **3**:1 (2002) 7–40
12. Spector, L.: Adaptive Populations of Endogenously Diversifying Pushpop Organisms are Reliably Diverse. Proc. Eighth Intl. Conf. on Artificial Life. The MIT Press, Cambridge, MA (2002) 142–145
13. Zaera, N., Cliff, D., Bruton, J.: (Not) Evolving Collective Behaviours in Synthetic Fish. From Animals to Animats 4: Proc. Second Intl. Conf. on Simulation of Adaptive Behavior. The MIT Press, Cambridge, MA (1996)

On Role of Implicit Interaction and Explicit Communications in Emergence of Social Behavior in Continuous Predators-Prey Pursuit Problem

Ivan Tanev¹ and Katsunori Shimohara^{1,2}

¹ ATR Human Information Science Laboratories,
2-2-2 Hikaridai, “Keihanna Science City”, Kyoto 619-0288, Japan

² Graduate School of Informatics, Kyoto University,
Yoshida-Honmachi, Sakyo-ku, Kyoto 606-8501, Japan
{i_tanev, katsu}@atr.co.jp

Abstract. We present the result of our work on use of genetic programming for evolving social behavior of agents situated in inherently cooperative environment. We use predators-prey pursuit problem to verify our hypothesis that relatively complex social behavior may emerge from simple, implicit, locally defined, and therefore – robust and highly-scalable interactions between the predator agents. We propose a proximity perception model for the predator agents where only the relative bearings and the distances to the closest predator agent and to the prey are perceived. The instance of the problem we consider is more realistic than commonly discussed in that the world, the sensory and moving abilities of agents are continuous; and the sensors of agents feature limited range of “visibility”. The results show that surrounding behavior, evolved using proposed strongly typed genetic programming with exception handling (STGPE) emerges from local, implicit and proximity-defined interactions between the predator agents in both cases when multi-agents systems comprises (i) partially inferior predator agents (with inferior moving abilities and superior sensory abilities) and with (ii) completely inferior predator agents. In the latter case the introduction of short-term memory and explicit communication contributes to the improvement of performance of STGPE.

1 Introduction

Over the past few years, multi-agent systems (MAS) have become more and more important in many aspects of computer science such as distributed artificial intelligence, distributed computing systems, robotics, artificial life, etc. MAS introduce the issue of collective intelligence and of the emergence of behavior through interactions between the agents. An agent is a virtual entity that can act, perceive the proximity of its environment and communicate with others; it is autonomous and has abilities to achieve its objectives. MAS contain a world (environment), entities (agents), relations between the entities, a way the world is perceived by the entities, a set of operations that can be performed by the entities and the changes of the world as a result of these actions. Currently, the main application areas of MAS are problem solving, simula

tion, collective robotics, software engineering, and construction of synthetic worlds [4]. Considering the latter application area and focusing on the autonomy of agents and the interactions that link them together [14], the following important issues can be raised: What is minimum amount of perception information needed to agents in order to perceive the world? How can agents cooperate? What are the methods, and what are the lower bounds of communications, required for them to coordinate their actions? What is the architecture they should feature so that they can achieve their goals? What approaches can be applied to automatically construct the agents' functionality, with the quality of such a design being competitive to the design handcrafted by human? These issues are of special interest, since the aim is to create MAS which is scalable, robust, flexible, and the able to automatically adapt to changes. These features of MAS are believed to be particularly important in real world applications where the approaches to construct synthetic worlds can be viewed as a practical methods, a techniques towards creating complex "situational aware" multi-computer, multi-vehicle, or multi-robot systems based on the concepts of agents, communication, cooperation and coordination of actions.

Within considered context, the *objective* of our research is an automatic design of autonomous agents which situated in inherently cooperative environment are capable of accomplishing complex tasks through interaction. We adhere to the methodological holism based on the belief that any complex system or society (Heraclitus, Aristotle, Hegel, and more recently – [12][13]), and multi-agent society in particular [7] is more than the sum of its individual entities, more than the sum of the parts that compose it. The social behavior, needed to accomplish the complex task might emerge in MAS from relatively simply defined interactions between the agents. We are particularly interested in the ultimate case of such simplicity – local, implicit, proximity defined, and therefore, robust, flexible and highly scalable interactions between the agents, situated in more realistic (than commonly considered), inherently cooperative environments. This document is intended to highlight the issues of applying genetic programming for investigating the sufficiency of implicit interaction and the role of explicit communication in emergence of social behavior in MAS.

The remaining of the document is organized as follows. Section 2 introduces the task which we use to test our hypotheses – an instance of the general, well-defined yet difficult to solve predator-prey pursuit problem. The same section addresses the issue of developing the software architecture of the agents. Section 3 elaborates the strongly typed genetic programming with exceptions handling (STGPE), proposed as an algorithmic paradigm used to evolve the functionality of agents. Empirical results are presented in Section 4 and conclusion is drawn in Section 5.

2 The Problem and the Agents Architecture

2.1 Instance of Predator Prey Pursuit Problem

The general, well-defined and well-studied yet difficult to solve predator-prey pursuit problem [2] is used to verify our hypothesis that relatively complex social behavior

might emerge from simple, local, implicit, proximity-defined, and therefore – robust and highly-scalable interactions between the predator agents. The problem comprises four predator agents whose goals are to capture a prey by surrounding it on all sides in a world. In our work we consider an instance of the problem, which is more realistic than commonly considered in the previous work [5][6][9]. The world is a simulated two-dimensional continuous torus 1600mm x 1000mm. The moving abilities of four predator agents are continuous too – the predators can turn left and right to any angle from their current heading and can run with speed equal to 0, 0.25, 0.5, 0.75 and 1.0 of maximum speed. In addition, we introduce an proximity perception model for predator agents in that they can see the prey and *only the closest predator agent*, and only when they are within the *limited range* of visibility of their simulated (covering an area of 360 degrees) sensors. The prey employs random wandering if there is no predator in sight and a priori handcrafted optimal escaping strategy as soon as predator(s) become “visible”. The maximum speed of prey is higher than the maximum speed of predator (i.e. predator-agents feature inferior moving abilities). In order to allow for predators to stalk and collectively approach the prey, in the first of the two considered cases the range of visibility of predators is more than the range of visibility of the prey (i.e. superior are only the sensory abilities of predators, therefore they can be considered *partially inferior*). In the second case the range of visibility of predators is equal to the range of visibility of the prey (i.e. *completely inferior* predator agents). We consider these two cases in order to create an inherently cooperative environment in that the mission of predators is nearly impossible unless they collaborate with each other. We are not interested in cases when predators are superior in their moving abilities, since the capturing of the prey in this case seems trivial, can be accomplished by single agent, and therefore does not require collective behavior from MAS. Analogically, the situation comprising completely inferior agents who besides being slower feature more myopic (rather than equal) sensory abilities than the prey is intractable for the conditions give above, and therefore is beyond our current consideration.

2.2 Architecture of the Agents

We adopted the subsumption architecture [3] of the agents comprising of functional modules distributed in three levels, corresponding to the three different aspects (“levels of competence”) of agent’s behavior: wandering/exploring, greedy chase and social behavior - surrounding (Figure 1a). Given that we focus our attention on evolving the top-level, highest priority module – surrounding the prey (assuming that the rest two modules are handcrafted), our objective of automatic design of autonomous agents via simulated evolution, which we declared earlier, can be rephrased as evolving the surrounding module in subsumption architecture of the agents.

In order to coordinate the functionalities of each of architectural modules we introduce the notion of agent’s state. At every instant, the agent can be in one of the three states, corresponding to the module which is currently governing the agent’s behavior: *surrounding*, *greedy chase* and *wandering/exploring*. The agent is in *surrounding* state if and only if there is a match (response) for the currently perceived proximity of the world (stimuli) in the functionality of evolved surrounding module. Being in *sur-*

rounding state, the agent is fully controllable by the functionality of the surrounding module, while functionalities of hunting and wandering/exploring modules are inhibited. If there is no match of the perceived proximity of the world in the functionality of evolved surrounding module, the agents state switches to *greedy chase* if prey is in sight or to *wandering/exploring* state otherwise (Figure 1b).

We would like to emphasize that the proposed implicit interstate-transition scheme is fully controllable by evolved functionality of the surrounding module, which allows to simultaneously evolve (i) the capability of agents to resolve social dilemmas, determined by the way *social behavior* overrides *greedy chase* when prey is in sight, and (ii) the capability to resolve the exploration-exploitation dilemma, determined by the ability of *social behavior* to override *wandering/exploring* when prey is invisible.

3 Algorithmic Paradigm Employed to Evolve Predator Agents

3.1 Strongly-Typed Genetic Programming with Exception Handling

Limiting the Search Space of Genetic Programming. We consider a set of stimulus-response rules as a natural way to model the reactive behavior of predator agents [7] which in general can be evolved using artificial neural networks, genetic algorithms, and genetic programming (GP). GP is a domain-independent problem solving approach in which a population of computer programs (individuals) is evolved to solve problems [8]. The simulated evolution in GP is based on the Darwinian principle of reproduction and survival of the fittest. In GP genetic programs (individuals) can be represented as parsing trees whose nodes are functions, variables or constants. The nodes that have sub-trees are non-terminals - they represent functions where the sub-trees represent the arguments to function of that node. Variables and constants are terminals - they take no arguments and they always are leaves in the parsing tree. The set of terminals for evolving agent's behavior includes the perceptions (stimuli), and the actions (response) which the agent is able to perform. The set of functions comprises the arithmetical and logical operators, and the IF-THEN function, establishing the relationship between certain stimulus and corresponding response(s). Without touching the details of representation of genetic programs, which will be elaborated later in Section 3.2, a human readable form of sample stimulus-response rule is shown in Figure 2. It expresses a reactive behavior of turning to the bearing of the peer agent (*Peer_a*) plus 10 (degrees) as a result of stimulus of its own speed being less than 20 (mm/s).

The strength of GP to automatically evolve a set of stimulus-response rules of arbitrary complexity without the need to a priori specify the extent of such complexity might imply an enormous computational effort caused by the need to discover a huge search space while looking for potential solution to the problem. Agreeing with [13] that for huge and multidimensional search spaces the introduction of "pruning algorithms" is a critical step towards efficient solution search, we impose a restriction on the syntax of evolved genetic programs based on some a priori known semantics. The

approach is known as strongly typed genetic programming and its advantage over canonical GP in achieving better computational effort is well proven [11].

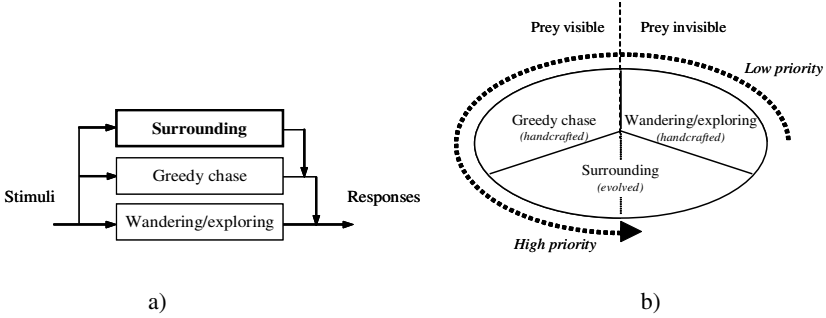


Fig. 1. Subsumption architecture of the agents: functional structure (a) and states (b)

```
IF (Speed<20) THEN Turn(Peer_a+10)
```

Fig. 2. Sample stimulus-response rule

Considering the sample rule, shown in Figure 2 it is noticeable that both the functions and their operands are associated with *data types* such as *speed* (e.g. Speed, 20), *angle of visibility (bearing)* (Peer_a, 10), and *Boolean* (Speed<20). An eventual arbitrary creation or modification of genetic program semantically would make little sense: indeed, it is unfeasible to maintain Boolean expressions comparing operands of different data types at least because they have different physical units. Moreover, since we introduce sensor's range limits, there is a clear possibility of maintaining introns in genetic programs when, for example Boolean expressions include comparison of perception variable of certain data type with constant value beyond the limits of the data type of that variable (e.g. Peer_d>1000, in case that sensor range is only 400). Analogically, the semantics of action Turn() implies a parameter of data type *angle*. And allowing just addition and subtraction as arithmetical operations implies that all the operands involved in the expression which defines the turning parameter should have the same data type *angle*. Addressing the mentioned concerns, the grammar of STGPE establishes generic data types of *visible angle*, *distance*, *speed*, and *Boolean* with the corresponding allowed ranges of the values for their respective instances (variables and ephemeral constants). In addition, it stipulates the data type of the results of arithmetical and logical expressions, and the allowed data type of operands (perception variables and ephemeral constants) involved in these expressions.

We would like to emphasize that proposed approach is not based on domain-specific knowledge, and therefore STGPE can not be considered as a “stronger” approach compromising the domain-neutrality of the very GP paradigm itself. The limitations imposed to the syntax of genetic programs are solely based (i) on the natural presumption that the predator agents are fully aware of their physically reasonable limits of his perception- and moving abilities; and (ii) on the common rule in strongly-

typed 3G algorithmic languages that all the operands in addition, subtraction and comparison operations should have the same data types. In no way these limitations incorporate a priori obtained knowledge, specific for the domain or for the world where the agents are situated.

Exception Handling. The notion of exception handling is introduced in a way, much similar to the 3G algorithmic languages. In our approach, an *exception* is an event, raised when a runtime error occurs in an evaluation of the Boolean expression in the conditional part of IF-THEN rule. Due to the limited range of simulated sensors of predator agents, such an error would happen when the Boolean condition involving perception variable(s) related to perceiving another entity in the world (e.g. closest predator agent and/or the prey) can not be evaluated because the corresponding entity is currently “invisible”. In addition to IF-THEN we introduce IF-THEN-NA (IF-THEN-“not available”) type of stimulus-response rule with exception handling capabilities. The human readable syntax and the corresponding semantics of sample stimulus response rule with exception are shown in Figure 3.

3.2 Main Attributes of STGPE

Function and Terminal Sets. Function and terminal sets of adopted STGPE are summarized in Table 1. Notice the local, proximity defined sensory abilities of agents.

Representation of Genetic Programs. Inspired by flexibility and recently emerged widespread adoption of document object model (DOM) and extensible markup language (XML), we represent genetic programs as a DOM-parsing trees featuring corresponding flat XML text. Our additional motivation stems from the fact that despite of the recently reported use of DOM/XML for representing computer architectures, source codes, and agents’ communication languages we are not aware about any attempts to employ XML technology for representing evolvable structures such as genetic programs in generic, standard, and portable way. Our approach implies performing genetic operations on DOM-parsing tree using off-the shelf, platform- and language neutral DOM-parsers, and using XML-text representation (rather than S-expression) as a flat format, feasible for migration of genetic programs among the computational nodes in eventual distributed implementation of STGPE. The fragment of XML representation of the above discussed sample stimulus-response rule (refer to Figure 3) is shown in Figure 4. The benefits of using DOM/XML-based representations of genetic programs, as documented in [15] can be briefly summarized as follows: (i) XML tags offer a generic support for maintaining data types in STGPE; (ii) W3C-standard XML schema offers generic way for representing the grammar of STGPE; (iii) using standard built-in API of DOM-parsers for maintaining and manipulating genetic programs; (iv) OS neutrality of parsers; (v) algorithmic language neutrality of DOM-parsers, and (vi) inherent Web-compliance of eventual parallel distributed implementation of STGPE.

Genetic Operations. Binary tournament selection is employed – a robust, commonly used selection mechanism, which has proved to be efficient and simple to code.

<pre> TRY IF (Peer_d<20) THEN Turn(Peer_a+10) EXCEPT Turn(10); </pre>	<pre> IF (Peer is Visible) THEN BEGIN IF (Peer_d<20) THEN Turn(Peer_a+10); END ELSE Turn(10); // invisible predator agent </pre>
a)	b)

Fig. 3. Syntax (a) and semantics (b) of sample stimulus-response rule with exception handling

Table 1. Function Set and Terminal Set of STGPE

Category		Designation	Remarks
Function set		IF-THEN, IF-THEN-NA LE, GE, WI, EQ, NE, +, -	IF-THEN without/with exception handling $\geq$ i in $\neq$
Terminal set	Sensory abilities	Prey_d; Peer_d Prey_a; Peer_a PreyVisible; PeerVisible	Distance to the prey and to the closest agent, <i>mm</i> . Bearing of the prey and of the closest agent, <i>degrees</i> True if prey / agent is “visible”, false otherwise
	State variable	Speed	Speed of the agent, <i>mm/s</i>
	Ephemeral constants	Integer	
	Moving abilities	n α Stop, Go_1.0 Go_0.25, Go_0.5, Go_0.75	n i α g α 0: i) Sets speed to 0, Sets speed to maximum Sets speed to 25%, 50%, 75% of maximum

<pre> <IF-THEN-NA> <COND-THEN-NA><COND_TDistance> <VAR_TDistance>Peer_d</VAR_TDistance > <OPER_TDistance>LE</OPER_TDistance > <CONST_TDistance>20</CONST_TDistance > </COND_TDistance ></COND-THEN-NA> <THEN> ... </THEN> <NA> ... </NA> </IF-THEN-NA> </pre>

Fig. 4. Fragment of XML representation of sample stimulus-response rule

Crossover operation is defined in a strongly typed way in that only the nodes (and corresponding subtrees) of the same data type (i.e. labeled with the same tag) from parents can be swapped. The sub-tree mutation is also allowed in strongly typed way in that a random node in genetic program is replaced by syntactically correct sub-tree. The routine refers to the type of node it is going to currently alter and applies the randomly chosen rule from the set of applicable rules as defined in the grammar of STGPE. The transposition mutation also operates on single genetic program by swapping two random nodes having the same data type.

Breeding Strategy. We adopted a homogeneous breeding strategy in which the performance of single genetic program, cloned to all the agents is evaluated. Anticipating that the symmetrical nature of the world, populated with identical predator agents is

unlikely to promote any specialization in the behavior of agents, we consider the features of such a homogeneous multi-agent society as (i) adequate to the world and (ii) consistent with our previously declared intention to create robust and well scalable multi-agent system.

Fitness Function. In order to obtain more general solutions to the problem the fitness of the genetic program is evaluated as average of the fitness measured over 10 different initial situations. However, based on the empirically proven data that on the initial stages of evolution agents are hardly able to successfully resolve more than few (out of 10) initial situations in order to enhance the computational performance of STGPE we applied the approach of noisy evaluation of the fitness function [10]. The amount of initial situations used to evaluate the genetic programs in population gradually increases as population evolves. Starting from 4 for the first generation of each run, the amount of situations is revised on completion of each generation and it is set to exceed by number of 2 the amount of situations, successfully solved by the best-of-generation genetic program. Given that with addition of another initial situation(s) they have to resolve, the agents would perform either better or, most probably worse, the fitness of the best-of-current generation could be occasionally somewhat worse than fitness of best genetic program from previous generation. Therefore, it is reasonable to expect non-monotonous fitness convergence characteristics of STGPE.

The fitness F measured for the trial starting with particular initial situation is evaluated as a length of the radius vector of the derived agents' behavior in the virtual energy-distance-time space as:

$$F = \sqrt{dE_A^2 + D_A^2 + T^2} \quad (1)$$

where dE_A is the average energy loss during the trial, D_A is the average distance to the prey by the end of the trial, and T is the elapsed time of the trial. The quantities dE_A and D_A are averaged over the all predator agents. The energy loss estimation dE for each of predator agents takes into account both the basal metabolic rate, equal to 0.05 units per second, and the energy loss for moving activities equal to 0.01 units per mm of path traversed during the trial. The trial is limited to 300s of "real" time or to the instance when prey is captured; and with sampling rate of 500ms it is simulated with up to 600 time steps. Smaller values of fitness function correspond to better performing predator agents.

4 Empirical Results

4.1 Parameter Values of STGPE

The parameters of STGPE used in our experiments are as follows: the population size is 400 genetic programs, the selection ratio is 0.1 (including 0.01 elitism), and the mutation ratio is 0.02, equally divided between sub-tree mutation and transposition. The termination criterion is defined as a disjunction of the following three termination conditions: (i) fitness of the best genetic program in less than 300 and the amount of

initial situations in which the prey is captured (successful situations) equals 10 (out of 10), (ii) the amount of elapsed generations is more than a 100, and (iii) the amount of recent generations without fitness improvement is more than 16.

4.2 Partially Inferior Predator Agents

In the first case a superior sensory abilities of predators (range of visibility 400mm vs. 200mm for prey) and inferior moving abilities are considered (20mm/s vs. 24mm/s). The computational effort (amount of genetic programs needed to be evaluated in order to obtain the solution with specified probability, e.g. 0.95), is obtained from the probability of success $p(t)$ by each of 20 independent runs in a way as suggested in [8]. The result, shown in Figure 5a indicates that $p(t)=0.95$ by generation 80 which yields a computational effort of about 32,000 genetic programs. Typical fitness convergence characteristic is shown in Figure 5b.

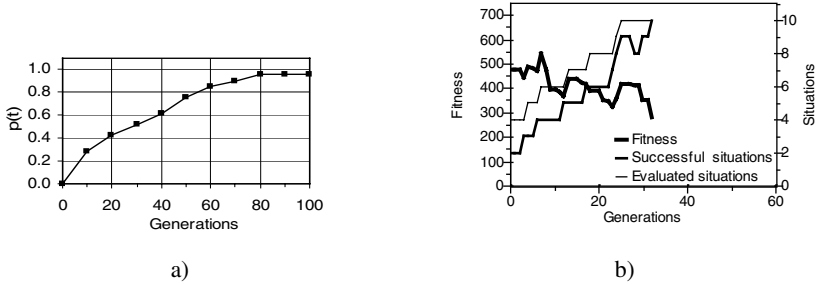


Fig. 5. Probability of success (a) and typical fitness convergence characteristic (b)

Human-readable representation of sample best-of-run genetic program is shown in Figure 6, and the traces of the entities in the world for one of the 10 initial situations is shown in Figure 7. The prey, originally situated in the center of the world, is captured by time step 140. The emergence of following behavioral traits of predator agents are noticeable: (i) switch from greedy chase into surrounding approach of agents #3 (time step 65, on right part of the world) and agent #2 (time step 120, top left) as soon as other agents appear in sight; (ii) zigzag move by agent #0 which results a lower chasing speed indicating “intention” to trap the prey (after time step 100, far right and far left) and (iii) surrounding approach demonstrated by agents #1 and #3 during the final stages of the trial (top left).

4.3 Completely Inferior Predator Agents

In this case the same range of visibility of predators (400mm vs. 400mm respectively) and inferior moving abilities are considered (20mm/s vs. 24mm/s). These conditions definitely render the task more difficult for predator agents. As empirical results indicate that for the same values of GP-parameters (as sated in Section 4.1) a probability

of success of 0.95 is hardly achievable. In order to illustrate the very ability of STGPE to discover the solution in these conditions we present plotted values of fitness of the best-of-run genetic program and the amount of successfully resolved situations for 20 independent runs. The results are shown in Figure 8a.

```

Program Main;
type TDistance = 0..400;
TVisAngle = -180..180;
TSpeed = 0..22;

var Peer_d, Prey_d : TDistance;
Peer_a, Prey_a : TVisAngle;
Speed : TSpeed;
PreyVisible, PeerVisible : Boolean;

Procedure GP;
begin
  try if (Prey_a >= -26)
    then try if (Prey_a within -5)
      then begin
        if (not PeerVisible) then Turn(Prey_a);
        try if (Prey_a <= -139)
          then begin
            Null;
            Go_0.25;
          end;
        except Turn(~24-Peer_a+7-24);
      end;
    except Null;
  except try if (Peer_d <= 136) then Null; except Turn(Prey_a);
end;
begin // main program
  GP;
end.

```

Fig. 6. Human-readable representation of sample best-of-run genetic program

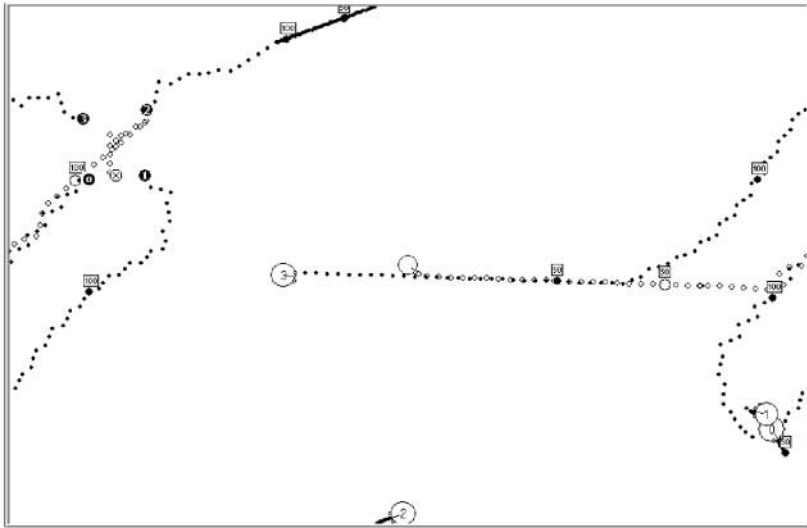


Fig. 7. Traces of the entities with agents governed by the genetic program shown in Figure 6. The prey is captured in 140 simulated time steps (top left). Larger white and small black circles denote the predator agents in their initial and final position respectively. The small white circle indicates the prey, initially situated in the center of the world. The numbers in rectangles show the timestamp information.

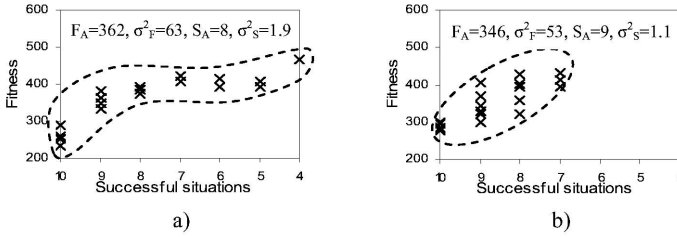


Fig. 8. Fitness and amount of successful situation of the best-of-run genetic program in 20 independent runs when agents are implicitly interacting (a) and when agents are employing short term memory with explicit communication.

As figure illustrates, only 25% of runs have been successfully completed (i.e. terminated by criteria (i) as described in Section 4.1) and the average and the standard deviation of fitness and successful situations are $F_A=362$, $\sigma_F^2=63$, $S_A=8$ and $\sigma_S^2=1.9$. The results with introduced short term (working) memory [1], storing the direction where in which prey has been recently seen and explicit communication allowing the exchange the currently or recently seen direction to the prey (i.e. predator agents still remain *mechanically* inferior) are shown in Figure 8b. Although the solution can be evolved by STGPE even with implicit interactions between the predator agents, introducing short term memory and explicit communication improves the performance of simulated evolution: 35% of 20 runs have been successfully completed with more favorable statistical results of $F_A=346$, $\sigma_F^2=53$, $S_A=9$ and $\sigma_S^2=1.1$.

5 Conclusion

We presented the result of our work on use of genetic programming for evolving social behavior of agents situated in inherently cooperative environment. We use predators-prey pursuit problem to verify our hypothesis that relatively complex social behavior may emerge from simple, implicit, locally defined, and therefore – robust and highly-scalable interactions between the predator agents. We proposed a proximity perception model for the predator agents where only the relative bearings and the distances to the closest predator agent and to the prey are perceived. The instance of the problem we consider is more realistic than commonly discussed in that the world, the sensory and moving abilities of agents are continuous; and the sensors of agents feature limited range of “visibility”. Adopted subsumption architecture and developed implicit inter-state transition model allow for simultaneous evolution of the capabilities of predator agents to resolve both the social dilemma and the dilemma between exploration and exploitation. The empirical results show that surrounding behavior, evolved using proposed strongly typed genetic programming with exception handling (STGPE) emerges from local, implicit and proximity-defined interactions between the predator agents in both cases when multi-agents systems comprises (i) partially inferior predator agents (with inferior moving abilities and superior sensory abilities) and with (ii) completely inferior predator agents. In the latter case the introduction of

short-term memory and explicit communication contributes to the improvement of performance of STGPE.

Acknowledgements. This research was conducted as part of “Research on Human Communication” with funding from the Telecommunications Advancement Organization of Japan.

References

1. Barborica, A., Ferrera, V.P., Estimating Invisible Target Speed from Neuronal Activity in Monkey Frontal Eye Field, *Nature Neuroscience* Vol.6, No.1 (2003) 66–74
2. Benda, M., Jagannathan, B., Dodhiawala, R.: On Optimal Cooperation of Knowledge Sources. Technical Report BCS-G2010-28, Boeing AI Center, Boeing Computer Services, Bellevue, WA (1986)
3. Brooks, R.A.: A Robust Layered Control System for a Mobile Robot. *IEEE Journal of Robotics and Automation*, Vol.2, No.1 (1986) 14–23
4. Ferber, J.: *Multi-Agent Systems: An Introduction to Distributed Artificial Intelligence*, Harlow: Addison Wesley Longman (1999)
5. Haynes, T., Sen, S.: *Evolving Behavioral Strategies in Predators and Prey* (1996)
6. Haynes, T., Wainwright, R., Sen, S., Schoenefeld, D.: Strongly Typed Genetic Programming in Evolving Cooperation Strategies (1997)
7. Holland, J.H.: *Emergence: From Chaos to Order*, Cambridge, Perseus Books (1999)
8. Koza, J.R.: *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, Cambridge, MA, MIT Press (1992)
9. Luke, S., Spector, L.: *Evolving Teamwork and Coordination with Genetic Programming*
10. Miller, B.,L., Goldberg, D., E.,: *Genetic Algorithms, Tournament Selection, and the Effects of Noise*, Illigal Report No. 95006, University of Illinois (1995).
11. Montana, D.: Strongly Typed Genetic Programming, *Evolutionary Computation*, Vol.3, No.2 (1995) 199–230
12. Morgan, C.: *Emergent Evolution*, New York (1923)
13. Morowitz, H.,J.: *The Emergence of Everything: How the World Became Complex*, Oxford University Press, New York (2002)
14. Parunak, H. Van D., Brueckner, S., Fleischer, M., Odell, J.: Co-X: Defining what Agents Do Together, *Proceedings of the AAMAS 2002 Workshop on Teamwork and Coalition Formation*, Onn Shehory, Thomas R. Ioeberger, Julita Vassileva, John Yen, eds., Bologna, Italy, (2002)
15. Tanev, I.: DOM/XML-Based Portable Genetic Representation of Morphology, Behavior and Communication Abilities of Evolvable Agents, *Proceedings of the 8th International Symposium on Artificial Life and Robotics (AROB’03)*, Beppu, Japan (2003), 185–188

Demonstrating the Evolution of Complex Genetic Representations: An Evolution of Artificial Plants

Marc Toussaint

Institut für Neuroinformatik, Chair for Theoretical Biology
Ruhr-Universität Bochum ND-04, 44780 Bochum, Germany
toussaint@neuroinformatik.rub.de

Abstract. A common idea is that complex evolutionary adaptation is enabled by complex genetic representations of phenotypic traits. This paper demonstrates how, according to a recently developed theory, genetic representations can self-adapt in favor of evolvability, i.e., the chance of adaptive mutations. The key for the adaptability of genetic representations is neutrality inherent in non-trivial genotype-phenotype mappings and neutral mutations that allow for transitions between genetic representations of the same phenotype. We model an evolution of artificial plants, encoded by grammar-like genotypes, to demonstrate this theory.

1 Introduction

In ordinary evolutionary systems, including natural evolution and standard Genetic Algorithms (GAs), the search strategy is determined by uncorrelated mutations and recombinations on the level of genes. In view of this trial-and-error search strategy, it might seem surprising that natural evolution was that effective in finding highly *structured*, complex solutions. One might have expected that the search strategy needs to be more sophisticated, more structured and adapted to the “objective function” to accomplish this efficiency.

An example for sophisticated, adapted, and structured search strategies are recent developments in evolutionary computation, which can be collected under the name of *Estimation-of-Distribution Algorithms* (EDAs, see [10] for a review). These algorithms structure search by learning probabilistic models of the distribution of good solutions.

Now, it hardly seems that ordinary evolutionary systems like natural evolution or GAs learn “models about the distribution of good solutions” and adapt their mutational exploration accordingly. However, it is possible for ordinary evolutionary systems to adapt their search strategy and natural evolution certainly exploits this possibility. The key to understand this possibility is to consider a non-trivial relation between genotype and phenotype, between the genes and the phenes (phenotypic traits that are fitness-relevant). In this case it is possible that the same phenotype can be represented by different genotypes. In other words, evolution could principally adapt the way it represents solutions.

Adapting the genetic representation of phenes while keeping mutation operators fixed is in some sense dual to adapting the mutation operators while keeping

the representation of phenes fixed. Especially in the biology literature it is often argued that this is the way natural evolution adapts its search strategy: The way genes encode phenotypic traits is surely not an incident but the outcome of a long adaptative process in favor of evolvability [15]. Much research is spend to understand these principles of gene interactions (like epistasis or canalization) [1, 16, 17, 4, 5].

But there would be another open question. Do genetic representations really adapt such that the search strategy becomes “better”? Is the induced adaptation of the search strategy similar to the adaptation in the case of EDAs, such that indeed a model of the distribution of good solutions is indirectly learned by adapting the representations of solutions? Recently, a theory on the adaptation of phenotypic search distributions (*exploration distributions*) in the case of non-trivial genotype-phenotype mappings was proposed [14]. Since the present paper aims to illustrate this process and thereby to open a more intuitive access to this theory, we want, at this place, briefly review the main results.

A non-trivial genotype-phenotype mapping is such that (1) the same phenotype is encoded by more than one genotype, and (2), at least for some of these phenotypes, the induced phenotypic variability depends on the genotype it is encoded with. In this case, every genotype g may be written as a tuple (x, σ) , where x is its phenotype and σ generally represents any kinds of *neutral traits* of that genotype (strategy parameters are only a special case). Different σ for the same x mean different genetic representations of the same phenotype. In [14] it is shown that a σ is one-to-one identifiable with a mutation distribution (i.e., search distribution) “around” g . Hence, the tuple (x, σ) actually lives in the product space of phenotype space and the space of distributions.

Now, the main approach is the following: Instead of investigating the evolution of genotypes g in a given evolutionary process, one can also analyze how x and σ evolve in that same process. The most interesting result is that the equation that describes the evolution of σ ’s turns out to be itself an equation similar to evolutionary dynamics. The equation allows to associate a quality measure to each σ and says that σ ’s evolve as to increase this quality measure just as normal evolutionary processes evolve as to increase the fitness of genotypes. This quality measure of σ ’s can be interpreted as a measure of “how good the mutational phenotypic variability (the phenotypic search distribution) matches the distribution of good organisms”—just as for EDAs. The results are summarized in following corollary:

Corollary 1 ([14]). *The evolution of genetic representations, or equivalently, of exploration distributions (σ -evolution) naturally has a selection pressure towards*

- *minimizing the Kullback-Leibler divergence between exploration and the exponential fitness distribution, and*
- *minimizing the entropy of exploration.*

Here, the exponential fitness distribution describes the distribution of good solutions and the Kullback-Leibler divergence is a distance measure between two distributions and thus measures the match between the search strategy (as given by the exploration distribution) and this exponential fitness distribution.

This paper will present a first case study for the evolution of genetic representations by modeling an evolution of artificial plants. We chose this case study for

several reasons: First, the genotype-phenotype mapping that we will introduce shortly is highly non-trivial in the strict sense we defined it for σ -evolution, and it is interpretable from an (abstract) biological point of view. Further, this kind of encoding and the possibility to visually display the phenotypes allows to really grasp the genetic encodings, how completely different genetic representations of the same phenotype are possible, that complex features like gene interactions and modularity of the genetic representations can emerge.

The next section introduces the genetic encoding and the mutability we assume on this encoding. The major novelty here are 2nd-type mutations that allow for neutral transitions between equivalent genetic representations within the grammar-like encoding. Section 3 then explains the experimental setup and presents the experiments that demonstrate the evolution of complex genetic representations. Conclusions follow thereafter.

2 A Genotype Model and 2nd-Type Mutations

The genotype-phenotype mapping we will assume is a variant of the L-systems proposed by [11] to encode plant-like structures. Let us introduce this encoding from another point of view that puts more emphasis on the principles of ontogenesis and gene interactions.

Let us consider the development of an organism as an interaction of its state Ψ (describing, e.g., its cellular configuration) and a genetic system Π , neglecting environmental interactions. Development begins with an initial organism in state $\Psi^{(0)}$, the “egg cell”, which is also inherited and which we model as part of the genotype. Then, by interaction with the genetic system, the organism develops through time, $\Psi^{(1)}, \Psi^{(2)}, \dots \in P$, where P is the space of all possible organism states. Hence, the genetic system may be formalized as an operator $\Pi : P \rightarrow P$ modifying organism states such that $\Psi^{(t)} = \Pi^t \Psi^{(0)}$. We make a specific assumption about this operator: we assume that Π comprises a whole sequence of operators, $\Pi = \langle \pi_1, \pi_2, \dots, \pi_r \rangle$, each $\pi_i : P \rightarrow P$. A single operator π_i (also called production rule) is meant to represent a transcription module, i.e., a single gene or an operon. Based on these ideas we define the general concept of a genotype and a genotype-phenotype mapping for our model:

A genotype consists of an initial organism $\Psi^{(0)} \in P$ and a sequence $\Pi = \langle \pi_1, \pi_2, \dots, \pi_r \rangle$, $\pi_i : P \rightarrow P$ of operators. A genotype-phenotype mapping ϕ develops the final phenotype $\Psi^{(T)}$ by recursively applying all operators on $\Psi^{(0)}$.

This definition is somewhat incomplete because it does not explain the stopping time T of development and in which order operators are applied. We keep to the simplest options: we apply the operators in sequential order and will fix T to some chosen value.

For the experiments we need to define how we represent an organism state Ψ and how operators are applied. We represent an organism by a sequence of symbols $\langle \psi_1, \psi_2, \dots \rangle$, $\psi_i \in \mathcal{A}$. Each symbol may be interpreted, e.g., as the state of a cell; we choose the sequence representation as the simplest spatially organized assembly of such states. Operators are represented as replacement rules $\langle a_0; a_1, a_2, \dots \rangle$, $a_i \in \mathcal{A}$, that apply on the organism by replacing all occurrences of a symbol a_0 by the sequence $\langle a_1, a_2, \dots \rangle$. If the sequence $\langle a_1, a_2, \dots \rangle$ has length

greater than 1, the organism is growing; if it has length 0, the organism shrinks. Calling a_0 promoter and $\langle a_1, a_2, \dots \rangle$ the structural genes gives the analogy to operons in natural genetic systems. For example, if the initial organism is given by $\Psi^{(0)} = \langle a \rangle$ and the genetic system is $\Pi = \langle \langle a:ab \rangle, \langle a:cd \rangle, \langle b:adc \rangle \rangle$, then the organism grows as: $\Psi^{(0)} = \langle a \rangle$, $\Psi^{(1)} = \langle cdadc \rangle$, $\Psi^{(2)} = \langle cdcdadcdc \rangle$, etc.

The general idea is that these operators are basic mechanisms which introduce correlating effects between phenotypic traits. [13] already claimed that the essence of the operon is to introduce correlations between formerly independent genes in order to adopt the *functional* dependence between the genes and their phenotypic effects and thereby increase the probability of successful variations.

The proposed model is very similar to the models by [8], [3], and [9], who use grammar-encodings to represent neural networks. It is also comparable to new approaches to evolve complex structures by means of so-called *symbiotic composition* [6, 7, 18].

The crucial novelty in our model are *2nd-type mutations*. These allow for genetic variations that explore the neutral sets which are typical for any grammar-like encoding. Without these neutral variations, self-adaptation of genetic representations and of exploration distributions is not possible.

Consider the three genotypes given in the first column,

genotype	phenotype	phenotypic neighbors
$\Psi^{(0)} = \langle a \rangle$, $\Pi = \langle \langle a:bcbc \rangle \rangle$	$\langle bc bc \rangle$	$\langle * \rangle$, $\langle *cbc \rangle$, $\langle b*bc \rangle$, $\langle bc*c \rangle$, $\langle bcb* \rangle$
$\Psi^{(0)} = \langle a \rangle$, $\Pi = \langle \langle a:dd \rangle, \langle d:bc \rangle \rangle$	$\langle bc bc \rangle$	$\langle * \rangle$, $\langle *bc \rangle$, $\langle bc* \rangle$, $\langle *c*c \rangle$, $\langle b*b* \rangle$
$\Psi^{(0)} = \langle bc bc \rangle$, $\Pi = \langle \rangle$	$\langle bc bc \rangle$	$\langle *cbc \rangle$, $\langle b*bc \rangle$, $\langle bc*c \rangle$, $\langle bcb* \rangle$

All three genotypes have, after developing for at least two time steps, the same phenotype $\Psi^{(t)} = \langle bc bc \rangle$, $t \geq 2$. The third genotype resembles what one would call a direct encoding, where the phenotype is directly inherited as $\Psi^{(0)}$. Assume that, during mutation, all symbols, except for the promoters, mutate with fixed, small probability. By considering all one-point mutations of the three genotypes, we get the phenotypic neighbors of $\langle bc bc \rangle$ as given in the third column of the table, where a star $*$ indicates the mutated random symbol. This shows that, although all the three genotypes represent the same phenotype, they induce completely different phenotypic variabilities, i.e., completely different phenotypic search distributions. Note that, wherever a phenotypic neighbor comprises two stars, these two phenotypic variations are completely correlated.

In order to enable a variability of genetic representations within such a neutral set we need to allow for mutational transitions between phenotypically equivalent genotypes. A transition from the 1st to the 3rd genotype requires a genetic mutation that applies the operator $\langle a:bc bc \rangle$ on the egg cell $\langle a \rangle$ and deletes it thereafter. Both, the application of an operator on some sequence (be it the egg cell or another operator's) and the deletion of operators will be mutations we provide in our model. The transition from the 2nd to the 1st genotype is similar: the 2nd operator $\langle d:ab \rangle$ is applied on the sequence of the first operator $\langle a:dd \rangle$ and deleted thereafter. But we must also account for the inverse of these transitions. A transition from the 3rd genotype to the 1st is possible if a new operator is created by randomly extracting a subsequence (here $bc bc$ from the egg cell) and encoding it in a new operator (here $\langle a:bc bc \rangle$). The original subsequence is then replaced by the promoter. Similarly, a transition from the 1st to the 2nd geno-

Table 1. The mutation operators in our model.

- First type mutations are ordinary symbol mutations that occur in every sequence (i.e., promoter or rhs of an operator) of the genotype; namely symbol replacement, symbol duplication, and symbol deletion, which occur with equal probabilities. The mutation frequency for every sequence is Poisson distributed with the mean number of mutations given by $(\alpha \cdot \text{sequence-length})$, where α is a global mutation rate parameter.
- The second type mutations aim at a neutral restructuring of the genetic system. A 2nd-type mutation occurs by randomly choosing an operator π and a sequence p from the genome, followed by one of the following operations:
 - application of an operator π on a sequence p ,
 - inverse application of an operator π on a sequence p ; this means that all matches of the operator’s rhs sequence with subsequences of p are replaced in p by the operator’s promoter,
 - deletion of an operator π , only if it was never applied during ontogenesis,
 - application of an operator π on *all* sequences of the genotype followed by deletion of this operator,
 - generation of a new operator ν by extracting a random subsequence of stochastic length $2 + \text{Poisson}(1)$ from a sequence p and encoding it in an operator with random promoter. The new operator ν is inserted in the genome behind the sequence p , followed by the inverse application of ν on p .

All these mutations occur with equal probabilities. The total number of second type mutations for a genotype is Poisson distributed with mean β . The second type mutations are not necessarily neutral but they are neutral with sufficient probability to enable an exploration of neutral sets.

- A genotype is mutated by first applying second type mutations and thereafter first type mutations, the frequencies of which are given by β and α , respectively.

type occurs when the subsequence `bc` is extracted from the operator `<a:bcbc>` and encoded in a new operator `<d:bc>`.

Basically, the mutations we will provide in our model are the generation of new operators by extraction of subsequences (*deflation*), and the application and deletion of existing operators (*inflation*). Technical details can be found in table 1; the main point of these mutation operators though is not their details but that they in principle enable a transition between phenotypically equivalent representations in our encoding. We call these mutations 2nd-type mutations to distinguish them from ordinary symbol mutation, deletion, and insertion.

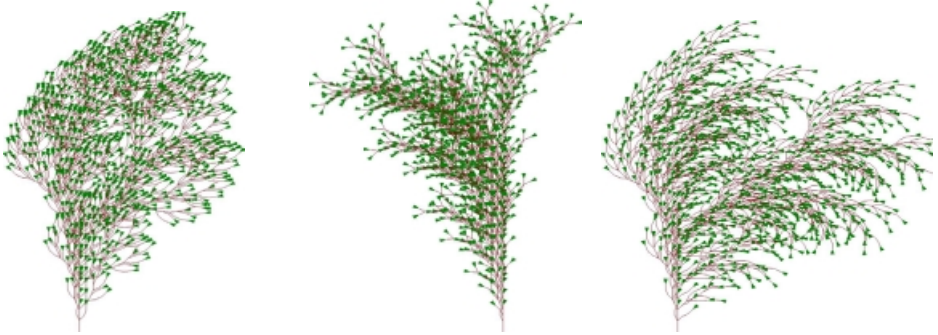
3 Evolving Plants

The symbols for encoding plants. The sequences we are evolving are strings of the alphabet $\{A, \dots, P\}$ which are mapped on plant-describing strings according to $\{A, \dots, I\} \mapsto \{\mathbb{F}, +, -, \&, ^, \backslash, /, [,]\}$ and $\{J, \dots, P\} \mapsto \{\cdot\}$. The meanings of the symbols of the plant-describing strings are summarized in table 2. For example, the sequence `<FF[+F][-F]>` represents a plant for which the stem grows two units upward before it branches in two arms, one to the right, the other to the left, each of which has one unit length and a leave attached at the end.

Table 3 demonstrates the implications of this encoding. The plant on the left is an examples taken from [12]. To the right of this “original” you find three

Table 2. Description of the plant grammars symbols, cf. [11].

F	attach a unit length stem
+, -, &	rotations of the “local coordinate system” (that apply on following attached units): four rotation of δ degree in all four directions away from the stem (+, -, &, ^) and two rotations of $\pm\delta$ degree around the axis of the stem (\, /).
^, \, /	
[	a branching (instantiation of a new local coordinate system)
]	the end of that branch: attaches a leaf and proceeds with the previous branch (return to the previous local coordinate system); also if the very last symbol of the total sequence is not], another leaf is attached
.	does nothing

Table 3. An example for a 2D plant structure and its phenotypic variability induced by *single* symbol mutations in the genotype. For all of them $T = 4$, $\delta = 20$, and $\Psi^{(0)} = \langle F \rangle$. Below each illustration, the single production rule that was mutated is given.


$\langle F:FF-[-F+F+F]+[+F-F-F] \rangle$
 $\langle F:FF-[-F+F+F]+[+F-F-F] \rangle$
 $\langle F:FF-[-F-F+F]+[+F-F-F] \rangle$

different variations, each of which was produced by a *single* symbol mutation in the genotype. Obviously, there are large-scale correlations in the phenotypic variability induced by uncorrelated genetic mutations.

The fitness function. Given a sequence we evaluate its fitness by first drawing the corresponding 3D plant in a virtual environment (the OpenGL 3D graphics environment). We chop off everything of the plant that is outside a bounding cube of size $b \times b \times b$. Then we grab a bird’s view perspective of this plant and measure the area of green leaves as observed from this perspective. The measurement is also height dependent: the higher a leaf (measured by OpenGL’s depth buffer in logarithmic scale, where 0 corresponds to the cube’s floor and 1 to the cube’s ceiling), the more it contributes to the green area integral

$$L = \int_{x \in \text{bird's view area}} \left[\text{color of } x = \text{green} \right] \cdot \left[\text{height at } x \right] \frac{d\text{Area}}{b^2} \in [0, 1]. \quad (1)$$

This integral is the positive component of a plant’s fitness. The negative component is related to the number and “weight” of branch elements: To each element i we associate a weight w_i which is defined recursively. The weight of a leaf is 1; the total weight of a subtree is the sum of weights of all the elements of that subtree; and the weight of a branch element i is 1 plus the total weight of the subtree that is attached to this branch element. E.g., a branch that has a single leaf attached has weight $1 + 1 = 2$, a branch that has two branches each with a single leaf attached has weight $1 + (2 + 1) + (2 + 1) = 7$, etc. The idea is that w_i

roughly reflects how “thick” i has to be in order to carry the attached weight. The total weight of the whole tree,

$$W = \sum_i w_i ,$$

gives the negative component of a plant’s fitness. In our experiments we used

$$f = L - \varrho W$$

as the fitness function, where the penalty factor ϱ was chosen $\varrho \in \{10^{-6}, 10^{-7}\}$ in the different experiments.

Details of the implementation. Evolving such plant structures already gets close to the limits of today’s computers, both, with respect to memory and computation time. Hence, we use some additional techniques to improve efficiency: First, we impose different limits on the memory resource that a genotype and phenotype is allowed to allocate, namely three: (1) The number of symbols in a phenotype was limited to be lower or equal than a number $M_{\max}$. This holds in particular during ontogenesis: If the application of an operator results in a phenotype with too many symbols, then the operator is simply skipped. (2) The number of operators in a genotype is limited to be $\leq R_{\max}$. If a mutation operator would lead to more chromosomes, this mutation operator is simply skipped (no other mutation is made in place). (3) There is a soft limit on the number of symbols in a single chromosome: A duplication mutation is skipped if the chromosome already has length $\geq U_{\max}$. The limit though does not effect 2nd-type mutations; an inflative mutation $\pi \cdot p$ may very well lead to chromosomes of length greater than $U_{\max}$.

Second, we adopt an elaborated technique of self-adaptation of the mutation frequencies. We used the scheme similar to the self-adaptation of strategy parameters proposed by [2]. Every genome i additionally encodes two real numbers α_i and β_i . Before any other mutations are made, they are mutated by

$$\begin{aligned} \alpha_i &\leftarrow \alpha_i (\mathcal{N}(0, \tau) + \tau') , \\ \beta_i &\leftarrow \beta_i (\mathcal{N}(0, \tau) + \tau') , \end{aligned} \tag{2}$$

where $\mathcal{N}(0, \tau)$ is a random sample (independently drawn for α_i and β_i) from the Gaussian distribution $\mathcal{N}(0, \tau)$ with zero mean and standard deviation τ . The parameter τ' allows to induce a pressure towards increasing mutation rates. After this mutation, α_i and β_i determine the mutation frequencies of 1st- and 2nd-type mutations respectively.

The 1st trial. Let us discuss two of the trials made with different parameters. Table 4 summarizes the experimental setup. See figure 1. For the 1st trial, the curves show some sudden changes at generation ~ 4000 where the fitness, the number of phenotypic elements, the number of operators in the genomes, and the total genome length explodes. Between generation ~ 4000 and ~ 5400 , the most significant curve in the graph is the repeatedly decaying genome size. Indeed we will find that the genomes in this period are too large and mutationally unstable. The innovations extinct and genome size decays until at generation ~ 5400 a comparably large number of phenotypic elements can be encoded by much smaller genomes that comprise more operators. In table 5, the illustrations of

Table 4. Parameter settings of the two trials. The “ $\leftrightarrow$ ” means “same as for the previous trial” and shows that only few parameters are changed from trial to trial.

	1st trial	2nd trial	
δ, b	20,20	$\leftrightarrow$	L-system angle δ , size of the bounding cube
T	1	$\leftrightarrow$	stopping time of development
ϱ	10^{-7}	10^{-6}	factor of the weight term W in a plant’s fitness f
α	.01	$\leftrightarrow$	(initial) frequency of first type mutations
β	.01	.3	(initial) frequency of second type mutations
τ (τ')	.5 (.1)	0 (0)	rate of self-adaptation of α and β
μ, λ		$\leftrightarrow$	type of selection
no		$\leftrightarrow$	crossover turned on?
$\langle \text{AAAFFFFJ} \rangle$		$\leftrightarrow$	initialization of $\Psi^{(0)}$ of all genotypes in the first population (they have no operators)
$\mathcal{A}$	$\{\text{A}, \dots, \text{P}\}$	$\leftrightarrow$	symbol alphabet
λ	100	$\leftrightarrow$	(offspring) population size
μ	30	$\leftrightarrow$	(selected) parent population size
$M_{\max}$	100 000	1 000 000	maximal number of symbols in a phenotype
$R_{\max}$	100	$\leftrightarrow$	maximal number of operators allowed in one genotype
$U_{\max}$	40	$\leftrightarrow$	symbol duplication mutations are allowed only if a sequence has less than $U_{\max}$ symbols

the best individual in selected generations explain in more detail what happened. For a long time this is not much until, in generation 4000, a couple of leaves turn up at certain places of the phenotype. Then, very rapidly, more leaves pop up until, in generation 4025, every phenotypic segment has a leave attached. This is exactly what we call a correlated phenotypic adaptation and was enabled by encoding all the segments that now carry a leave within one operator, namely the A-operator. The resulting “long-arm-building-block” triggers a revolution in phenotypic variability and leads to the large structures as illustrated for generation 4400 (3467 elements). However, these structures are not encoded efficiently, the genome size is too large (512) and phenotypic variability becomes chaotic. The species almost extinguishes until, in generation 5100, evolution finds a much better structured genome to encode large phenotypes. The J-operator becomes dominant and allows to encode 1479 phenotypic elements with a genome size of 217. This concept is further improved and evolves until, in generation 8000, a genome of size 141 with 2 operators encodes a regularly structured phenotype of 3652 elements.

The 2nd trial. For the 2nd trial we turned off the self-adaptation mechanism for the mutation frequencies (based on the experience with previous trials we can now estimate a good choice of $\alpha = .01$ and $\beta = .3$ for the mutation frequencies and fix it) and increase the limit $M_{\max}$ to maximally 1 000 000 elements per phenotype. The severe change in the resulting structures is also due to the increase of the weight penalty factor ϱ to 10^{-6} —the final structure of the 1st trial has a weight of about $.3 \cdot 10^{-6}$ which would now lead to a crucial penalty. The weight punishing factor ϱ enforces structures that are regularly branched instead of long curling arms. Table 6 presents the results of the 2nd trial. Comparing the illustrations for generation 950 and 1010 we see that evolution very quickly developed a fan-like structure that is attached at various places of the

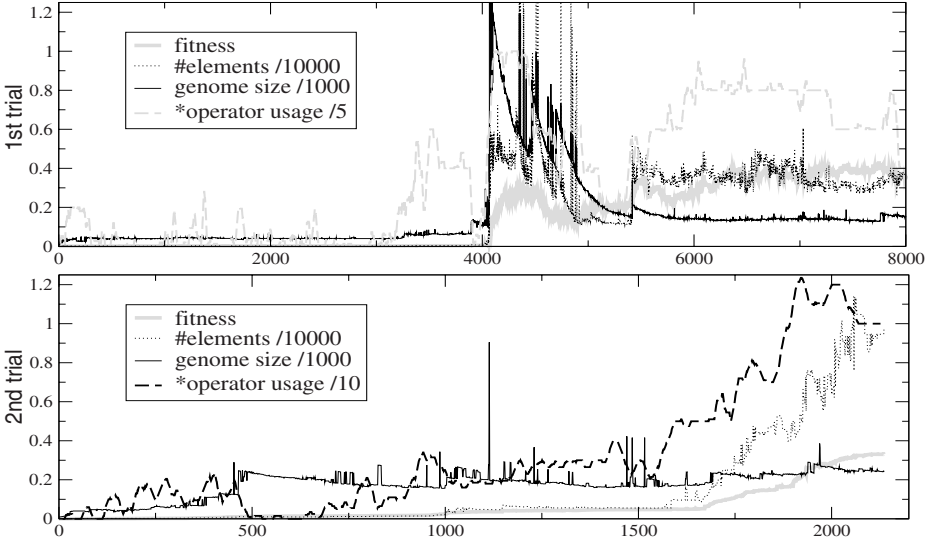


Fig. 1. The graphs display the curves of the fitness, the number of phenotypic elements, the genome size, and the operators usage of the best individual in all generations of the trials. Note that every quantity has been rescaled to fit in the range of the ordinate, e.g., the number of phenotype elements has been divided by 10 000 as indicated by the notation “#elements/10000”. The * for the operator usages indicates that the curve has been smoothed by calculating the running average over an interval of about a hundredth of the abscissae (80 and 20 in the respective graphs).

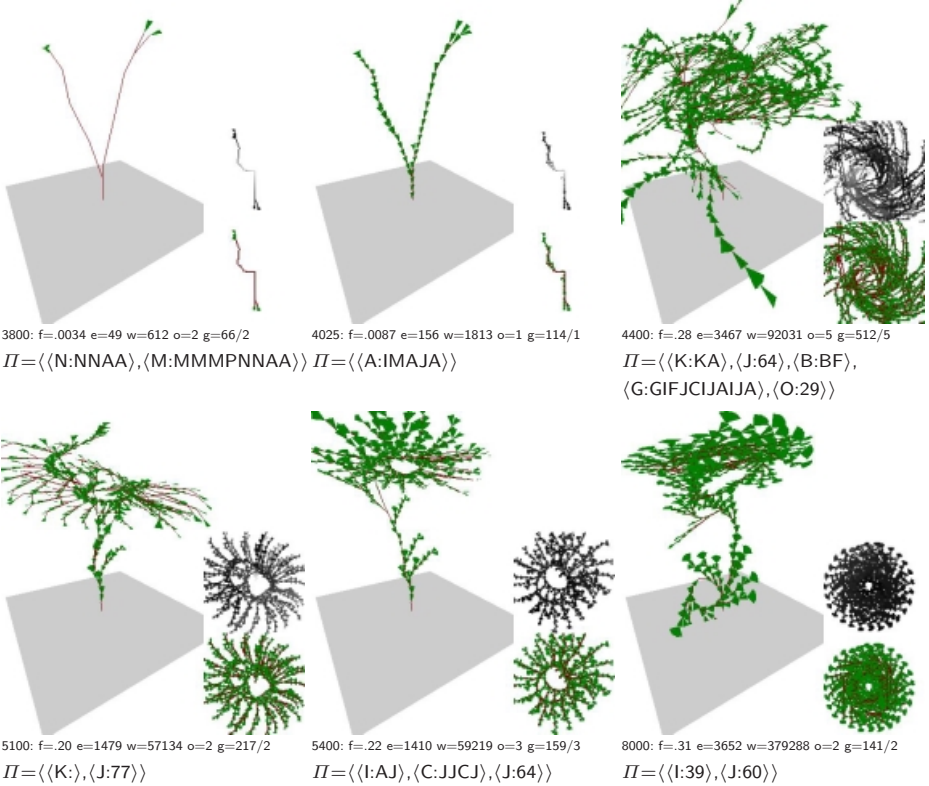
phenotype. The fans arise from an interplay of two operators: The N-operator encodes the fan-like structures while the F-operator encodes the spokes of these fans. Adaptation of these fans is a beautiful example for correlated exploration. The N-operator encodes more and more spokes until the fan is complete in generation 1010, while the F-operator makes the spokes longer. Elongation proceeds and results in the “hairy”, long-armed structures. Note that, in generation 1650, one N- and two B-operators are redundant. Until generation 1900, leaves are attached to each segment of the arms, similar to generation 4025 of the 1st trial. At that time, the plant’s weight is already 105 099 and probably prohibits to make the arms even longer (since weight would increase exponentially). Instead a new concept develops: At the tip of each arm two leaves are now attached instead of one and this quickly evolves until there are three leaves, in generation 1910, and eventually a complete fan of six leaves attached at the tip of each arm. In generation 2100, a comparably short genome with 10 used operators encodes a very dense phenotype structure of 9483 elements.

More trials, data, and source code can be found at the author’s home page.

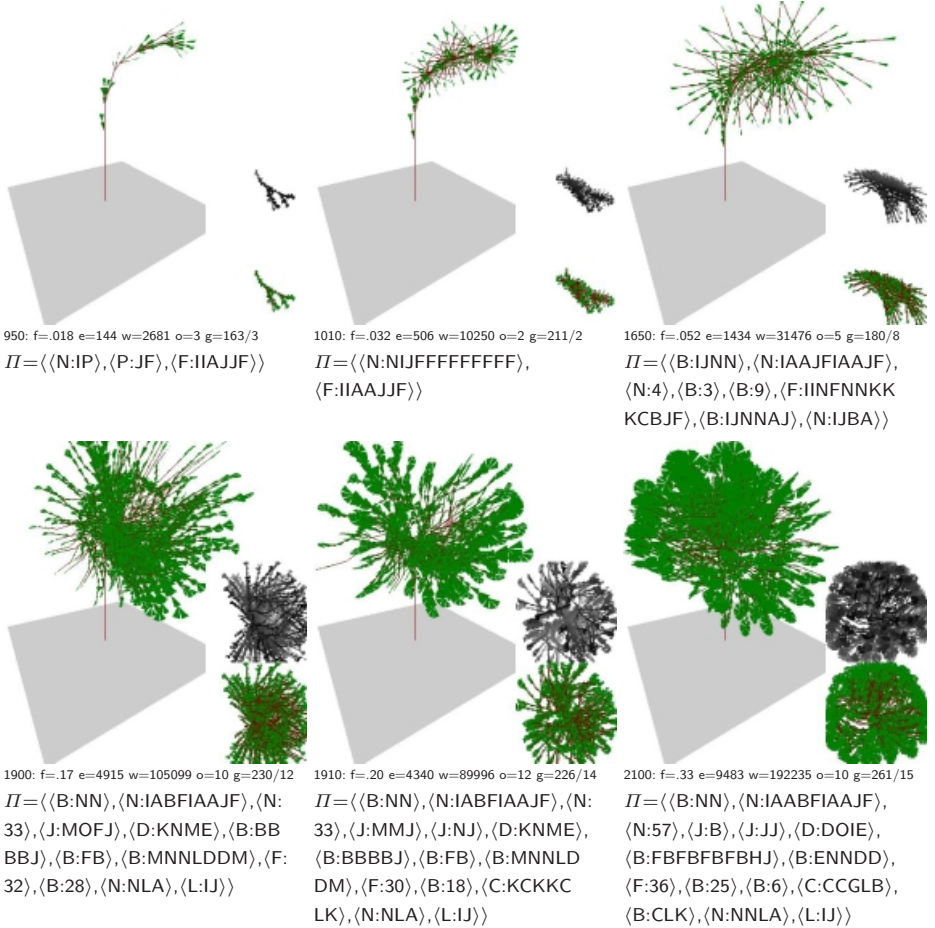
4 Conclusions

Let us briefly discuss whether similar result could have been produced with a more conventional GA that uses, instead of our non-trivial genotype-phenotype mapping, a direct encoding of sequences in $\{F, +, -, \&, ^, \backslash, /, [,]\}$ that describe the plants. For example, setting $\beta = 0$ in our model corresponds to such a

Table 5. The 1st trial. The illustrations display the phenotypes at selected generations. The two squared pictures in the lower right corner of each illustration display exactly the bird's view perspective that is used to calculate the fitness: The lower (colored) picture displays the plant as seen from above and determines which area enters the green area integral in equation (1), and the upper gray-scale picture displays the height value of each element which enters the same equation (where white and black refer to height 0 and 1, respectively). Below each illustration you find some data corresponding to this phenotype: $\langle \text{generation} \rangle$: $f = \langle \text{fitness} \rangle$ $e = \langle \text{number of elements} \rangle$ $w = \langle \text{plant's total weight} \rangle$ $o = \langle \text{number of used operators} \rangle$ $g = \langle \text{genome size} \rangle / \langle \text{number of operators in the genome} \rangle$. The genetic system Π is also displayed. ($\Psi^{(0)}$ is generally too large to be displayed here.) For some operators, the size of the rhs is given instead of the sequence. See the text for a discussion of this evolution.



GA since no operators will be created and the evolution takes places solely on the “egg cell” $\Psi^{(0)}$, which is equal to the final phenotype in the absence of operators. We do not need to present the results of such a trial—not much happens. The obvious reason is the unsolvable dilemma of long sequences in a direct encoding: On the one hand, mutability must be small such that long sequences can be represented stably below the error threshold of reproduction; on the other hand mutability should not vanish in order to allow evolutionary progress. This dilemma becomes predominant when trying to evolve sequences of length $\sim 10^4$, as it is the case for the plants evolved in the 2nd trial. Also elaborate methods of self-adaptation of the mutation rate cannot circumvent

Table 6. The 2nd trial. Please see the caption of table 5 for explanations.

this problem completely; the only way to solve the dilemma is to allow for an adaptation of genetic representations. The key novelty in our model that enabled the adaptation of genetic representations are the 2nd-type mutations we introduced.

In our example, two important features of the genetic representations coincide. First, this is the capability to find compact representations that allow to encode large phenotypes with small genotypes solving the error threshold dilemma. Second, this is the ability for complex adaptation, i.e., to induce highly structured search distributions that incorporate large-scale correlations between phenotypic traits. For example, the variability of one leaf is, in certain representations, not independent of the variability of another leaf. A GA with direct encoding would have to optimize each single phenotypic element by itself, step by step. The advantage of correlated exploration is that many phenotypic elements can be adapted simultaneously in dependence of each other.

Our experiments demonstrated the theory of σ -evolution which mainly states that the evolution of genetic representations is guided by a fundamental principle: they evolve such that the match between the evolutionary search distribution and the distribution of good solutions becomes better. *The way genetic systems are organized is a mirror of what evolution has learned about the problem.*

References

1. L. Altenberg. Genome growth and the evolution of the genotype-phenotype map. In W. Banzhaf and F. H. Eeckman, editors, *Evolution and Biocomputation: Computational Models of Evolution*, pages 205–259. Springer, Berlin, 1995.
2. T. Bäck. *Evolutionary Algorithms in Theory and Practice*. Oxford University Press, 1996.
3. F. Gruau. Automatic definition of modular neural networks. *Adaptive Behaviour*, 3:151–183, 1995.
4. T. F. Hansen and G. P. Wagner. Epistasis and the mutation load: A measurement-theoretical approach. *Genetics*, 158:477–485, 2001.
5. T. F. Hansen and G. P. Wagner. Modeling genetic architecture: A multilinear model of gene interaction. *Theoretical Population Biology*, 59:61–86, 2001.
6. G. S. Hornby and J. B. Pollack. The advantages of generative grammatical encodings for physical design. In *Proceedings of the 2001 Congress on Evolutionary Computation (CEC 2001)*, pages 600–607. IEEE Press, 2001.
7. G. S. Hornby and J. B. Pollack. Evolving L-systems to generate virtual creatures. *Computers and Graphics*, 25:1041–1048, 2001.
8. H. Kitano. Designing neural networks using genetic algorithms with graph generation systems. *Complex Systems*, 4:461–476, 1990.
9. S. Lucas. Growing adaptive neural networks with graph grammars. In *Proc. of European Symp. on Artificial Neural Netw. (ESANN 1995)*, pages 235–240, 1995.
10. M. Pelikan, D. E. Goldberg, and F. Lobo. A survey of optimization by building and using probabilistic models. Technical Report IlliGAL-99018, Illinois Genetic Algorithms Laboratory, 1999.
11. P. Prusinkiewicz and J. Hanan. *Lindenmayer Systems, Fractals, and Plants*, volume 79 of *Lecture Notes in Biomathematics*. Springer, New York, 1989.
12. P. Prusinkiewicz and A. Lindenmayer. *The Algorithmic Beauty of Plants*. Springer, New York, 1990.
13. R. Riedl. A systems-analytical approach to macro-evolutionary phenomena. *Quarterly Review of Biology*, 52:351–370, 1977.
14. M. Toussaint. On the evolution of phenotypic exploration distributions. In C. Cotta, K. De Jong, R. Poli, and J. Rowe, editors, *Foundations of Genetic Algorithms 7 (FOGA VII)*. Morgan Kaufmann, 2003. In press.
15. G. P. Wagner and L. Altenberg. Complex adaptations and the evolution of evolvability. *Evolution*, 50:967–976, 1996.
16. G. P. Wagner, G. Booth, and H. Bagheri-Chaichian. A population genetic theory of canalization. *Evolution*, 51:329–347, 1997.
17. G. P. Wagner, M. D. Laubichler, and H. Bagheri-Chaichian. Genetic measurement theory of epistatic effects. *Genetica*, 102/103:569–580, 1998.
18. R. Watson and J. Pollack. A computational model of symbiotic composition in evolutionary transitions. *Biosystems, Special Issue on Evolvability*, 2002.

Sexual Selection of Co-operation

M. Afzal Upal

Opal Rock Technologies
42741 Center St, Chantilly, VA
afzal@opalrock.com

Abstract. Advocates of sexual selection theory have argued that various male traits, such as male co-operative behavior towards females, can evolve through female preference for mating with those males who possess that trait. This paper reports on the results of a simulation performed to test the hypothesis that female preference for mating with co-operative males can lead to an increase in the proportions of males in a population who co-operate with females. We simply model the sex differences using a single variable measuring the cost of reproduction. Our results show that even in such a simple environment there are a large number of interacting variables, which complicate the relationship between the sexual selection of co-operative males by females and the proportion of males actually co-operating with females. In fact, in most situations we modeled, sexual selection of co-operative males by females ended up causing the proportion of females that co-operate with males to increase while the proportion of males co-operating with females showed no significant increase over the random selection experiments.

1 Introduction and Background

Co-operation is a fundamental part of the repertoire of animal behavior. It has been extensively documented among various social animals such as primates. Co-operative behavior has been observed between members of a group who defend themselves against predators (Corning 1998), between pair-bonded males and females, between parents and children, between infants and non-parent adults (Brown 1970), between challengers to the authority of the dominant male, and between second and third ranking males and the harem females (de Waal 1982, Noe 1992). Evolutionary theorists have identified a number of mechanisms that can confer evolutionary advantages to co-operating individuals in certain situations. However, none of these mechanisms can satisfactorily explain the prevalence of co-operation among a variety of animals in a variety of situations.

Theorists study this problem by using an abstract model known as the Prisoner's Dilemma (Hamilton 1964). Prisoner's Dilemma models a world in which two agents that have to decide whether to co-operate or defect without knowing about the other agent's decision. If both agents decide to co-operate then they get a reward R , if both

decide to defect then both get the mutual defection punishment P , if one decides to co-operate while the other defects then the defector gets the defection reward D while the co-operator gets the sucker's payoff S . When $R > \text{average}(D, S)$ (e.g., as in Table 1) a game-theoretic analysis reveals that defection is the best strategy if the two players never have to play each other again. However, if the agents repeatedly play against each other, different results emerge. Robert Axelrod (1984) held two tournaments to find out the winning strategy in a repeated Prisoner's Dilemma game. He invited others to submit strategies for playing the game and ran the submitted strategies against one another in a round robin fashion to find out which strategy can earn the largest number of points. A strategy called Tit-For-Tat (TFT) turned out to be better than all other strategies including some selfish-looking strategies. TFT strategists start out co-operating and then do what the other player did on the previous move (Axelrod 1984).

Table 1. Common values used for the payoff in the Prisoner's Dilemma. The matrix entries list the player's payoffs as row column.

Decision	Co-operate	Defect
Co-operate	R=3, R=3	S=0, D=5
Defect	D=5, S=0	D=1, D=1

However, this does not explain how co-operation evolves in the first place. To answer this question, Axelrod and Hamilton (1987) set up a simulation in which they randomly generated 100 strategies, allowed them to play against one another. At the end of the playing phase, the relatively more successful strategies were allowed to reproduce while the less successful ones were not. The strategies of the offsprings were generated by crossing-over the strategies of the parents and through random mutation. Axelrod and Hamilton (1987) report that after a significant number of generations, co-operative strategies such as TFT come to dominate. However, subsequent detailed analysis of repeated Prisoner's Dilemma has shown that neither TFT nor any other pure or mixed strategy is evolutionarily stable (Nowak *et al.* 1995). Which strategy emerges as winner depends on the prevalence of the competing strategies in the population. This has lead researchers to consider other factors that can enhance the evolution of co-operative behavior. Key and Aiello (2000) hypothesized that the differences in male and *reproductive cost* may be responsible for evolution of co-operation in a mixed sex environment.

Trivers (1971) defines reproductive cost as the cost of reproduction measured to the extent that it detracts from an individual's ability to invest in future offspring. It consists of the parental investment and the mating costs. Parental investment measures the cost of parent's behavior that directly increases their offspring's reproductive success. For female mammals this includes the high cost of gestation, lactation, and rearing of the offspring. Mating costs are also significantly higher for females (because of the relatively higher costs of producing an egg, and not being able to reproduce while carrying a fetus to term). However, males spend more energy on maintaining their relatively larger bodies and on acquiring mates to reproduce with.

Key and Aiello (2000) modified Axelrod and Hamilton's simulation set up so that the agents were divided into two groups; males and females. The only difference between a male and a female agent was their cost of reproduction. The sex of each one of the 650 players was selected randomly and in most trials roughly half of the agents were males and the other half females. Similar to Axelrod and Hamilton's simulation, initial strategies of the players were randomly selected. A number of rounds of prisoner's dilemma games were played. In each round, two random players (regardless of their sex) were chosen to play prisoner's dilemma game a fixed number of times. However, only relatively successful players of opposite sex were allowed to mate to reproduce off springs. Key and Aiello (2000) ran their simulation by varying the male reproductive cost (MRC) from 1 to 600 and keeping the female reproductive cost (FRC) fixed at 1000. They concluded that at low costs of reproduction, males co-operate more with females than females do with them.

However, Key and Aiello's agents were not able to choose their mates. Mate selection is known to be significant factor affecting the evolution of various male traits. In a number of bird species, females are known to select males by visiting them at their special gathering places (known as leks) where males gather to show off their capabilities. Sexual selection of males by females has been used to explain various male features such as a male peacock's tail, and even speciation itself (Darwin 1900). Recently, some researchers (Miller 2000, Tallamy 2000) have argued that co-operation among males may also have evolved through sexual selection by females. This paper reports on a simulation based study we performed to test the hypothesis that female preference for mating with co-operative males can enhance the evolution of male co-operative behavior.

2 Experimental Setup

Our model involved building a heterosexual population of 100 agents. Each agent was modeled as a Java object with the attributes of sex, reproductive cost, and four 21-bit binary strings encoding the game playing strategy of the player. As shown in Fig.1, the first 21-bits of the 84-bit strategy string are used to encode the strategy that is to be used when a male is playing against another male. The second part when a male is playing against a female, the third when a female is playing against a male and the final part to cover the situations in which a female is playing against another female. If the first bit of the 21-bit string is 1, the player co-operates with its opponent on the first move and defects otherwise. On the second move, the player uses the knowledge of its own last move and its opponent's last move to decide what to do. Since, there are four possibilities (both co-operated, both defected, it co-operated and the opponent defected, it defected and the opponent co-operated) the second move strategy requires 4-bits to encode. Similarly, the player uses the knowledge of its own last two moves and its opponent's last two move to decide what to do on the third and the following moves. This means that 16-bits are needed to encode the strategy for the third and the following moves. As shown by Ikegami (1994), this two-step history is all that an agent needs to encode in order to learn a strategy.

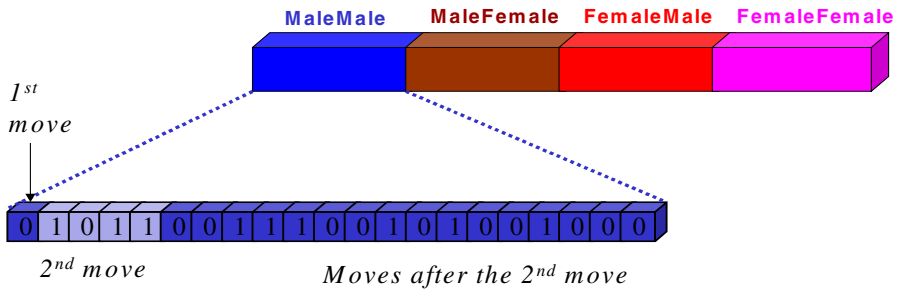


Fig. 1. 84-bit strategy string encoding a player's strategy for playing Prisoner's Dilemma game in a mixed sex environment.

The sex of an agent was randomly chosen to be male or female. Game playing strategies of the first generation of agents were also randomly generated. Next, two players were randomly selected to play 100 rounds of Prisoner's Dilemma game. After 150 game playing rounds, relatively successful players of the opposite sex were allowed to mate and reproduce children. Strategies of the off springs were produced by crossing over the strategies of their parents and through random mutation as shown in Fig. 2. The chances of a strategy gene randomly mutating from 0 to 1 or vice versa were 1 in 5000. Similar to Key and Aiello's experiments, the only difference between a male and female agent was the cost of reproduction. We ran the experiment by varying MRC from 1 to 1000 and keeping FRC fixed at 1000. The metrics we measured were the proportion of males and females receiving co-operation and defection from players of the same and opposite sex.

Similar to Key and Aiello (2000), interactions of a the players of a sex G with a player A were deemed co-operative if the average number of points per game gained by player A from players of sex G (i.e., sum of the points obtained by A from players of sex G divided by the number of Prisoner's Dilemma games A played with players of sex G) exceeded 2.25. If A obtained less than 2.75 points per game from players of sex G , the players of sex G were considered to have been weakly co-operative to A . If A on the other hand, fared less than 1.75 points per game with players of sex G , its interactions with players of sex G , were considered to have been dominated by defections. Those managing between 1.75 and 2.25 points per game from players of sex G were considered to have received weak defections from players of that sex. During each experiment, we computed the proportion of males and females who received co-operation/weak-co-operation/defection/weak-defection from players of the same and the opposite sex. The experiment was run 20 times and the average proportions calculated.

We performed this experiment with two mate selection strategies; random selection and selection of most co-operative male by females. In the first experiment, females randomly selected a male agent to mate with. In the second experiment, a female preferred to mate with the male player who had been the most co-operative to her during their game playing round. If the most co-operative player was not able to mate (because of not having accumulated enough points to mate), the next most co-

operative player was chosen. The purpose of conducting these two experiments was to test the hypothesis that males co-operate significantly more with females when females sexually select males for co-operation. We performed t-tests to see if the differences between the mean proportions obtained from the two experiments indicated two different distributions and if the mean proportion of the females receiving co-operation from males in the sexual selection experiment was significantly (with 0.05 being the level of significance) larger than the mean proportion of females receiving co-operation from males in the random selection experiment.

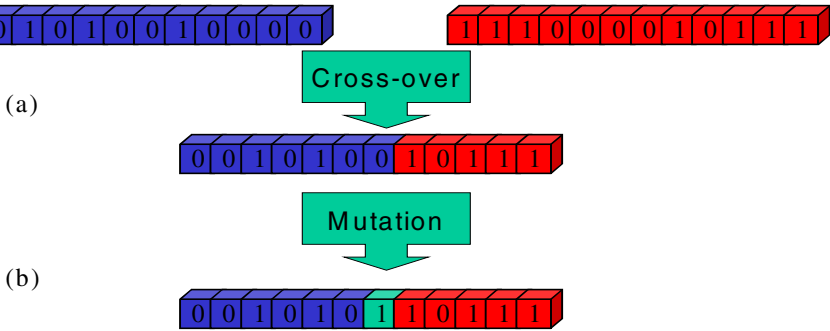


Fig. 2. (a) A random partition point P between 1 and the length of the strategy string N is selected and the first P bits are copied from the first string while the remaining $N-P$ bits are copied from the second bit-string to create the cross-over bit-string. (b) A random bit location L is selected. Another random number R between 1 and 5000 is generated. If R equals 2500 the value of the L th bit is flipped from 1 to 0 or vice versa.

3 Results and Analysis

Fig. 3 and Fig. 4 show the mean proportion of males who receive co-operation/weak-co-operation/weak-defection/defection from other males and females under the random selection and sexual selection conditions for various values of MRC. Fig. 5 and Fig. 6 show the mean proportion of females who receive co-operation/weak-co-operation/weak-defection/defection from males and from other females. The results show that in the random selection experiments, males and females co-operate more with members of the opposite sex than they do with members of the same sex. Similarly, when females randomly select mates, they receive more co-operation from males than they offer in turn (especially at low values of MRC) i.e., at low values of MRC males can afford to be suckers. This confirms the results obtained by Key and Aiello (2000).

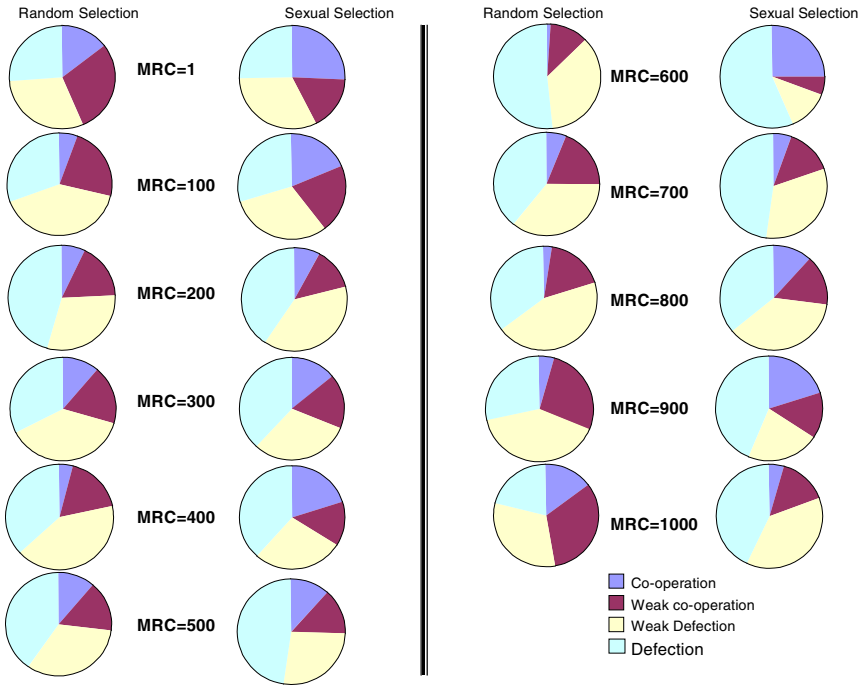


Fig. 3. The mean proportion of males in the 100th generation who co-operate/weakly co-operate/defect/weakly-defect with other males in random and sexual selection experiments for various values of MRC.

Comparing the sexual selection experiment results with random selection results shows that sexual selection of co-operative males by females results in an increase of co-operation between members of the same sex for most values of MRC. However, co-operation between members of opposite sex increases under some conditions while it decreases under different conditions. At very small values of MRC ($MRC < 200$) proportion of females receiving co-operation from males increases as predicted by the null hypothesis but then it declines as MRC increases further. We performed t-tests to see if the increase in male co-operation with females was statistically significant. The graph shown in Fig. 7 plots the t-test values of the difference in the proportion of males co-operating with females between the sexual selection and the random selection populations. The t-test values initially increase and then drop but the increases are not statistically significant at any point. These results contradict our null hypothesis that the sexual selection by females of the most co-operative mate leads to a significant increase in the proportion of females receiving co-operation from males.

The main reason why the proportion of males co-operating with females does not increase (especially at large values of MRC where it actually decreases) is that the male population in the sexual selection environment faces two conflicting selection pressures. Besides the selection pressure exerted by female selection favoring co-operative males, value of male reproductive cost (MRC) also exerts a selection pressure by allowing only those males to mate and pass their genes if they have collected enough points during their game playing phase. The reproductive cost pressure selects for those males who have more competitive strategies and are better point collectors. The two selection pressures would pull in the same direction if co-operation with females also improved a male player's point collection. This would happen if all females adopted reciprocal co-operation strategies (such as tit-for-tat) in which female co-operation with males depended on male co-operation with them.

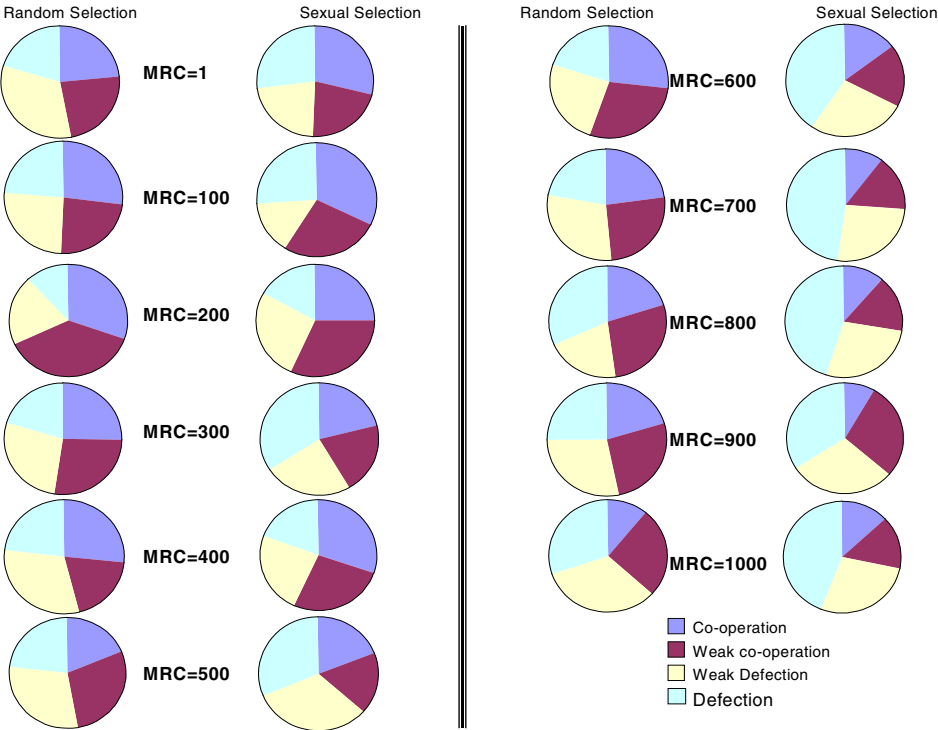


Fig. 4. The mean proportion of males in the 100th generation who co-operate/weakly co-operate/defect/weakly-defect with females in random and sexual selection experiments for various values of MRC.

Increase in co-operation by males in response to other simple female strategies such as always co-operate or always defect would decrease a male's point collection and hence the two selection pressures will pull in the opposite directions. When the two

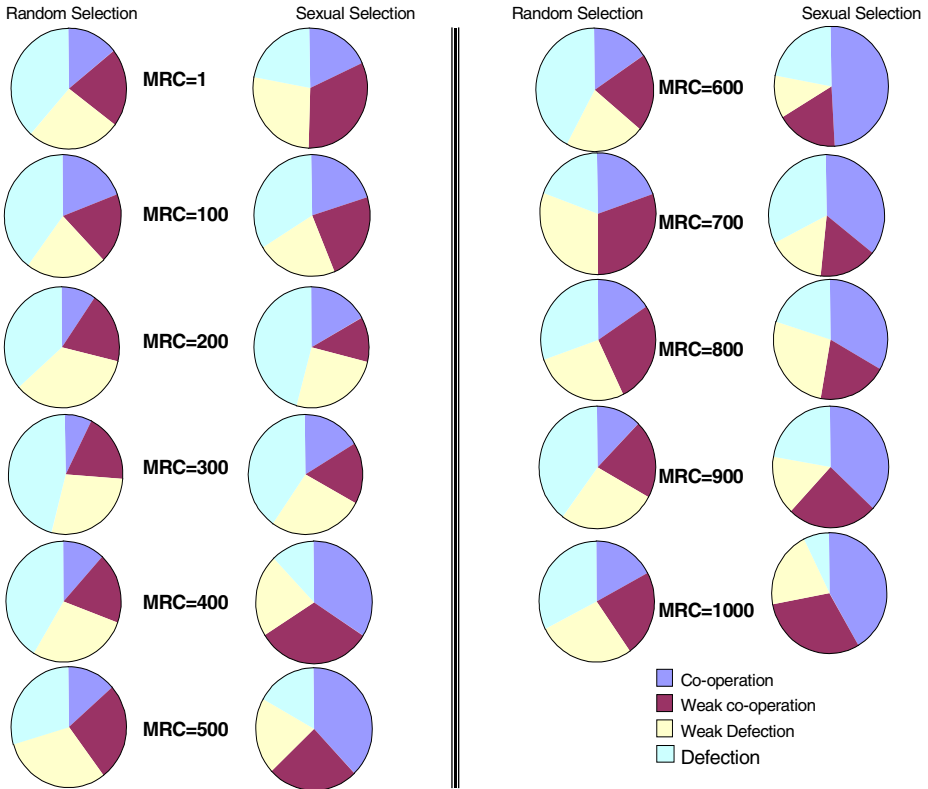


Fig. 5. The mean proportion of females in the 100th generation who co-operate/weakly co-operate/defect/weakly-defect with males in random and sexual selection experiments for various values of MRC.

pressures select for different strategies more complicated interactions ensue. However, even when both selection pressures favor more co-operative male strategies, males may never evolve to adopt the strategy of “always co-operate with females”. This is because such strategies are not evolutionarily stable. If males become mostly co-operative towards females, females would learn the strategy of always defect because that allows them to collect more points. Females have a stronger selection pressure on them to be competitive because of a higher reproductive cost in most cases. Faced with the “always defect with males” strategy, reproductive pressure on males will favor “always defect with females” strategy which runs counter to the co-operative strategies favored by sexual selection pressure. When male reproductive cost is low, male co-operation with females will arise because the sexual selection pressure dominates the reproductive selection pressure. This is what happens as the graph in Fig. 7 illustrates. However, as the male reproductive cost increases, co-operative males despite being favored by the females are simply not able to collect enough points to enable them to mate and propagate their genes.

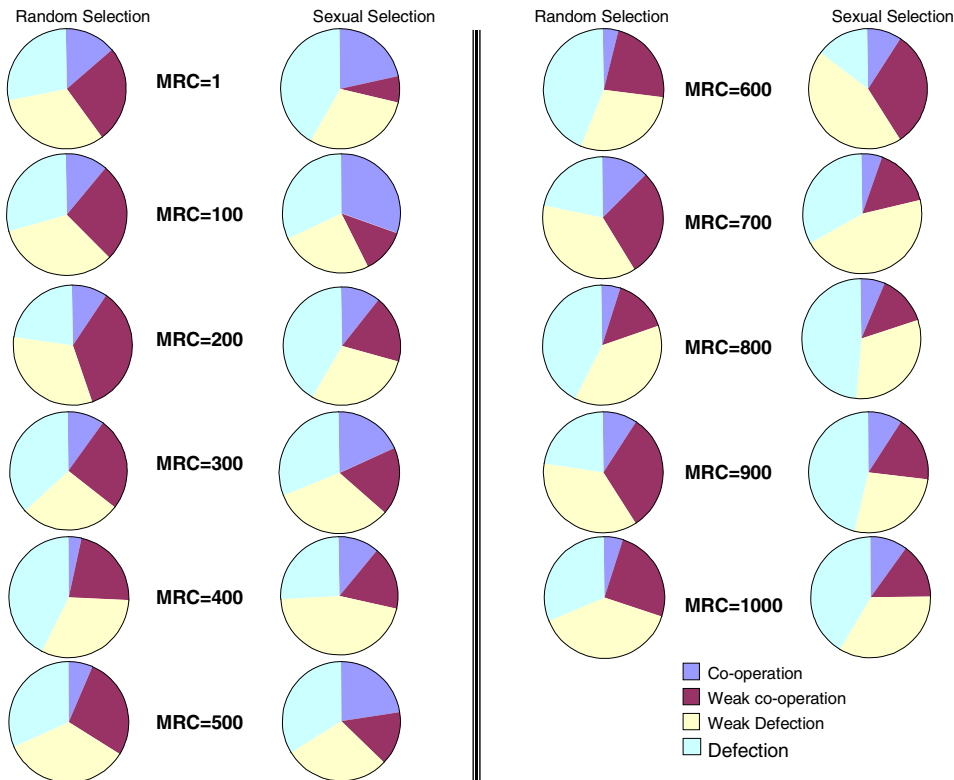


Fig. 6. The mean proportion of females in the 100th generation who co-operate/weakly co-operate/defect/weakly-defect with other females in random and sexual selection experiments for various values of MRC.

There is another solution to the male point collection problem namely to collect more points from other males to compensate for the loss of points that males incur by becoming more co-operative with females. As Fig. 3 shows, this is indeed what happens; male-male co-operation increases at most points. Female-female co-operation also increases at most points as shown in Fig. 6 because females also need to compensate for the loss of points resulting from a drop in male co-operation. However, regardless of how competitive females are against other females, they cannot obtain all their points from females simply because players cannot select the sex of their opponent during the game playing phase (which is done randomly). The reproductive cost selection pressure assures that females must be good at collecting point from both males as well as females. Even though males become less co-operative in the sexual selection population than they were in the random selection population, they are still more co-operative to females than females are to other females.

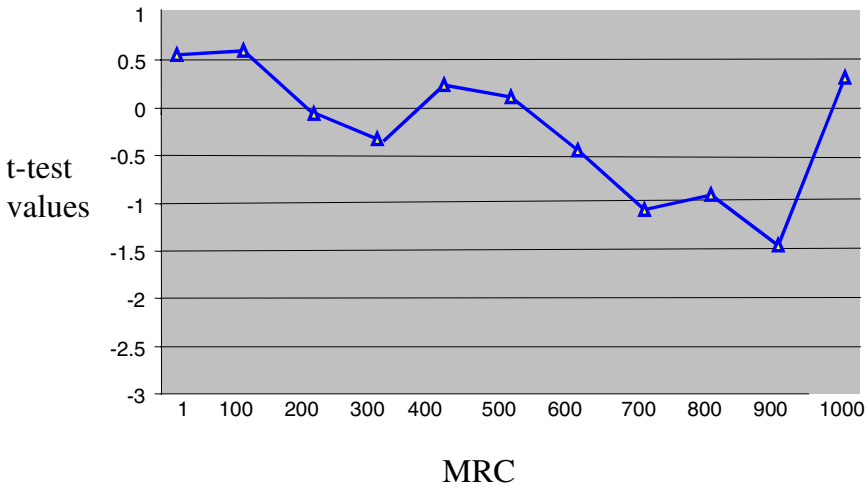


Fig. 7. Student t-test values for the difference between the mean proportion of males co-operating with females between the sexual selection and the random selection population plotted against the value of MRC varied from 1 to 1000.

As male strategies become more competitive, they evolve from being mostly co-operative towards females to more complicated strategies such as reciprocal co-operation strategies that co-operate with only those females who co-operate with them. This means that female strategies of mostly defection gather less points. This means that females must co-operate more with males in order to gain points from them. This is what happens as shown in Fig. 5. Initially, at very small values of MRC ($MRC < 200$) as the male co-operation with females increases (as discussed earlier) female co-operation with them declines. However, at larger values of MRC, as male strategies become more competitive, female co-operation with males increases. We performed t-tests to see if the increase in female co-operation with males in the sexual selection population over the random selection population was statistically significant. As Fig. 8 shows, the t-test values are above the 0.05 threshold at most points (especially at points with larger values of MRC). This shows that there is an emergent selection pressure on the female population favoring those females who are more co-operative to males even though no such selection pressure were explicitly programmed (nor expected when we began this study).

4 Conclusion

Advocates of sexual selection theory have argued that various male traits such as male co-operative behavior towards females can evolve through female preference for males who co-operate with them. This paper presents the results of simulation experi-

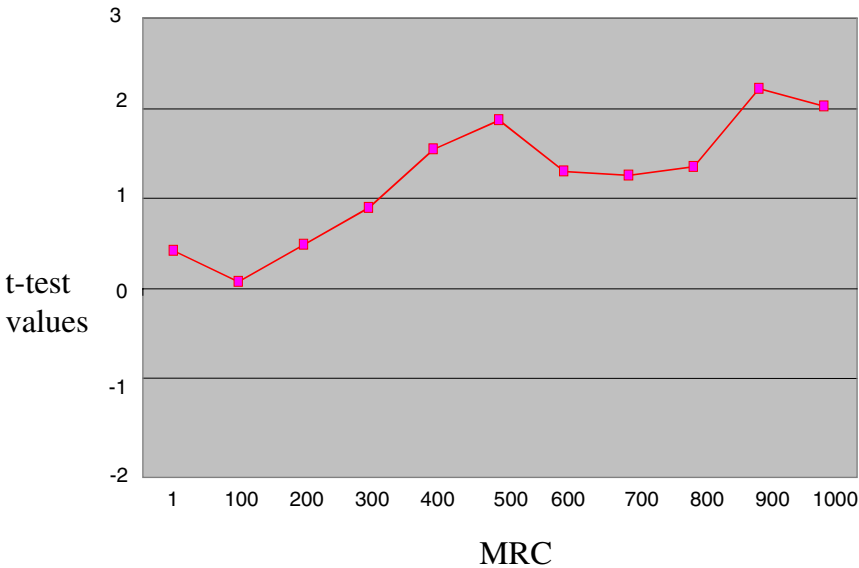


Fig. 8. Student t-test values for the difference between the two population means plotted against the increasing value of MRC from 1 to 1000.

ment we performed to test this hypothesis in a simple population of male and female agents only distinguished by their reproductive costs. Our results show that sexual selection of co-operative males by females does not lead to a significantly larger proportion of males co-operating with females. What we found instead was that it is the proportions of females co-operating with males that ends up increasing. It remains to be seen if these results will hold up in other environments such as n-person Prisoner’s Dilemma and the game of Chicken (Nowak 1995).

References

1. Axelrod, R. (1984) *The Evolution of Co-operation*, Basic Books, New York.
2. Axelrod, R. and Hamilton, W. (1987) “The Evolution of Co-operation”, In Axelrod R. *The Complexity of Co-operation*, Princeton University Press, Princeton.
3. Brown, J. L. (1970) “Cooperative Breeding and Altruistic Behaviour in the Mexican Jay”. *Animal Behaviour*, vol. 18, pp. 366–378. 1970.
4. Corning, P. (1998) “The Cooperative Gene: On the Role of Synergy in Evolution”. *Evolutionary Theory*, vol. 11, pp. 183–207. 1998.
5. Darwin, C. (1900) *The Decent of Man and Selection in Relation to Sex*, Werner Company, Akron, Ohio.
6. De Waal, F. (1982) *Chimpanzee Politics: Power and Sex Among Apes*, John Hopkins University Press, Baltimore MD.
7. Dugatkin, L. (1999) *Cheating Monkeys and Citizen Bees: The Nature of Cooperation in Animals and Humans*. New York: Free Press.

8. Hamilton, W. (1964) The Genetic Evolution of Social Behaviour, *Journal of Theoretical Biology*, vol. 7, pp. 1–52.
9. Ikegami, T. “From Genetic Evolution to Emergence of Game Strategies”, *Physica D*, vol. 75, pp.310–327. 1994
10. Key, C. and Aiello, L. (2000), A Prisoner’s Dilemma Model of the Evolution of Paternal Care, *Folia Primatologica*, vol. 71, pp. 77–92.
11. Miller, (2000), *The Mating Mind: How Sexual Choice Shaped the Evolution of Human Nature*, Doublesday, London.
12. Noe, R. “Alliance Formation Among Male Baboons”, In Harcourt A. & de Waal F. *Coalitions and Alliances in Human and Other Animals*, Oxford University Press, 1992, pp. 285–321.
13. Nowak, M., May, R., and SIGMUND, K. (1995), The Arithmetics of Mutual Help, *Scientific American*, June 1995, pp. 50–55.
14. Tallamy, D. (2000) Sexual Selection and the Evolution of Exclusive Paternal Care in Arthropods, *Animal Behaviour*, vol. 60, pp. 559–567.
15. Trivers, R. (1972) “Parental Investment and Sexual Selection”. In *Sexual Selection and the Decent of Man*, London, Heinemann, 1972, pp. 136–179.

Optimization Using Particle Swarms with Near Neighbor Interactions

Kalyan Veeramachaneni, Thanmaya Peram, Chilukuri Mohan,
and Lisa Ann Osadciw

Department of Electrical Engineering and Computer Science
Syracuse University
Syracuse, NY 13244-1240

(315)443-3366(office)/(315)443-2583(fax)

kveerama, tperam, mohan, laosadci@ecs.syr.edu

Abstract. This paper presents a modification of the particle swarm optimization algorithm (PSO) intended to combat the problem of premature convergence observed in many applications of PSO. In the new algorithm, each particle is attracted towards the best previous positions visited by its neighbors, in addition to the other aspects of particle dynamics in PSO. This is accomplished by using the ratio of the relative fitness and the distance of other particles to determine the direction in which each component of the particle position needs to be changed. The resulting algorithm, known as Fitness-Distance-Ratio based PSO (FDR-PSO), is shown to perform significantly better than the original PSO algorithm and several of its variants, on many different benchmark optimization problems. Avoiding premature convergence allows FDR-PSO to continue search for global optima in difficult multimodal optimization problems, reaching better solutions than PSO and several of its variants.

1 Introduction

The Particle Swarm Optimization algorithm (PSO), originally introduced in terms of social and cognitive behavior by Kennedy and Eberhart in 1995 [1], [2], has proven to be a powerful competitor to other evolutionary algorithms such as genetic algorithms [3]. The PSO algorithm simulates social behavior among individuals (particles) “flying” through a multidimensional search space, each particle representing a single intersection of all search dimensions[7]. The particles evaluate their positions relative to a goal (fitness) at every iteration, and particles in a local neighborhood share memories of their “best” positions, then use those memories to adjust their own velocities and positions as shown in equations (1) and (2) below. The PSO formulae define each particle as a potential solution to a problem in a D-dimensional space, with the i th particle represented as $X_i = (x_{i1}, x_{i2}, x_{i3}, \dots, x_{iD})$. Each particle also remembers its previous best position, designated as $pbest$, $P_i = (p_{i1}, p_{i2}, p_{i3}, \dots, p_{iD})$ and its velocity $V_i = (v_{i1}, v_{i2}, v_{i3}, \dots, v_{iD})$ [7]. In each generation, the velocity of each particle is updated, being pulled in the direction of its own previous best position (p_i) and the best of all positions (p_g) reached by all particles until the preceding generation.

The original PSO formulae developed by Kennedy and Eberhart were modified by Shi and Eberhart [4] with the introduction of an inertia parameter, ω , that was shown empirically to improve the overall performance of PSO.

$$V_{id}^{(t+1)} = \omega \times V_{id}^{(t)} + \psi_1 \times (p_{id} - X_{id}^{(t)}) + \psi_2 \times (p_{gd} - X_{id}^{(t)}) \quad (1)$$

$$X_{id}^{(t+1)} = X_{id}^{(t)} + V_{id}^{(t+1)} \quad (2)$$

Several interesting variations of the PSO algorithm have recently been proposed by researchers in [12], [13], [14], [15], [16], [17]. Many of these PSO improvements are essentially extrinsic to the particle dynamics at the heart of the PSO algorithm and can be applied to augment the new algorithm presented in this paper. By contrast to most other PSO variations, this paper proposes a significant modification to the dynamics of particles in PSO, moving each particle towards other nearby particles with a more successful search history, instead of just the best position discovered so far. This is in addition to the terms in the original PSO update equations.

Section 2 motivates and describes the new Fitness- Distance-Ratio based PSO (FDR-PSO) algorithm. Section 3 defines the benchmark continuous optimization problems used for experimental comparison of the algorithms, and the experimental settings for each algorithm. Section 4 presents and discusses the results. Conclusions and future work are presented in Section 5.

2 FDR-PSO Algorithm

Theoretical results [10][13] have shown that the particle positions in PSO oscillate in damped sinusoidal waves until they converge to points in between their previous best positions and the global best positions discovered by all particles so far. If some point visited by a particle during this oscillation has better fitness than its previous best position (as is very likely to happen in many fitness landscapes), then particle movement continues, generally converging to the global best position discovered so far. All particles follow the same behavior, quickly converging to a good local optimum of the problem. However, if the global optimum for the problem does not lie on a path between original particle positions and such a local optimum, then this convergence behavior prevents effective search for the global optimum. It may be argued that many of the particles are wasting computational effort in seeking to move in the same direction (towards the local optimum already discovered), whereas better results may be obtained if various particles explore other possible search directions. This paper explores an alternative in which each particle is influenced by several other particles, not just moving towards or away from the best position discovered so far.

The most logical choices, for deciding which other particles ought to influence a given particle, are drawn from natural observations and expectations of animal behavior:

1. An organism is most likely to be influenced by others in its neighborhood.

2. Among the neighbors, those that have been more successful (than itself) are likely to affect its behavior.

Attempting to introduce the effects of multiple other (neighboring) particles on each particle must face the possibility of crosstalk effects encountered in neural network learning algorithms. In other words, the pulls experienced in the directions of multiple other particles may mostly cancel each other, reducing the possible benefit of all the associated computations. To counteract this possibility, the FDR-PSO algorithm selects only one other particle when updating each velocity dimension, which is chosen to satisfy two criteria:

1. It must be near the particle being updated.
2. It should have visited a position of higher fitness.

Experiments have been conducted with several possible ways of selecting particles that satisfy these criteria, without significant difference in the performance of the resulting algorithm. The simplest and most robust variation was to update each velocity dimension by selecting a particle that maximizes the ratio of the fitness difference to the one-dimensional distance. In other words, the d th dimension of the i th particle's velocity is updated using a particle called the $nbest$, with prior best position P_j , chosen to maximize

$$FDR(j, i, d) = \frac{Fitness(P_j) - Fitness(X_i)}{|P_{jd} - X_{id}|} \quad (3)$$

where $|\dots|$ denotes the absolute value, and it is presumed that the fitness function is to be maximized. The above expression is called the Fitness-Distance-Ratio, suggesting the name FDR-PSO for the algorithm; for a minimization problem, we would instead use $(Cost(P_j) - Cost(X_i))$ in the numerator of the above expression.

This version of the algorithm has been more successful than variations such as selecting a single particle in whose direction all velocity components are updated. The pseudocode for this algorithm is given in Figure 1.

3 Experimental Settings and Benchmark Problems

Experiments were conducted with several variations of FDR-PSO, obtained by changing the parameter values ψ_1, ψ_2, ψ_3 . The results in the tables and figures use the notation “FDR-PSO(ψ_1, ψ_2, ψ_3)”. Note that FDR-PSO(1,1,0) is the same as the usual PSO algorithm described by Kennedy and Eberhart. On the other hand, FDR-PSO(0,1, ψ_3) and FDR-PSO(1,0, ψ_3) correspond to the variations in which one of the main components of the old PSO algorithm is completely deleted.

“FDR-PSO (1,1, ψ_3)” refers to an instance of the new algorithm in which the relative weightage of the new term is “ ψ_3 ” and the terms of the old PSO algorithm remain unchanged. In all the implementations, the inertia parameter is decremented with number of iterations as in [11].

$$\omega(i) = \frac{(\omega - 0.4) \times (gsize - i)}{gsize + 0.4} \quad (4)$$

where $\omega = 0.9$;

where $gsize$ is the maximum number of generations for which the algorithm runs, i is the present generation number.

FDR-PSO was compared against two variants of random search algorithms, to verify whether the particle dynamics are of any use at all. In the “Random Velocity Update algorithm” the new velocity term = old velocity term + a number chosen from the interval $[-width/10, width/10]$, where “width” is the difference between the max. and min. possible values for that dimension. In the “Random Position Update algorithm”, with no explicit velocity contributing, new position = old position + a random number chosen in the same manner.

Algorithm FDR-PSO:

For $t = 1$ to the max. bound of the number on generations,

For $i = 1$ to the population size,

For $d = 1$ to the problem dimensionality,

Apply the velocity update equation:

$$V_{id}^{t+1} = \omega \times V_{id}^t + \psi_1 \times (p_{id} - X_{id}) + \psi_2 \times (p_{gd} - X_{id}) + \psi_3 \times (p_{nd} - X_{id})$$

where P_i is the best position visited so far by X_i ,

P_g is the best position visited so far by any particle

and P_n is chosen by maximizing

$$\frac{Fitness(P_j) - Fitness(X_i)}{|P_{jd} - X_{id}|};$$

Limit magnitude:

$$V_{id}^{(t+1)} = \min(V_{max}, \max(-V_{max}, V_{id}^{(t+1)}));$$

Update Position:

$$X_{id}^{(t+1)} = \min(Max_d, \max(-Min_d, X_{id}^{(t)} + V_{id}^{(t+1)}));$$

End- for- d ;

Compute fitness of $(X_i^{(t+1)})$;

If needed, update historical information regarding P_i and P_g ;

End-for- i ;

Terminate if P_g meets problem requirements;

End-for- t ;

End algorithm.

Fig. 1. Pseudocode for FDR-PSO algorithm

All the experiments were conducted using a population size of 10, with each algorithm executed for a maximum of 1000 generations. Experiments were conducted with the following benchmark problems for a dimensionality of $n=20$. All the benchmarks have global minima at the origin.

3.1 De Jong's function 1

$$f(x) = \sum_{i=1}^n x_i^2 \quad \text{where } -5.12 \leq x_i \leq 5.12 \quad (5)$$

3.2 Axis parallel hyper-ellipsoid

$$f(x) = \sum_{i=1}^n i \times x_i^2 \quad \text{where } -5.12 \leq x_i \leq 5.12 \quad (6)$$

3.3 Rotated hyper-ellipsoid

$$f(x) = \sum_{i=1}^n \left(\sum_{j=1}^i x_j \right)^2 \quad \text{where } -65.536 \leq x_i \leq 65.536 \quad (7)$$

3.4 Rosenbrock's Valley (Banana function)

$$f(x) = \sum_{i=1}^{n-1} 100 \times (x_{i+1} - x_i^2)^2 + (1 - x_i)^2 \quad \text{where } -2.048 \leq x_i \leq 2.048 \quad (8)$$

3.5 Griewangk's function

$$f(x) = \sum_{i=1}^n \frac{x_i^2}{4000} - \prod_{i=1}^n \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1 \quad \text{where } -600 \leq x_i \leq 600 \quad (9)$$

3.6 Sum of different powers

$$f(x) = \sum_{i=1}^n |x_i|^{i+1} \quad \text{where } -1 \leq x_i \leq 1 \quad (10)$$

4 Results and Discussion

Figures 2 through 7 present the results on the optimization functions defined in the previous section. The graphs show results averaged over 30 trials. In each trial, the population is randomly initialized and the same population is used for PSO and FDR-PSO.

As shown in Figures 2, 3, 4, 5, 6, 7 and Table 1, the new FDR-PSO algorithm outperforms the classic PSO algorithm on each of the benchmark problems on which the experiments have been conducted so far. In each case, the original PSO algorithm performs well in initial iterations but fails to make further progress in later iterations.

The significant improvement achieved by the FDR-PSO algorithm can be attributed to the near neighbor interactions. Population diversity is achieved by allowing particles to learn from their nearest best neighbor which may be of poorer fitness than the global best. FDR-PSO's learning is consistent with the social behavior of the individuals in groups, i.e., learning from the nearest best neighbors with successful search history rather than learning from only the global best. In some cases, the nearest best neighbors can be the

global best itself and hence can imply re-emphasizing the social learning. However, the probability of the nearest best neighbor being the global best decreases with the increase in population as well as the dimensionality of the problem. The algorithm in this paper has been implemented for a population of 10. The probability of the global best being the nearest best neighbor was observed to be as low as 0.4. Increasing the population size would result in a more robust implementation of this algorithm and is expected to result in a more diverse population. Such an implementation can be used for a more difficult multimodal search space where a diverse and localized PSO is a requirement.

The population diversity that is achieved can be demonstrated by the fact that the best fitness and average population fitness became identical within 500 generations when the PSO algorithm was applied to Rosenbrock's problem, whereas this did not occur until about 1000 generations when the FDR-PSO algorithm was applied. Similar results were observed for all the other benchmark problems, this shows that the new algorithm is less plagued by the premature convergence problem faced by the PSO.

Table 1. Minima achieved for different optimization functions using different algorithms

Algorithm	De Jong's	Rosenbrock's	Axis Parallel Hyper-Ellipsoid	Rotated Hyper-Ellipsoid	Griewangk's	Sum of Powers
PSO	0.0239	6.8309	0.1250	55.85	5.0501	1.8e-7
FDR(111)	0.0027	6.0802	0.0230	20.5686	3.6946	7.32e-11
FDR (112)	2.02e-5	4.8717	1.07e-5	1.2776	0.0475	5.3e-19
FDR (102)	2.63e-7	5.7389	7.6e-5	365.0034	0.4172	4.8e-17
FDR (012)	8.36e-6	5.0130	3.6e-4	0.9080	0.0308	3.8e-11
FDR(002)	0.0010	8.2869	0.0035	1513.2	2.1735	3.3e-12

The results also show that the FDR-PSO algorithm can perform well in the absence of the social or cognitive terms. By ranking the algorithm in the decreasing order of their performance, it can be seen that the top three positions are shared by FDR-PSO(112), FDR-PSO(012), FDR-PSO(102) with FDR-PSO(112) being the best in most of the benchmark problems. It is interesting to note that FDR-PSO(111) has always been a poor performer in the FDR-PSO family. This demonstrates the sensitivity of the algorithm to the weight given to the "nbest" term. The "near neighbor" term, however, remains the most important term of the new algorithm with PSO related terms adding a little more to the performance. This can be seen from the fact that the FDR-PSO(002) outperforms the standard PSO and the FDR-PSO(111) in four benchmark problems. The versions, random velocity update and random position updates are worst performers of all.

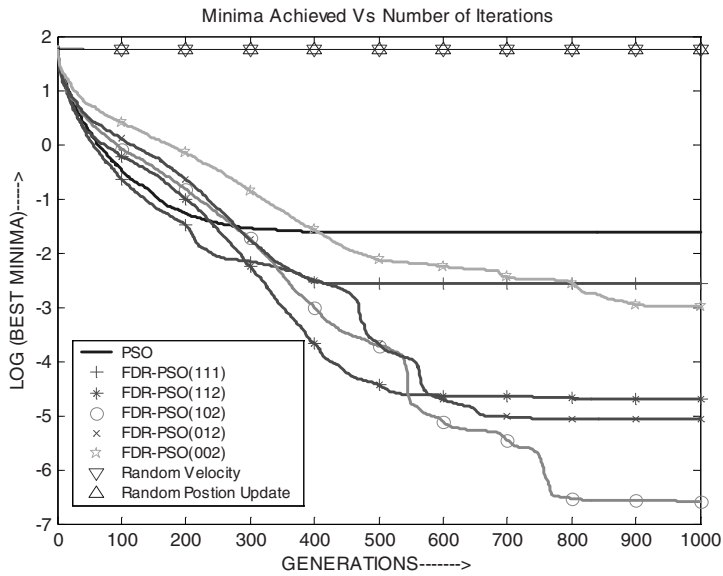


Fig. 2. Best minima plotted against the number of generations for each algorithm, for DeJong's function, averaged over 30 trials

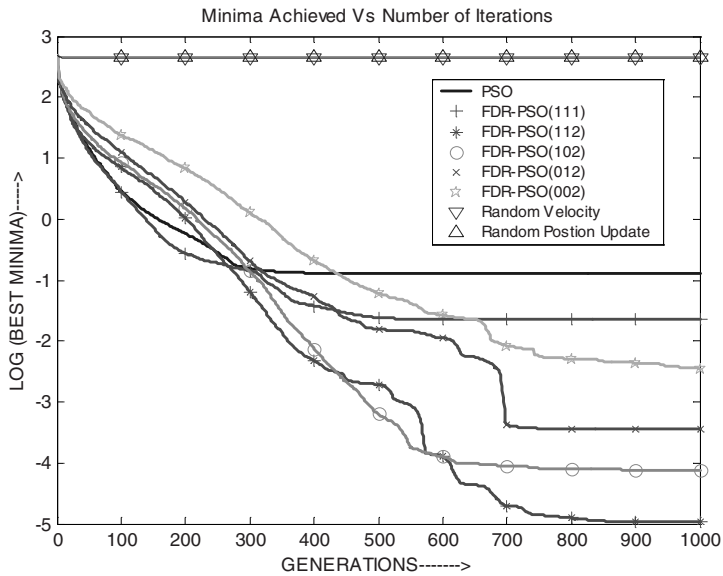


Fig. 3. Best minima plotted against the number of generations for each algorithm, for Axis parallel hyper-ellipsoid, averaged over 30 trials

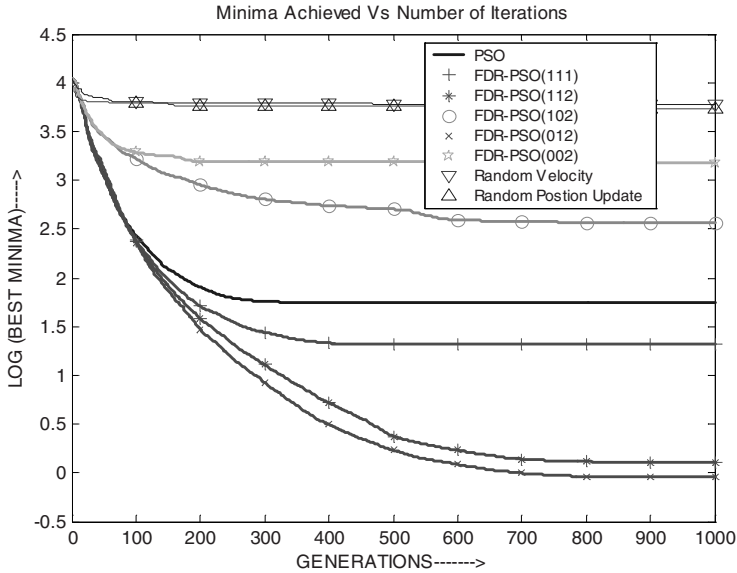


Fig. 4. Best minima plotted against the number of generations for each algorithm, for Rotated hyper-ellipsoid, averaged over 30 trials

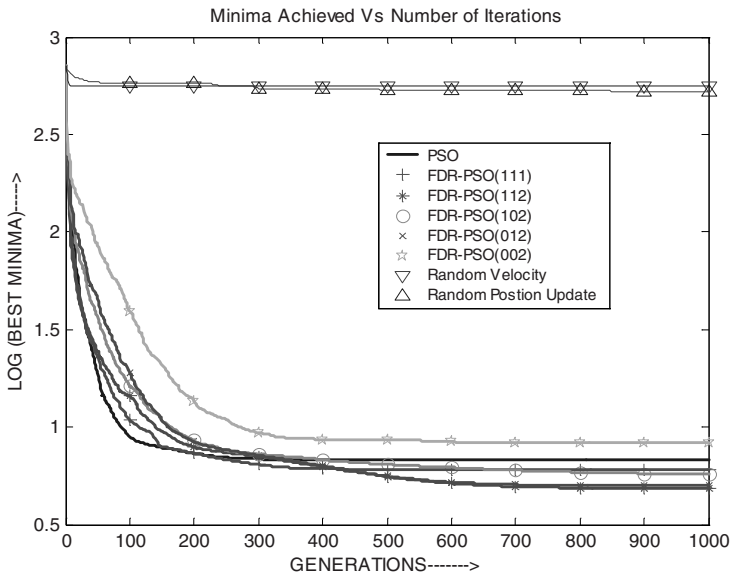


Fig. 5. Best minima plotted against the number of generations for each algorithm, for Rosenbrock's Valley, averaged over 30 trials

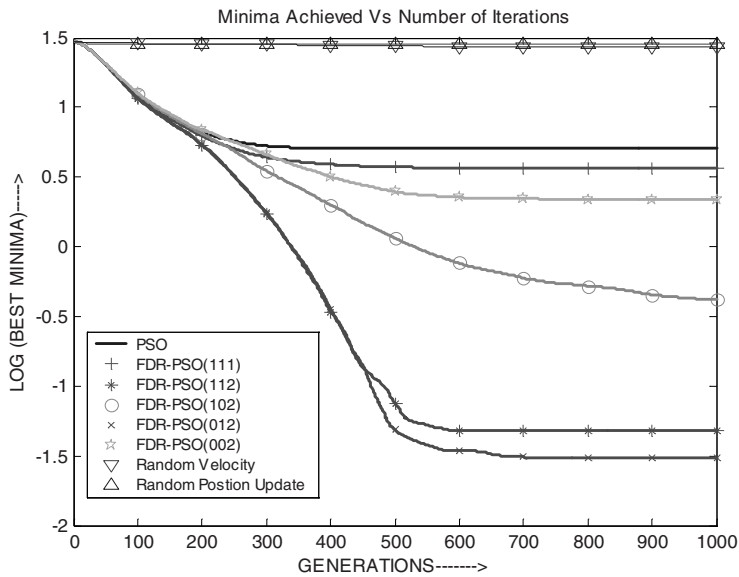


Fig. 6. Best minima plotted against the number of generations for each algorithm, for Griewangk's Function, averaged over 30 trials

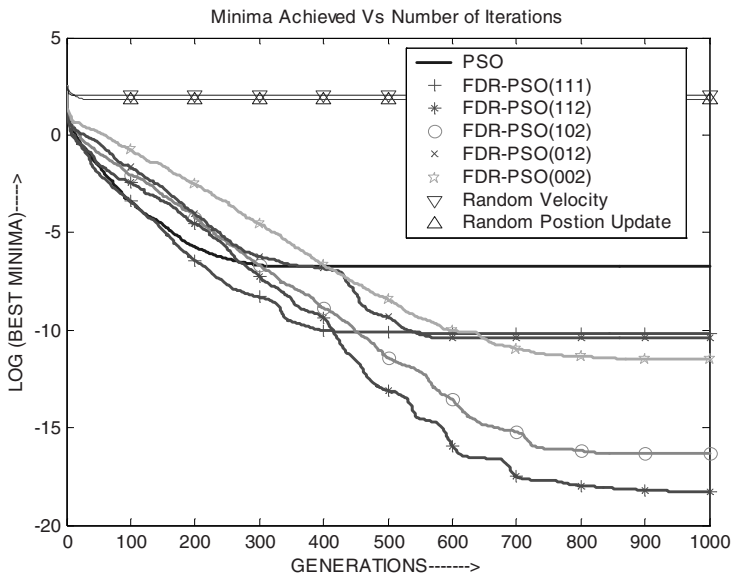


Fig. 7. Best minima plotted against the number of generations for each algorithm, for Sum of Powers, averaged over 30 trials

Several other researchers have proposed different variations of PSO. For example, ARPSO[17] uses a diversity measure to have the algorithm alternate between two phases i.e., attraction and repulsion. In this algorithm, 95% of the fitness improvements were achieved in the attraction phase and the repulsion phase merely increases the diversity. In the attraction phase the algorithm runs as the basic PSO, while in the repulsion phase the particles are merely pushed in opposite direction of the best solution achieved so far.

The random restart mechanism has also been proposed under the name of “PSO with Mass Extinction”[15]. In this, after every “ I_e ” generations, called the extinction interval, the velocities of the swarm are reinitialised with random numbers. Researchers have also explored increasing diversity by increasing randomness associated with velocity and position updates, thereby discouraging swarm convergence, in the “Dissipative PSO”[16]. Lovbjerg and Krink have explored extending the PSO with “Self Organized Criticality”[14], aimed at improving population diversity. In their algorithm, a measure, called “criticality”, describing how close to each other are the particles in the swarm, is used to determine whether to relocate particles. Lovbjerg, Rasmussen, and Krink also proposed in [6], an idea of splitting the population of particles into subpopulations and hybridizing the algorithm, borrowing the concepts from Genetic algorithms. All these variations perform better than the PSO. These variations however seem to add new control parameters, such as, extinction interval in [15], diversity measure in [17], criticality in [14], and various genetic algorithm related parameters in [6], which can be varied and have to be carefully decided upon. The beauty of FDR-PSO lies in the fact that it has no more additional parameters than the PSO and achieves the objectives achieved by any of these variations and reaches a better minima. Table 2 compares the FDR-PSO algorithm with these variations. The comparisons were performed by experimenting FDR-PSO(1, 1, 2) on the benchmark problems with approximately the same settings as reported in the experiments of those variations. In all the cases the FDR-PSO outperforms the other variations.

Table 2. Minima achieved by different variations of PSO and FDR-PSO

Algorithm	Dimensions	Generations	Griewangk's Function	Rosenbrock's Function
PSO	20	2000	0.0174	11.16
GA	20	2000	0.0171	107.1
ARPSO	20	2000	0.0250	2.34
FDR-PSO(112)	20	2000	0.0030	1.7209
PSO	10	1000	0.08976	43.049
GA	10	1000	283.251	109.81
Hybrid(1)	10	1000	0.09078	43.521

Algorithm	Dimensions	Generations	Griewangk's Function	Rosenbrock's Function
Hybrid(2)	10	1000	0.46423	51.701
Hybrid(4)	10	1000	0.6920	63.369
Hybrid(6)	10	1000	0.74694	81.283
HPSO1	10	1000	0.09100	70.41591
HPSO2	10	1000	0.08626	45.11909
FDR-PSO(112)	10	1000	0.0148	9.4408

5 Conclusions

This paper has proposed a new variation of the particle swarm optimization algorithm called FDR-PSO, introducing a new term into the velocity component update equation: particles are moved towards nearby particles' best prior positions, preferring positions of higher fitness. The implementation of this idea is simple, based on computing and maximizing the relative fitness-distance-ratio. The new algorithm outperforms PSO on many benchmark problems, being less susceptible to premature convergence, and less likely to be stuck in local optima. FDR-PSO algorithm outperforms the PSO even in the absence of the terms of the original PSO.

From one perspective, the new term in the update equation of FDR-PSO is analogous to a recombination operator where recombination is restricted to individuals in the same region of the search space. The overall evolution of the PSO population resembles that of other evolutionary algorithms in which offspring are mutations of parents, whom they replace. However, one principal difference is that algorithms in the PSO family retain historical information regarding points in the search space already visited by various particles; this is a feature not shared by most other evolutionary algorithms.

In current work, a promising variation of the algorithm, with the simultaneous influence of multiple other neighbors on each particle under consideration, is being explored. Future work includes further experimentation with parameters of FDR-PSO, testing the new algorithm on other benchmark problems, and evaluating its performance relative to EP and ES algorithms.

References

1. Kennedy, J. and Eberhart, R., "Particle Swarm Optimization", IEEE International Conference on Neural Networks, 1995, Perth, Australia.
2. Eberhart, R. and Kennedy, J., "A New Optimizer Using Particles Swarm Theory", Sixth International Symposium on Micro Machine and Human Science, 1995, Nayoga, Japan.
3. Eberhart, R. and Shi, Y., "Comparison between Genetic Algorithms and Particle Swarm Optimization", The 7th Annual Conference on Evolutionary Programming, 1998, San Diego, USA.

4. Shi, Y. H., Eberhart, R. C., "A Modified Particle Swarm Optimizer", IEEE International Conference on Evolutionary Computation, 1998, Anchorage, Alaska.
5. Kennedy J., "Small Worlds and MegaMinds: Effects of Neighbourhood Topology on Particle Swarm Performance", Proceedings of the 1999 Congress of Evolutionary Computation, vol. 3, 1931-1938. IEEE Press.
6. Lovbjerg, M., Rasmussen, T. K., Krink, T., "Hybrid Particle Swarm Optimiser with Breeding and Subpopulations", Proceedings of Third Genetic Evolutionary Computation, (GECCO 2001).
7. Carlisle, A. and Dozier, G. "Adapting Particle Swarm Optimization to Dynamic Environments", Proceedings of International Conference on Artificial Intelligence, Las Vegas, Nevada, USA, pp. 429-434, 2000.
8. Kennedy, J., Eberhart, R. C., and Shi, Y. H., *Swarm Intelligence*, Morgan Kaufmann Publishers, 2001.
9. GEATbx: Genetic and Evolutionary Algorithm Toolbox for MATLAB, Hartmut Pohlheim, http://www.systemtechnik.tu-ilmenau.de/~pohlheim/GA_Toolbox/index.html.
10. E. Ozcan and C. K. Mohan, "Particle Swarm Optimization: Surfing the Waves", Proceedings of Congress on Evolutionary Computation (CEC'99), Washington D. C., July 1999, pp 1939-1944.
11. Particle Swarm Optimization Code, Yuhui Shi, www.engr.iupui.edu/~shi
12. van den Bergh, F., Engelbrecht, A. P., "Cooperative Learning in Neural Networks using Particle Swarm Optimization", South African Computer Journal, pp. 84-90, Nov. 2000.
13. van den Bergh, F., Engelbrecht, A. P., "Effects of Swarm Size on Cooperative Particle Swarm Optimisers", Genetic and Evolutionary Computation Conference, San Francisco, USA, 2001.
14. Lovbjerg, M., Krink, T., "Extending Particle Swarm Optimisers with Self-Organized Criticality", Proceedings of Fourth Congress on Evolutionary Computation, 2002, vol. 2, pp. 1588-1593.
15. Xiao-Feng Xie, Wen-Jun Zhang, Zhi-Lian Yang, "Hybrid Particle Swarm Optimizer with Mass Extinction", International Conf. on Communication, Circuits and Systems (ICCCAS), Chengdu, China, 2002.
16. Xiao-Feng Xie, Wen-Jun Zhang, Zhi-Lian Yang, "A Dissipative Particle Swarm Optimization", IEEE Congress on Evolutionary Computation, Honolulu, Hawaii, USA, 2002.
17. Jacques Riget, Jakob S. Vesterstorm, "A Diversity-Guided Particle Swarm Optimizer - The ARPSO", EVALife Technical Report no. 2002-02.

Revisiting Elitism in Ant Colony Optimization

Tony White, Simon Kaegi, and Terri Oda

School of Computer Science, Carleton University
1125 Colonel By Drive, Ottawa, Ontario, Canada K1S 5B6
arpwhite@scs.carleton.ca, skaegi@sympatico.ca, terri@zone12.com

Abstract. Ant Colony Optimization (ACO) has been applied successfully in solving the Traveling Salesman Problem. Marco Dorigo et al. used Ant System (AS) to explore the Symmetric Traveling Salesman Problem and found that the use of a small number of elitist ants can improve algorithm performance. The elitist ants take advantage of global knowledge of the best tour found to date and reinforce this tour with pheromone in order to focus future searches more effectively. This paper discusses an alternative approach where only local information is used to reinforce good tours thereby enhancing the ability of the algorithm for multiprocessor or actual network implementation. In the model proposed, the ants are endowed with a memory of their best tour to date. The ants then reinforce this “local best tour” with pheromone during an iteration to mimic the search focusing of the elitist ants. The environment used to simulate this model is described and compared with Ant System.

Keywords: Heuristic Search, Ant Algorithm, Ant Colony Optimization, Ant System, Traveling Salesman Problem.

1 Introduction

Ant algorithms (also known as Ant Colony Optimization) are a class of heuristic search algorithms that have been successfully applied to solving NP hard problems [1]. Ant algorithms are biologically inspired from the behavior of colonies of real ants, and in particular how they forage for food. One of the main ideas behind this approach is that the ants can communicate with one another through indirect means by making modifications to the concentration of highly volatile chemicals called pheromones in their immediate environment.

The Traveling Salesman Problem (TSP) is an NP complete problem addressed by the optimization community having been the target of considerable research [7]. The TSP is recognized as an easily understood, hard optimization problem of finding the shortest circuit of a set of cities starting from one city, visiting each other city exactly once, and returning to the start city again. Formally, the TSP is the problem of finding the shortest Hamiltonian circuit of a set of nodes. There are two classes of TSP problem: symmetric TSP, and asymmetric TSP (ATSP). The difference between the

two classes is that with symmetric TSP the distance between two cities is the same regardless of the direction you travel, with ATSP this is not necessarily the case.

Ant Colony Optimization has been successfully applied to both classes of TSP with good, and often excellent, results. The ACO algorithm skeleton for TSP is as follows [7]:

```
procedure ACO algorithm for TSPs
  Set parameters, initialize pheromone trails
  while (termination condition not met) do
    ConstructSolutions
    ApplyLocalSearch % optional
    UpdateTrails
  end
end ACO algorithm for TSPs
```

The earliest implementation, Ant System, was applied to the symmetric TSP problem initially and as this paper presents a proposed improvement to Ant System this is where we will focus our efforts.

While the ant foraging behaviour on which the Ant System is based has no central control or global information on which to draw, the use of global best information in the Elitest form of the Ant System represents a significant departure from the purely distributed nature of ant-based foraging. Use of global information presents a significant barrier to *fully* distributed implementations of Ant System algorithms in a live network, for example. This observation motivates the development of a fully distributed algorithm – the Ant System Local Best Tour (AS-LBT) – described in this paper. As the results demonstrate, it also has the by-product of having superior performance when compared to the Elitest form of the Ant System (AS-E). It also has fewer defining parameters.

The remainder of this paper consists of 5 sections. The next section provides further detail for the algorithm shown above. The Ant System Local Best Tour (AS-LBT) algorithm is then introduced and the experimental setup for its evaluation described. An analysis section follows, and the paper concludes with an evaluation of the algorithm with proposals for future work.

2 Ant System (AS)

Ant System was the earliest implementation of Ant Colony Optimization meta heuristic. The implementation is built on top of the ACO algorithm skeleton shown above. A brief description of the algorithm follows. For a comprehensive description of the algorithm, see [1, 2, 3 or 7].

2.1 Algorithm

Expanding upon the algorithm above, an ACO consists of two main sections: *initialization* and a *main loop*. The main loop runs for a user-defined number of iterations. These are described below:

Initialization

- Any initial parameters are loaded.
- Each of the roads is set with an initial pheromone value.
- Each ant is individually placed on a random city.

Main Loop Begins

Construct Solution

- Each ant constructs a tour by successively applying the probabilistic choice function and randomly selecting a city it has not yet visited until each city has been visited exactly once.

$$p_{ij}^k(t) = \frac{[\tau_{ij}(t)]^\alpha \cdot [\eta_{ij}]^\beta}{\sum_{l \in N_i^k} [\tau_{il}(t)]^\alpha \cdot [\eta_{il}]^\beta}$$

- The probabilistic function, $p_{ij}^k(t)$, is designed to favor the selection of a road that has a high pheromone value, τ , and high visibility value, η , which is given by: $1/d_{ij}$, where d_{ij} is the distance to the city. The pheromone scaling factor, α , and visibility scaling factor, β , are parameters used to tune the relative importance of pheromone and road length in selecting the next city.

Apply Local Search

- Not used in Ant System, but is used in several variations of the TSP problem where 2-opt or 3-opt local optimizers [7] are used.

Best Tour Check

- For each ant, calculate the length of the ant's tour and compare to the best tour's length. If there is an improvement, update it.

Update Trails

- Evaporate a fixed proportion of the pheromone on each road.
- For each ant perform the "ant-cycle" pheromone update.
- Reinforce the best tour with a set number of "elitist ants" performing the "ant-cycle" pheromone update.

In the original investigation of Ant System algorithms, there were three versions of Ant System that differed in how and when they laid pheromone. The "Ant-density" heuristic updates the pheromone on a road traveled with a fixed amount after every step. The "Ant-quantity" heuristic updates the pheromone on a road traveled with an amount proportional to the inverse of the length of the road after every step. Finally,

the “Ant-cycle” heuristic first completes the tour and then updates each road used with an amount proportional to the inverse of the total length of the tour.

Of the three approaches “Ant-cycle” was found to produce the best results and subsequently receives the most attention. It will be used for the remainder of this paper.

2.2 Discussion

Ant System in general has been identified as having several good properties related to directed exploration of the problem space without getting trapped in local minima [1]. The initial form of AS did not make use of elitist ants and did not direct the search as well as it might. This observation was confirmed in our experimentation performed as a control and used to verify the correctness of our implementation.

The addition of elitist ants was found to improve ant capabilities for finding better tours in fewer iterations of the algorithm, by highlighting the best tour. However, by using elitist ants to reinforce the best tour the problem now takes advantage of global data with the additional problem of deciding on how many elitist ants to use. If too many elitist ants are used the algorithm can easily become trapped in local minima [1, 3]. This represents the dilemma of exploitation versus exploration that is present in most optimization algorithms.

There have been a number of improvements to the original Ant System algorithm. They have focused on two main areas of improvement [7]. First, they more strongly exploit the globally best solution found. Second, they make use of a fast local search algorithm like 2-opt, 3-opt, or the Lin-Kernighan heuristic to improve the solutions found by the ants.

The algorithm improvements to Ant System have produced some of the highest quality solutions when applied to the TSP and other NP complete (or NP hard) problems [1].

As described in section 2.1, augmenting AS with a local search facility would be straightforward; however, it is not considered here. The area of improvement proposed in this paper is to explore an alternative to using the globally best tour (GBT) to reinforce and focus on good areas of the search space. The Ant System Local Best Tour algorithm is described in the next section.

3 Ant System Local Best Tour (AS-LBT)

The use of an elitist ant in Ant System exposes the need for a global observer to watch over the problem and identify what the best tour found to date is on a per iteration basis. As such, it represents a significant departure from the purely distributed AS algorithm.

The idea behind the design of AS-LBT is specifically to remove this notion of a global observer from the problem. Instead, each individual ant keeps track of the best tour it has found to date and uses it in place of the elitist ant tour to reinforce tour goodness.

It is as if the scale of the problem has been brought down to the ant level and each ant is running its individual copy of the Ant System algorithm using a single elitist ant. Remarkably, the ants work together effectively even if indirectly and the net effect is very similar to that of using the pheromone search focusing of the elitist ant approach. In fact, AS-E and AS-LBT can be thought of as extreme forms of a Particle Swarm algorithm. In Particle Swarm Optimization (PSO), particles (effectively equivalent to ants in ACO) have their search process moderated by both local and global best solutions.

3.1 Algorithm

The algorithm used is identical to that described for Ant System with the replacement of the elitist ant step with the ant's local best tour step. Referring, once again, to the algorithm described in section 2.1, the following changes are made:

That is, where the elitist ant step was:

- Reinforce the best tour with a set number of “elitist ants” performing the “ant-cycle” pheromone update.

For Local Best Tour we now do the following:

- For each ant perform the “ant-cycle” pheromone update using its local best tour.

The rest of the Ant System algorithm is unchanged, including the newly explored tour's “ant-cycle” pheromone update.

3.2 Experimentation and Results

For the purposes of demonstrating AS-LBT we constructed an Ant System simulation and applied it to a series of TSP Problems from the TSPLIB95 collection [6]. Three symmetric TSP problems were studied: eil51, eil76 and kro101. The eil51 problem is a 51-city TSP instance set up in a 2 dimensional Euclidean plane for which the optimal tour is known. The weight assigned to each road comes from the linear distance separating each pair of cities. The problems eil76 and kro101 represent symmetric TSP problems of 76 and 101 cities respectively.

The simulation created for this paper was able to emulate the behavior of the original Ant System (AS), Ant System with elitist ants (AS-E), and finally Ant System using the local best tour (AS-LBT) approach described in section 2.

3.2.1 Parameters and Settings

Ant System requires you to make a number of parameter selections. These parameters are:

Pheromone sensitivity (α) = 1	Pheromone additive constant
Visibility sensitivity (β) = 5	Number of ants
Pheromone decay rate (ρ) = 0.5	Number of elitist ants
Initial pheromone (τ_0) = 10^{-6}	

In his original work on Ant System Marco Dorigo performed considerable experimentation to tune and find appropriate values for a number of these parameters [3]. The values Dorigo found that provide for the best performance when averaged over the problems he studied were used in our experiments. These best-practice values are shown in the list above.

For those parameters that depend on the size of the problem our simulation made an effort to select good values based on knowledge of the problem and number of cities. Recent work [5] on improved algorithm parameters was unavailable to us when developing the LBT algorithm. We intend to explore the performance of the new parameters settings and will report the results in a future communication.

The Pheromone additive constant (Q) was eliminated altogether as a parameter by replacing it with the global best tour (GBT) length in the case of standard Ant System and the local best tour (LBT) length for the approach in this paper. We justify this decision by noting that Dorigo found that differences in the value of Q only weakly affected the performance of the algorithm and a value within an order of magnitude of the optimal tour length was acceptable. This means that the pheromone addition on an edge becomes:

$$\frac{L_{best}}{L_{ant}} \quad \text{For a normal "ant-cycle" pheromone update}$$

$$\frac{L_{best}}{L_{best}} = 1 \quad \text{For an elitist or LBT "ant-cycle" pheromone update}$$

The key factor in the pheromone update is that it remains inversely proportional to the length of the tour and this still holds with our approach. The ants now are not tied to a particular value of Q in the event of a change in the number of cities in the problem. We consider the removal of a user-defined parameter another attractive feature of the LBT algorithm and a contribution of the research reported here.

For the number of ants, we set this equal to the number of cities, as this seems to be a reasonable selection according to the current literature [1, 3, 7].

For the number of elitist ants we tried various values dependent on the size of the problem and used a value of $1/6^{\text{th}}$ of the number of cities for the results reported in this paper. This value worked well for the relatively low number of cities we used in our experimentation but for larger problems this value might need to be tuned, possibly using the techniques used in [5]. The current literature is unclear on the best value of the number of elitest ants to be used.

With AS-LBT, all ants perform the LBT "ant-cycle" update so subsequently the number of elitist ants is not needed. We consider the removal of the requirement to specify a value for the number of elitest ants an advantage. Hereafter, we refer to AS with elitest ants as AS-E.

3.2.2 Results

Using the parameters from the previous section, we performed 100 experiments for eil51, eil76 and kro101; the results are shown in Figures 1, 2 and 3 respectively. In the case of eil51 and eil76, 2000 iterations of each algorithm were performed, whereas

3500 iterations were used for kro101. The results of the experimentation showed considerable promise for AS-LBT. While experiments for basic AS were performed, they are not reported in detail here as they were simply undertaken in order to validate the code written for AS-E and AS-LBT.

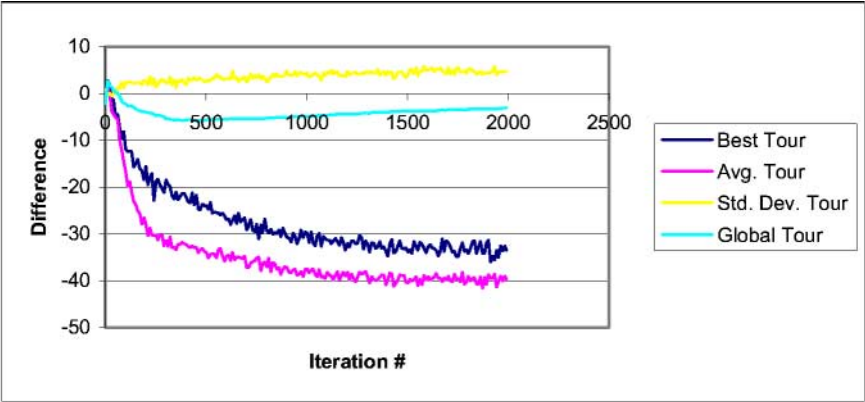


Fig. 1. Difference between LBT and Elitest Algorithms (eil51)

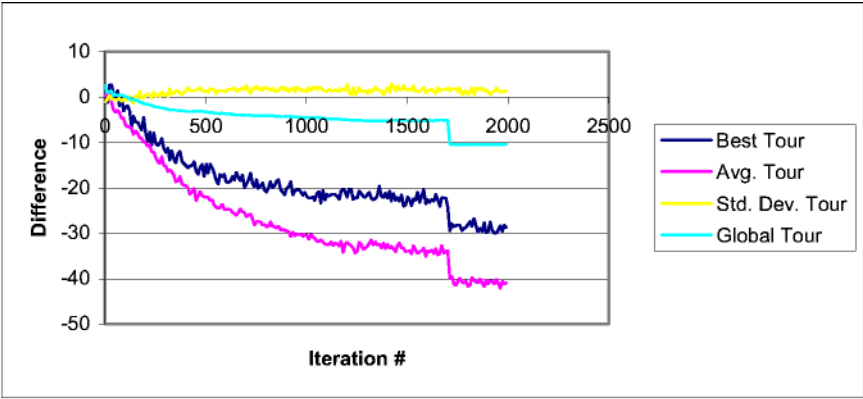


Fig. 2. Difference between LBT and Elitest Algorithms (eil76)

Figures 1, 2 and 3, each containing 4 curves, require some explanation. Each curve in each figure is the difference between the AS-LBT and AS-E per-iteration average of the 100 experiments performed. Specifically, the “Best Tour” curve represents the difference in the average best tour per iteration between AS-LBT and AS-E. The “Avg. Tour” curve represents the difference in the average tour per iteration between AS-LBT and AS-E. The “Std. Dev. Tour” curve represents the difference in the standard deviation of all tours per iteration between AS-LBT and AS-

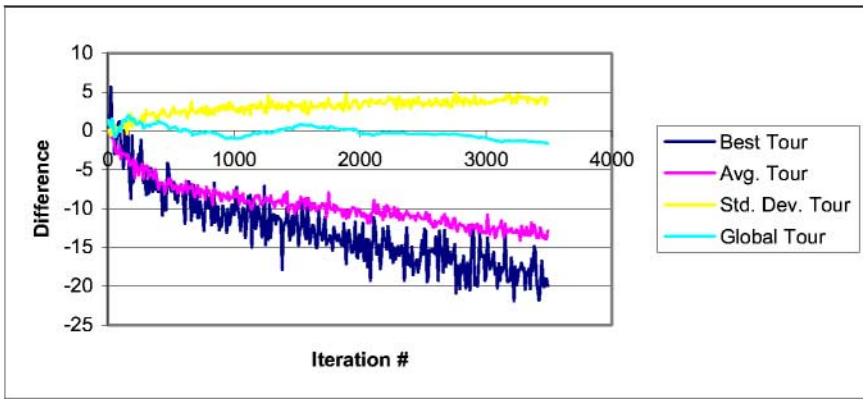


Fig. 3. Difference between LBT and Elitist Algorithms (kro101)

E. Finally, the “Global Tour” curve represents the difference in the best tour found per iteration between AS-LBT and AS-E. As the TSP is a minimization problem, negative difference values indicate superior performance for AS-LBT. The most important measure is the “Global Tour” measure, at least at the end of the experiment. This information is summarized in Table 1, below.

Table 1. Difference in Results for AS-LBT and AS-E

	Best Tour	Average Tour	Std. Dev Tour	Global Tour
eil51	-33.56	-39.74	4.91	-3.00
eil76	-29.65	-41.25	1.08	-10.48
Kro101	-19.97	-12.86	3.99	-1.58

The results in Table 1 clearly indicate the superior nature of the AS-LBT algorithm. The “Global Tour” is superior, on average, in all 3 TSP problems at the end of the experiment. The difference between AS-E and AS-LBT is significant for all 3 problems for a t-test with an α value of 0.05. Similarly, the “Best Tour” and “Average Tour” are also better, on average, for AS-LBT. The results for eil76 are particularly impressive, owing much of their success to the ability of AS-LBT to find superior solutions at approximately 1710 iterations.

The one statistic that is higher for AS-LBT is the average standard deviation of tour length on a per-iteration basis. This, too, is an advantage for the algorithm in that it means that there is still considerable diversity in the population of tours being explored. It is, therefore, more effective at avoiding local optima.

4 Analysis

Best Tour Analysis: As has been shown in the Results section, AS-LBT is superior to the AS-E approach as measured by the best tour found. In this section we take a comparative look at the evolution of the best tour in all three systems and then a look at the evolution of the best tour found per iteration.

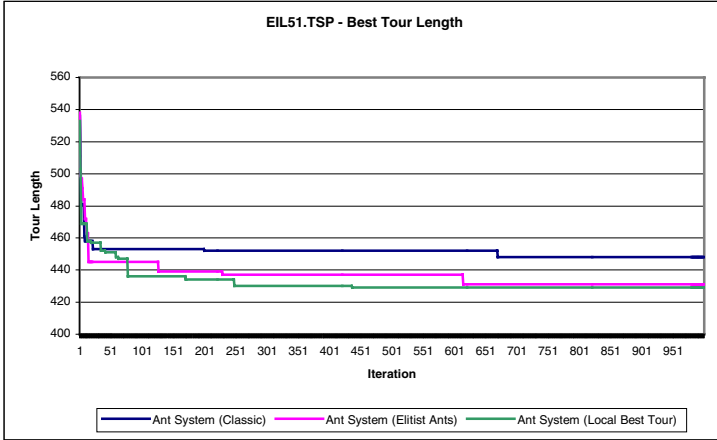


Fig. 4. Evolution of Best Tour Length

In Figure 4, which represents a single typical experiment, we can see the key difference between AS-E and AS-LBT. Whereas AS-E quickly finds a few good results, holds steady and then improves in relatively large pronounced steps, AS-LBT improves more gradually at the beginning but continues its downward movement at a steadier rate. In fact, if one looks closely at the graph one can see that even the classical AS system has found a better result during the early stages of the simulation when compared to AS-LBT. However, by about iteration 75, AS-LBT has overtaken the other two approaches and continues to gradually make improvements and maintains its overall improvement until the end of the experiment. This is confirmed in Figure 1, which is the average performance of AS-LBT for *eil51* over 100 experiments.

Overall, the behavior of AS-LBT could be described as slower but steadier. It takes slightly longer at the beginning to focus pheromone on good tours but after it has, it improves more frequently and steadily and on average will overtake the other two approaches given enough time. Clearly this hypothesis is supported by experimentation with the *eil76* and *kro101* TSP problem datasets as shown in Figures 2 and 3.

Average Tour Analysis: In the Best Tour Analysis we saw that there was a tendency for the AS-LBT algorithm to gradually improve in many small steps. With our analysis of the average tour we want to confirm that the relatively high deviation of ant

algorithms is working in the average case meaning that we are continuing to explore the problem space effectively. In this section we look at the average tour length per iteration to see if we can identify any behavioural trends.

In Figure 5 we see a very similar situation to that of the Best Tour Length per Iteration. The AS-LBT algorithm is on average exploring much closer to the optimal solution. Perhaps more importantly, the AS-LBT graph trend line is behaving very similarly in terms of its deviation as that with the other two systems. This suggests that the AS-LBT system is working as expected and is in fact searching in a better-focused fashion closer to the optimal solution.

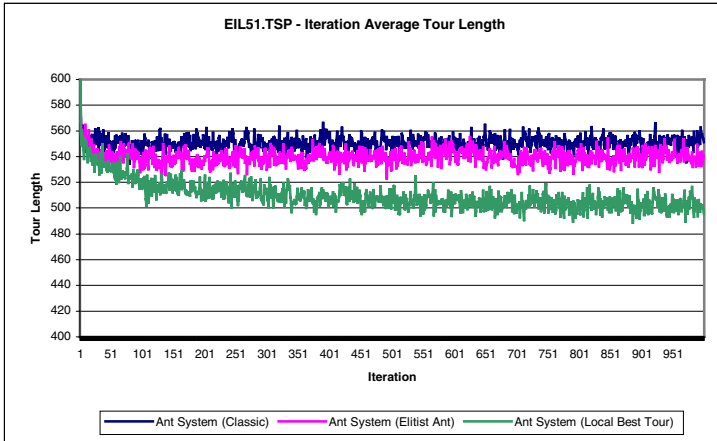


Fig. 5. Average Tour Length for Individual Iterations

Evolution of the Local Best Tour: The Local Best Tour approach is certainly very similar to the notion of elitist ants; only it is applied at the local level instead of at the global level. In this section we look at the evolution of the local best tour in terms of the average and worst tours, and compare them with the global best tour used by elitist ants.

From Figure 6 we can see that over time both the average and worst LBTs approach the value of global best tour. In fact the average in this simulation is virtually the same as the global best tour. From this figure, it is clear that the longer the simulation runs the closer the LBT “ant-cycle” pheromone update becomes to that of an elitist ant’s update scheme.

5 Discussion and Future Work

Through the results and analysis shown in this paper, Local Best Tour has proven to be an effective alternative to the use of the globally best tour for focusing ant search through pheromone reinforcement. In particular, the results show that AS-LBT has

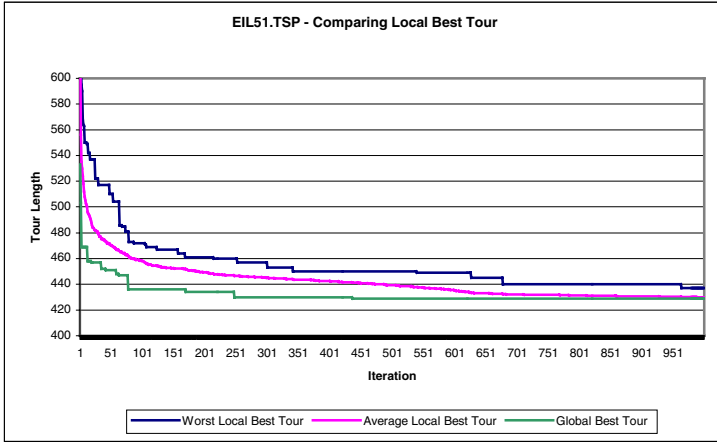


Fig. 6. Evolution of the Local Best Tour

excellent average performance characteristics. By removing the need for the global information required for AS-E, we have improved the ease with which a parallel or live network implementation can be achieved; i.e. a completely distributed implementation of the TSP is possible.

Analysis of the best tour construction process shows that AS-LBT, while initially converging more slowly than AS-E, is very consistent at incrementally building a better tour and on average will overtake the AS-E approach early in the search of the problem space.

Average and best iteration tour analysis has shown that AS-LBT shares the same variability characteristics of the original Ant System that make it resistant to getting stuck in local minima. Furthermore, AS-LBT is very effective in focusing its search towards the optimal solution.

Finally, AS-LBT follows in the notion that the use of best tours to better focus an ant's search is an effective optimization. The emergent behaviour of a set of autonomous LBT ants is to, in effect, become elitist ants over time.

As described earlier in this paper, a relatively straightforward way to further improve the performance of AS-LBT would be to add a fast local search algorithm like 2-opt, 3-opt or the Lin Kernighan heuristic. Alternatively, the integration of recent network transformation algorithms [4] should prove useful as local search operators.

Finally, future work should include the application of the LBT algorithm to other problems such as: the asymmetric TSP, the Quadratic Assignment Problem (QAP), the Vehicle Routing Problem (VRP) and other problems to which ACO has been applied [1].

6 Conclusions

This paper has demonstrated that an ACO algorithm using only local information can be applied to the TSP. The AS-LBT algorithm is truly distributed and is characterized by fewer parameters when compared to AS-E. Considerable experimentation has demonstrated that *significant* improvements are possible for 3 TSP problems. We believe that AS-LBT with the improvements outlined in the previous section will further enhance our confidence in the hypothesis and look forward to reporting on these improvements in a future research paper. Finally, we believe that a Particle Swarm Optimization algorithm, where search is guided by both local best tour and global best tour terms may yield further improvements in performance for ACO algorithms.

References

1. Bonabeau E., Dorigo M., and Theraulaz G. Swarm Intelligence From Natural to Artificial Systems. Oxford University Press, New York NY, 1999.
2. Dorigo M. and L.M. Gambardella. Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem. IEEE Transactions on Evolutionary Computation, 1(1):53–66, 1997.
3. Dorigo M., V. Maniezzo and A. Coloni. The Ant System: Optimization by a Colony of Cooperating Agents. IEEE Transactions on Systems, Man, and Cybernetics-Part B, 26(1):29–41, 1996.
4. Dumitrescu A. and Mitchell J., Approximation Algorithms for Geometric Optimization Problems, in the Proceedings of the 9th Canadian Conference on Computational Geometry, Queen's University, Kingston, Canada, August 11–14, 1997, pp. 229–232.
5. Pilat M. and White T., Using Genetic Algorithms to optimize ACS-TSP. In Proceedings of the 3rd International Workshop on Ant Algorithms, Brussels, Belgium, September 12–14 2002.
6. Reinelt G. TSPLIB, A Traveling Salesman Problem Library. ORSA Journal on Computing, 3:376–384, 1991.
7. Stützle T. and Dorigo M. ACO Algorithms for the Traveling Salesman Problem. In K. Miettinen, M. Makela, P. Neittaanmaki, J. Periaux, editors, Evolutionary Algorithms in Engineering and Computer Science, Wiley, 1999.

A New Approach to Improve Particle Swarm Optimization

Liping Zhang, Huanjun Yu, and Shangxu Hu

College of Material and Chemical Engineering, Zhejiang University,
Hangzhou 310027, P.R. China
zhanglp@infotech.zju.edu.cn
yuhj@zjuem.zju.edu.cn
sxhu@mail.hz.zj.cn

Abstract. Particle swarm optimization (PSO) is a new evolutionary computation technique. Although PSO algorithm possesses many attractive properties, the methods of selecting inertia weight need to be further investigated. Under this consideration, the inertia weight employing random number uniformly distributed in $[0,1]$ was introduced to improve the performance of PSO algorithm in this work. Three benchmark functions were used to test the new method. The results were presented to show that the new method is effective.

1 Introduction

Particle swarm optimization (PSO) is an evolutionary computation technique introduced by Kennedy and Eberhart in 1995[1-3]. The underlying motivation for the development of PSO algorithm was social behavior of animals such as bird flocking, fish schooling, and swarm [4]. Initial simulations were modified to incorporate nearest-neighbor velocity matching, eliminate ancillary variable, and acceleration in movement. PSO is similar to genetic algorithm (GA) in that the system is initialized with a population of random solutions. However, in PSO, each individual of the population, called particle, has an adaptable velocity, according to which it moves over the search space. Each particle keeps track of its coordinate in hyperspace, which are associated with the solution (fitness) it has achieved so far. This value is called *pbest*. Another “best” value is called *gbest* that is obtained so far by any particle in the population and stored the overall best value.

Suppose that the search space is D -dimensional, then the i -th particle of the swarm can be represented by a D -dimensional vector, $X_i=(x_{i1}, x_{i2},...,x_{iD})$. The velocity of this particle, can be represented by another D -dimensional vector $V_i=(v_{i1}, v_{i2},...,v_{iD})$. The best previously visited position of the i -th particle is denoted as $P_i=(p_{i1}, p_{i2},...,p_{iD})$. Defining g as the index of the best particle in the swarm, then the velocity of particle and its new position will be assigned according to the following two equations:

$$v_{id} = v_{id} + c_1 r_1 (p_{id} - x_{id}) + c_2 r_2 (p_{gd} - x_{id}) \quad (1)$$

$$x_{id} = x_{id} + v_{id} \quad (2)$$

where c_1 and c_2 are positive constant, called acceleration, and r_1 and r_2 are two random numbers, uniformly distributed in $[0,1]$.

Velocities of particles on each dimension are clamped by a maximum velocity V_{max} . If the sum of accelerations would cause the velocity on that dimension to exceed V_{max} , which is a parameter specified by the user, then the velocity on that dimension is limited to V_{max} . V_{max} influences PSO performance sensitively. A larger V_{max} facilitates global exploration, while a smaller V_{max} encourages local exploitation [5].

The PSO algorithm is still far from mature, many authors have modified the original version. Firstly, in order to better control exploration, an inertia weight in the PSO algorithm was first introduced in 1998 [6]. Recently, for insuring convergence, Clerc proposed the use of a constriction factor in the PSO [7]. Equation (3), (4), and (5) describes the modified algorithm.

$$v_{id} = \chi(wv_{id} + c_1r_1(p_{id} - x_{id}) + c_2r_2(p_{gd} - x_{id})) \quad (3)$$

$$x_{id} = x_{id} + v_{id} \quad (4)$$

$$\chi = \frac{2}{\left| 2 - \varphi - \sqrt{\varphi^2 - 4\varphi} \right|} \quad (5)$$

where w is the inertia weight, and χ is a constriction factor, and $\varphi = c_1 + c_2, \varphi > 4$.

The use of the inertia weight for controlling the velocity has resulted in high efficiency for PSO. Suitable selection of the inertia weight provides a balance between global and local explorations. The performance of PSO using an inertia weight was compared with performance using a constriction factor [8], and Eberhart et al. concluded that best approach is to use the constriction factor while limiting the maximum velocity V_{max} to the dynamic range of the variable X_{max} on each dimension. For example, $V_{max} = X_{max}$.

In this work, we proposed a method using random number inertia weight called RNM to improve the performance of PSO.

2 The Ways to Determine the Inertia Weight

As mentioned precedingly, the inertia weight was found to be an important parameter to PSO algorithms. However, the determination of inertia weight is still an unsolved problem. Shi et al. provided methods to determine the inertia weight. In their earlier work, inertia weight was set as constant [6]. By setting maximum velocity to be 2.0, it was found that PSO with an inertia weight in the range $[0.9, 1.2]$ on average has a better performance. In a later work, inertia weight was set to be continuously decreased linearly during run [9]. Still later, a time decreasing inertia weight from 0.9 to 0.4 was found to be better than a fixed inertia weight. The linearly decreasing inertia

weight (LDW) was used by many authors so far [10-12]. Recently another approach was suggested to use a fuzzy variable to adapt the inertia weight [12,13]. The results reported in their papers showed that the performance of PSO can be significantly improved. However, it is relatively complicated.

The right side of equation (1) consists of three parts: the first part is the previous velocity of the particle; the second and third parts are contributing to the change of the velocity of a particle. Shi and Eberhart concluded that the role of the inertia weight w is considered to be crucial for the convergence of PSO [6]. A larger inertia weight facilitates global exploration (searching new areas), while a smaller one tends to facilitate local exploitation. A general rule of thumb suggests that it is better to initially set the inertia weight to a larger value, and gradually decrease it. Unfortunately, the phenomenon that the global search ability is decreasing when inertia weight is decreasing to zero indicates that inertia weight may exit some unclear mechanism [14]. However, the deceased inertia weight is subject to trap the algorithms into the local optima and slows the convergence speed when it is near a minimum. Under this consideration, many cases were tested, and we finally set the inertia weight as random numbers uniformly distributed in $[0,1]$, which is more capable of escaping from the local optima than LDW, therefore better results were obtained. Our motivation is that local exploitation combining with global exploration can be processing parallel. The new version is:

$$v_{id} = r_0 v_{id} + c_1 r_1 (p_{id} - x_{id}) + c_2 r_2 (p_{gd} - x_{id}) \quad (6)$$

where r_0 is a random number uniformly distributed in $[0,1]$, and the other parameters are same as before.

Our method can overcome two drawbacks of LDW. For one thing, decreasing the dependence of inertial weight on the maximum iteration that is difficultly predicted before experiments. Another is avoiding the lacks of local search ability at early of run and global search ability at the end of run.

3 Experimental Studies

In order to test the influence of inertia weight on the PSO performance, three non-linear benchmark functions reported in literature [15,16] were used since they are well known problems. The first function is the Rosenbrock function:

$$f_1(x) = \sum_{i=1}^n (100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2) \quad (7)$$

where $\mathbf{x}=[x_1, x_2, \dots, x_n]$ is an n -dimensional real-valued vector.

The second is the generalized Rastrigrin function:

$$f_2(x) = \sum_{i=1}^n (x_i^2 - 10 \cos(2\pi x_i) + 10) \quad (8)$$

The third is the generalized Griewank function:

$$f_3(x) = \frac{1}{4000} \sum_{i=1}^n x_i^2 - \prod_{i=1}^n \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1 \quad (9)$$

Three different amounts dimensions were tested: 10, 20 and 30. The maximum numbers of generations were set as 1000, 1500 and 2000 corresponding to the dimensions 10, 20 and 30, respectively. For investigation the scalability of PSO algorithm, three population sizes 20, 40 and 80 were used for each function with respect to different dimensions. Acceleration constants took the values $c_1=c_2=2$. Constriction factor $\chi=1$. For the purpose of comparison, all the V_{max} and X_{max} were assigned by same parameter settings as in literature [13] and listed in table 1. 500 trial runs were taken for each case.

Table 1. X_{max} and V_{max} values used for tests

Function	X_{max}	V_{max}
f_1	100	100
f_2	10	10
f_3	600	600

4 Results and Discussions

Table 2, 3 and 4 listed the mean best fitness value of the best particle found for the Rosenbrock, Rastrigrin, and Griewank function with two inertia weight selecting methods, LDW and RNW respectively.

Table 2. Mean best fitness value for the Rosenbrock function

Population Size	No. of Dimensions	No. of Generations	LDW Method	RNW Method
20	10	1000	106.63370	65.28474
	20	1500	180.17030	147.52372
	30	2000	458.28375	409.23443
40	10	1000	61.36835	41.32016
	20	1500	171.98795	95.48422
	30	2000	289.19094	253.81490
80	10	1000	47.91896	20.77741
	20	1500	104.10301	82.75467
	30	2000	176.87379	156.00258

By comparing the results of two methods, it is clearly to see that the performance of PSO can be improved with random number inertia weight for Rastrigrin and Ro-

senbrock function, while for the Griewank function, results of two methods are comparable.

Table 3. Mean best fitness value for the Rastrigrin function

Population Size	No. of Dimensions	No. of Generations	LDW Method	RNW Method
20	10	1000	5.25230	5.04258
	20	1500	22.92156	20.31109
	30	2000	49.21827	42.58132
40	10	1000	3.56574	3.22549
	20	1500	17.74121	13.84807
	30	2000	38.06483	32.15635
80	10	1000	2.37332	1.85928
	20	1500	13.11258	9.95006
	30	2000	30.19545	25.44122

Table 4. Mean best fitness value for the Griewank function

Population Size	No. of Dimensions	No. of Generations	LDW Method	RNW Method
20	10	1000	0.09620	0.09926
	20	1500	0.03000	0.03678
	30	2000	0.01674	0.02007
40	10	1000	0.08696	0.07937
	20	1500	0.03418	0.03014
	30	2000	0.01681	0.01743
80	10	1000	0.07154	0.06835
	20	1500	0.02834	0.02874
	30	2000	0.01593	0.01718

5 Conclusions

In this work, the performance of the PSO algorithm with random number inertia weight has been extensively investigated by experimental studies of three non-linear functions. Because local exploitation combining with global exploration can be processing parallel, random number inertia weight (RNW) method can obtain better results than linearly decreasing inertia weight (LDW) method. Lacks of local search ability at early stage of run and global search ability at the end of run using linearly decreasing inertia weight method were overcome. However, only three benchmark problems had been tested. To fully claim the benefits of the random number inertia weight to PSO algorithm, more problems need to be tested.

References

1. J. Kennedy and R. C. Eberhart. Particle swarm optimization. Proc. IEEE Int. Conf. on Neural Networks (1995) 1942–1948
2. R. C. Eberhart and J. Kennedy. A new optimizer using particle swarm theory. Proceedings of the Sixth International Symposium on Micro Machine and Human Science. Nagoya, Japan (1995) 39–43
3. R. C. Eberhart, Simpson, P. K., and Dobbins, R. W. Computational Intelligence PC Tools. Boston, MA: Academic Press Professional (1996)
4. M. M. Millonas. Swarm, phase transition, and collective intelligence. In C.G. Langton, Eds., Artificial life III. Addison Wesley, MA (1994)
5. K. E. Parsopoulos and M. N. Vrahatis. Recent approaches to global optimization problems through particle swarm optimization. Natural Computing 1 (2002) 235–306
6. Y. Shi and R. Eberhart. A modified particle swarm optimizer. IEEE Int. Conf. on Evolutionary Computation (1997) 303–308
7. M. Clerc. The swarm and queen: towards a deterministic and adaptive particle swarm optimization. Proc. Congress on Evolutionary Computation, Washington, DC., Piscataway, NJ:IEEE Service Center (1999) 1951–1957
8. R. C. Eberhart and Y. Shi. Comparing Inertia weight and constriction factors in particle swarm optimization. In Proc. 2000 Congr. Evolutionary Computation, San Diego, CA (2000) 84–88
9. H. Yoshida, K. Kawata, Y. Fukuyama, and Y. Nakanishi. A particle swarm optimization for reactive power and voltage control considering voltage stability. In G. L. Torres and A. P. Alves da Silva, Eds., Proc. Int. Conf. on Intelligent System Application to Power Systems, Rio de Janeiro, Brazil (1999) 117–121
10. C. O. Ouyue, E. C. Biscaia, and J. J. Pinto. The use of particle swarm optimization for dynamical analysis in chemical processes. Computers and Chemical Engineering 26 (2002) 1783–1793
11. Y. Shi and R. Eberhart. Parameter selection in particle swarm optimization. Proc. 7th Annual Conf. on Evolutionary Programming (1998) 591–600
12. Y. Shi, and Eberhart, R. Experimental study of particle swarm optimization. Proc. SCI2000 Conference, Orlando, FL (2000)
13. Y. Shi and R. Eberhart. Fuzzy adaptive particle swarm optimization. 2001. Proceedings of the 2001 Congress on Evolutionary Computation, vol. 1 (2001) 101–106
14. X. Xie, W. Zhang, and Z. Yang. A dissipative particle swarm optimization. Proceedings of the 2002 Congress on Evolutionary Computation, Volume: 2 (2002) 1456–1461
15. J. Kennedy. The particle swarm: social adaptation of knowledge. Proc. IEEE International Conference on Evolutionary Computation (Indianapolis, Indiana), IEEE Service Center, Piscataway, NJ (1997) 303–308
16. P. J. Angeline. Using selection to improve particle swarm optimization. IEEE International Conference on Evolutionary Computation, Anchorage, Alaska, May (1998) 4–9
17. J. Kennedy, R.C. Eberhart, and Y. Shi. Swarm Intelligence, San Francisco: Morgan Kaufmann Publishers (2001)

Clustering and Dynamic Data Visualization with Artificial Flying Insect

S. Aupeitit¹, N. Monmarché¹, M. Slimane¹, C. Guinot², and G. Venturini¹

¹ Laboratoire d'Informatique de l'Université de Tours,
École Polytechnique de l'Université de Tours - Département Informatique
64, Avenue Jean Portalis, 37200 Tours, France.
{monmarche,oliver,venturini}@univ-tours.fr
sebastien.aupeitit@etu.univ-tours.fr

² CE.R.I.E.S.,
20 rue Victor Noir, 92521 Neuilly sur Seine Cédex.
christiane.guinot@ceries-lab.com

Abstract. We present in this paper a new bio-inspired algorithm that dynamically creates and visualizes groups of data. This algorithm uses the concepts of flying insects that move together in complex manner with simple local rules. Each insect represents one datum. The insect moves aim at creating homogeneous groups of data that evolve together in a 2D environment in order to help the domain expert to understand the underlying class structure of the data set.

1 Introduction

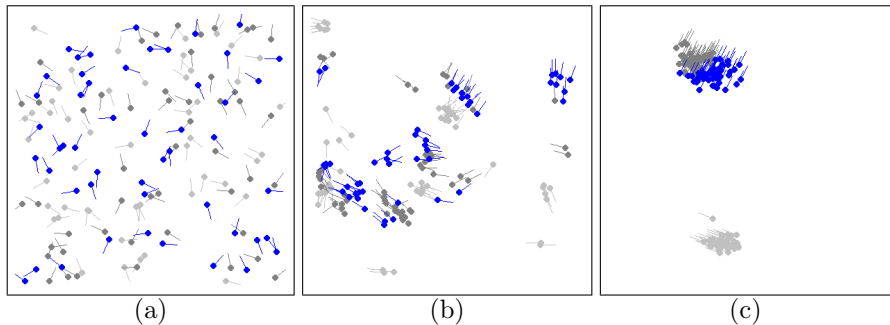
Many clustering algorithms are inspired from biology like genetic algorithms [1, 2] or artificial ant algorithms [3,4] for instance. The main advantages of these algorithms are that they are distributed and they generally do not need an initial partition of data as it can be often needed. This study takes its inspiration from different kinds of animals that use social behavior for their movement (clouds of insects, schooling fishes or bird flocks) that have not been applied and extensively tested on clustering problems yet. Models of these behaviors that can be found in literature are characterized by a “swarm intelligence” which consists in the appearance of macroscopic patterns obtained with simple entities obeying to simple local coordination rules [6,5].

2 Principle

In this work, we use the notion of flying insect/entity in order to treat dynamic visualization and data clustering problems. The main idea is to consider that insects represent data to cluster and that they move following local behavior rule in a way that, after few movements, homogeneous insect clusters appear and move together. Cluster visualization allow the domain expert to perceive

the partitioning of the data. Another algorithm can analyze these clusters and give precise classification as output.

An example can be observed in the following pictures :



where (a) corresponds to the initial step for 150 objects (Iris dataset), (b) and (c) are screen shots showing the dynamic formation of clusters.

3 Conclusion

This work has demonstrated that flying animals can be used to visualize data structure in a dynamic way. Future work will concerns an application of these principles to present results obtained by a search engine.

References

1. R. Cucchiara. Analysis and comparison of different genetic models for the clustering problem in image analysis. In R.F. Albrecht, C.R. Reeves, and N.C. Steele, editors, *International Conference on Artificial Neural Networks and Genetic Algorithms*, pages 423–427. Springer-Verlag, 1993.
2. D.R. Jones and M.A. Beltrano. Solving partitioning problems with genetic algorithms. In Belew and Booker, editors. *Fourth International Conference on Genetic Algorithms*. Morgan Kaufmann, San Mateo, CA, 1991., pages 442–449.
3. E.D. Lumer and B. Faieta. Diversity and adaptation in populations of clustering ants. In D. Cliff, P. Husbands, J.A. Meyer, and Stewart W., editors, *Proceedings of the Third International Conference on Simulation of Adaptive Behavior*, pages 501–508. MIT Press, Cambridge, Massachusetts, 1994.
4. N. Monmarché, M. Slimane, and G. Venturini. On improving clustering in numerical databases with artificial ants. In D. Floreano, J.D. Nicoud, and F. Mondala, editors, *5th European Conference on Artificial Life (ECAL'99), Lecture Notes in Artificial Intelligence*, volume 1674, pages 626–635, Swiss Federal Institute of Technology, Lausanne, Switzerland, 13-17 September 1999. Springer-Verlag.
5. G. Proctor and C. Winter. Information flocking: Data visualisation in virtual worlds using emergent behaviours. In J.-C. Heudin, editor, *Proc. 1st Int. Conf. Virtual Worlds, VW*, volume 1434, pages 168–176. Springer-Verlag, 1998.
6. C. W. Reynolds. Flocks, herds, and schools: A distributed behavioral model. *Computer Graphics (SIGGRAPH '87 Conference Proceedings)*, 21(4):25–34, 1987.

Ant Colony Programming for Approximation Problems^{*}

Mariusz Boryczka¹, Zbigniew J. Czech², and Wojciech Wieczorek¹

¹ University of Silesia, Sosnowiec, Poland, {boryczka,wieczor}@us.edu.pl

² University of Silesia, Sosnowiec and Silesia University of Technology, Gliwice, Poland, zjc@us.edu.pl

Abstract. A method of automatic programming, called genetic programming, assumes that the desired program is found by using a genetic algorithm. We propose an idea of ant colony programming in which instead of a genetic algorithm an ant colony algorithm is applied to search for the program. The test results demonstrate that the proposed idea can be used with success to solve the approximation problems.

1 Introduction

Approximation problems which consist in a choice of an optimum function from some class of functions are considered. While solving an approximation problem by ant colony programming the desired approximating function is built as a computer program, i.e. a sequence of assignment instructions which evaluates the function.

2 Ant Colony Programming for Approximation Problems

The ant colony programming system consists of: (a) the nodes of set N of graph $G = (N, E)$ which represent the assignment instructions out of which the desired program is built; the instructions comprise the terminal symbols, i.e. constants, input and output variables, temporary variables and functions; (b) the tabu list which holds the information about the path pursued in the graph; (c) the probability of moving ant k located in node r to node s in time t which is equal to:

$$p_{rs}^k(t) = \frac{\tau_{rs}(t) \cdot [\psi_s]^\beta}{\sum_{i \in J_k^t} [\tau_{ri}(t)] \cdot [\psi_i]^\beta}$$

Here $\psi_s = 1/e$, where e is an approximation error given by the program while expanded by the instruction represented by node $s \in N$.

^{*} This work was carried out under the State Committee for Scientific Research (KBN) grant no 7 T11C 021 21.

3 Test Results

The genetic (GP) and ant colony programming (ACP) methods to solve approximation problems were implemented and compared on the real-valued function of three variables:

$$t = (1 + x^{0.5} + y^{-1} + z^{-1.5})^2 \quad (1)$$

where $x, y, z \in [1.0, 6.0]$. The experiments were conducted in accordance to the learning model. Both methods were first run on a training set, T , of 216 data items, and then on a testing set, S , of 125 data items. The results of the

Table 1. (a) The Average Percentage Error, e_T , e_S , and the Standard Deviation, σ_T , σ_S , for the Training, T , and Testing, S , Data; (b) Comparison of Results

a)					Model/method	e_T	e_S
Method	e_T	σ_T	e_S	σ_S	GMDS model	4.70	5.70
					ACP (this work)	2.60	2.70
100 experiments, 15 min each					Fuzzy model 1	1.50	2.10
GP	1.86	1.00	2.15	1.35	GP (this work)	1.07	1.18
ACP	6.81	2.60	6.89	2.61	Fuzzy model 2	0.59	3.40
10 experiments, 1 hour each					FNN type 1	0.84	1.22
GP	1.07	0.58	1.18	0.60	FNN type 2	0.73	1.28
ACP	2.60	2.17	2.70	2.28	FNN type 3	0.63	1.25
					M-Delta	0.72	0.74
					Fuzzy INET	0.18	0.24
					Fuzzy VINET	0.08	0.18

experiments are summarized in Table 1. It can be seen (Table 1a) that the average percentage errors (e_T and e_S) for the ACP method are larger than those for the GP method. The range of this error for the training process and 100 experiments was 0.0007...9.9448 for the ACP method, and 0.0739...6.6089 for the GP method. The error 0.0007 corresponds to a perfect fit solution with respect to function (1). Such a solution was found 8 times in the series of 100 experiments by the ACP method, and was not found at all by the GP method. Table 1b compares our GP and ACP experimental results (for function (1)) with the results cited in the literature.

4 Conclusions

The idea of ant colony programming for solving approximation problems was proposed. The test results demonstrated that the method is effective. There are still some issues which remain to be investigated. The most important is the issue of establishing the set of instructions, N , which defines the solution space explored by the ACP method. On the one hand this set should be as small as possible so that the searching process is fast. On the other hand it should be large enough so that the large number of local minima, and hopefully the global minimum, are encountered.

Long-Term Competition for Light in Plant Simulation

Claude Lattaud

Artificial Intelligence Laboratory of Paris V University (LIAP5)
45, rue des Saints Pères
75006 Paris, France
Claude.Lattaud@math-info.univ-paris5.fr

Abstract. This paper presents simulations of long-term competition for light between two plant species, oaks and beeches. These artificial plants, evolving in a 3D environment, are based on a multi-agent model. Natural oaks and beeches develop two different strategies to exploit light. The model presented in this paper uses these properties during the plant growth. Most of the results are close to those obtained in natural conditions on long-term evolution of forests.

1 Introduction

The study of ecosystems is now deeply related to economic resources and their comprehension becomes an important field of research since the last century. P. Dansereau in [1] says that “An ecosystem is a limited space where resource recycling on one or several trophic levels is performed by a lot of evolving agents, using simultaneously and successively mutually compatible processes that generate long or short term usable products”. This paper tries to focus on one aspect of this co-evolution in the ecosystem, the competition for a resource between two plant species.

In nature, most of the plants compete for light. Photosynthesis being one of the main factors for plant growth, trees, in particular, tend to develop several strategies to optimize the quantity of light they receive. This study is based on the observation of a French forest composed mainly of oaks and beeches. In [2] B. Boullard says : “In the forest of Chauv [...] stands were, in 1824, composed of 9/10 of oaks and 1/10 of beeches. In 1964, proportions were reversed [...] Obviously, under the oak grove of temperate countries, the decrease of light can encourage the rise of beeches to the detriment of oaks, and slowly the beech grove replaces the oak grove”.

2 Plant Modeling

The plant model defined in this paper is based on multi-agent systems [3]. The main idea of this approach is to decentralize all the decisions and processes on several autonomous entities, the agents, able to communicate together, instead of on a unique super-entity. A plant is then determined by a set of agents, representing the plant organs, which allow the emergence of plant global behaviors by their cooperation.

Each of these organs have their own mineral and carbon storage with a capacity proportional to its volume. These storages stock plant resources and are used for its survival and its growth at each stage.

During each stage, an organ receives and stocks resources, directly from ground minerals or sunlight, or indirectly from other organs, and uses them for its survival, organic functions and development. The organ is then able to convert carbon and mineral resources in structural mass for the growth process or to distribute them to nearby organs.

The simulations presented in this paper focus on the light resource. Photosynthesis is the process by which the plants increase their carbon storage by converting light they receive from the sky. Each point of the foliage can receive light from the sky according to three directions in order to simulate a simple daily sun movement.

As simulations are performed on the long-term, a reproduction process has been developed. At each stage, if a plant reaches its sexual maturity, the foliage assigns a part of its resources to its seeds, then eventually spreads them in the environment.

All the plants are disposed in a virtual environment, defined as a particular agent, composed with the ground and the sky. The environment manages synchronously all the interactions between plants, like mineral extraction from the ground, competition for light and physical encumbrance.

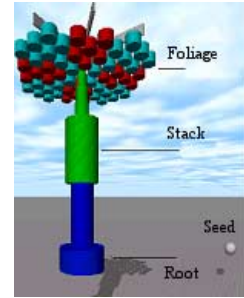


Fig. 1. Plant organs

3 Conclusion

Two sets of simulations were performed to understand the evolution of oak and beech populations. They exhibit a global behavior of plant communities close to that of those observed in nature : oaks competing for light against beeches slowly disappear. Artificial oaks develop a short-term strategy to exploit light, while artificial beeches tend to develop a long-term strategy. The main factor to be considered in this competition was the foliage and stack properties of virtual plants, but simulation showed that another unexpected phenomenon occurred. The competition for light did not only happen in altitude at the foliage level, but also on the ground where seeds grow. Shadow generated by plants played a capital role in the seed growth dynamics, especially in the seed sleeping phase. In this competition, beeches always outnumber oaks on the long-term.

References

1. Dansereau, P. : Repères «Pour une éthique de l'environnement avec une méditation sur la paix.» In Bélanger, R., Plourde S. (eds.) : Actualiser la morale: mélanges offerts à René Simon, Les Éditions Cerf, Paris (1992).
2. Boullard, B.: «Guerre et paix dans le règne végétal», Ed. Ellipse (1990).
3. Ferber, J., « Les systèmes multi-agents », Inter Editions, Paris (1995).

Using Ants to Attack a Classical Cipher

Matthew Russell, John A. Clark, and Susan Stepney

Department of Computer Science, University of York, York, YO10 5DD, U.K.
{matthew,jac,susan}@cs.york.ac.uk

1 Introduction

Transposition ciphers are a class of historical encryption algorithms based on rearranging units of plaintext according to some fixed permutation which acts as the secret key. Transpositions form a building block of modern ciphers, and applications of metaheuristic optimisation techniques to classical ciphers have preceded successful results on modern-day cryptological problems. In this paper we describe the use of Ant Colony Optimisation (ACO) for the automatic recovery of the key, and hence the plaintext, from only the ciphertext.

2 Cryptanalysis of Transposition Ciphers

The following simple example of a transposition encryption uses the key 31524:

31524 31524 31524 31524 31524
THEQU ICKBR OWNFO XJUMP EDXXX
⇒ HQTUE CBIRK WFOON JMXPJ DXEXX

Decryption is straightforward with the key, but without it the cryptanalyst has a *multiple anagramming* problem, namely rearranging columns to discover the plaintext:

H	Q	T	U	E	⇒	T	H	E	Q	U
C	B	I	R	K		I	C	K	B	R
W	F	O	O	N		O	W	N	F	O
J	M	X	P	U		X	J	U	M	P
D	X	E	X	X		E	D	X	X	X

Traditional cryptanalysis has proceeded by using a statistical heuristic for the likelihood of two columns being adjacent. Certain pairs of letters, or *bigrams*, occur more frequently than others. For example, in English, ‘TH’ is very common. Using some large sample of normal text an expected frequency for each bigram can be inferred. Two columns placed adjacently create several bigrams. The heuristic d_{ij} is defined as the sum of their probabilities; that is, for columns i and j , $d_{ij} = \sum_r P(i_r j_r)$, where i_r and j_r denote the r th letter in the column and $P(xy)$ is the standard probability for the bigram “ xy ”.

Maximising the sum of d_{ij} over a permutation of the columns can be enough to reconstruct the original key, and a simple greedy algorithm will often suffice.

However, the length of the ciphertext is critical as short ciphertexts have large statistical variation, and two separate problems eventually arise: (1) the greedy algorithm fails to find the global maximum, and, more seriously, (2) the global maximum does not correspond to the correct key.

In order to attempt cryptanalysis on shorter texts, a second heuristic can be employed, based on counting dictionary words in the plaintext, weighted by their length. This typically solves problem (2) for much shorter ciphertexts, but the fitness landscape it defines is somewhat discontinuous and difficult to search, while the original heuristic yields much useful, albeit noisy, information.

3 Ants for Cryptanalysis

A method has been found that successfully deals with problems (1) and (2), combining both heuristics using the ACO algorithm Ant System [2]. In the ACO algorithm, ants construct a solution by walking a graph with a distance matrix, reinforcing with pheromone arcs that correspond to better solutions. An ant's choice at each node is affected by both the distance measure and the amount of pheromone deposited in previous iterations.

For our cryptanalysis problem the graph nodes represent columns, and the distance measure used in the ants' choice of path is given by the d_{ij} bigram-based heuristic, essentially yielding a maximising Asymmetric Travelling Salesmen Problem. The update to the pheromone trails, however, is determined by the dictionary heuristic, not the usual sum of the bigram distances. Therefore both heuristics have influence on an ant's decision at a node: the bigram heuristic is used directly, and the dictionary heuristic provides feedback through pheromone.

In using ACO with these two complementary heuristics, we found that less ciphertext was required to completely recover the key, compared both to a greedy algorithm, and also to other metaheuristic search methods previously applied to transposition ciphers: genetic algorithms, simulated annealing and tabu search [4,3,1]. It must be noted that these earlier results make use of only bigram frequencies, without a dictionary word count, and they could conceivably be modified to use both heuristics. However, ACO provides an elegant way of combining the two heuristics.

References

1. Andrew Clark. *Optimisation Heuristics for Cryptology*. PhD thesis, Queensland University of Technology, 1998.
2. Marco Dorigo, Vittorio Maniezzo, and Alberto Coloni. The Ant System: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics Part B: Cybernetics*, 26(1):29–41, 1996.
3. J. P. Giddy and R. Safavi-Naini. Automated cryptanalysis of transposition ciphers. *The Computer Journal*, 37(5):429–436, 1994.
4. Robert A. J. Matthews. The use of genetic algorithms in cryptanalysis. *Cryptologia*, 17(2):187–201, April 1993.

Comparison of Genetic Algorithm and Particle Swarm Optimizer When Evolving a Recurrent Neural Network

Matthew Settles¹, Brandon Rodebaugh¹, and Terence Soule¹

Department of Computer Science, University of Idaho,
Moscow, Idaho U.S.A

Abstract. This paper compares the performance of GAs and PSOs in evolving weights of a recurrent neural network. The algorithms are tested on multiple network topologies. Both algorithms produce successful networks. The GA is more successful evolving larger networks and the PSO is more successful on smaller networks.¹

1 Background

In this paper we compare the performance of two population based algorithms, a genetic algorithm (GA) and particle swarm optimization (PSO), in training the weights of a strongly recurrent artificial neural network (RANN) for a number of different topologies. The goal is to develop a recurrent network that can reproduce the complex behaviors seen in biological neurons [1]. The combination of a strongly connected recurrent network and an output with a long period makes this a very difficult problem. Previous research in using evolutionary approaches to evolve RANNs have either evolved the topology and weights or used a hybrid algorithm that evolved the topology and used a local search or gradient descent search for the weights (see for example [2]).

2 Experiment and Results

Our goal is to evolve a network that produces a simple pulsed output when an activation ‘voltage’ is applied to the network’s input. The error is the sum of the absolute value of the difference between the desired output and the actual output at each time step plus a penalty (0.5) if the slope of the desired output differs in direction from the slope of the actual output.

The neural network is strongly connected with a single input node and a single output node. The nodes use a symmetric sigmoid activation function. The activation levels are calculated synchronically.

The GA uses a chromosomes consisting of real values. Each real value corresponds to the weight between one pair of nodes.

¹ This work supported by NSF EPSCoR EPS-0132626. The experiments were performed on a Beowulf cluster built with funds from NSF grant EPS-80935 and a generous hardware donation from Micron Technologies.

The GA is generational, 250 generations, 500 individuals per generation. The two best individuals are copied into the next generation (elitism). Tournament selection is used, with a tournament of size 3. The initial weights were randomly chosen in the range $(-1.0, 1.0)$. The mutation rate is $1/(LN)^2$. Mutation changes a weight by up to 25% of the weight's original value. Crossover is applied to two individuals at the same random (non-input) node. The crossover rate is 0.8.

The PSO uses position and velocity vectors which refer to the particles' position and velocity within the search space. They are real valued vectors, with one value for each network weight. The PSO is run for 250 generations on a population of 500 particles. The initial weights were randomly chosen in the range $(-1.0, 1.0)$. The position vector was allowed to explore values in the range of $(-2.0, 2.0)$. The inertial weight is reduced linearly from 0.9 to 0.4 each epoch [3].

Tables 1 and 2 show the number of successful trials out of the fifty. Successful trials evolve a network that produces periodic output with the desired frequency. Unsuccessful trials fail to produce periodic behavior.

Both the GA and PSO perform well for medium sized networks. The GAs optimal network size is around 3-4 layers with 5 nodes per layer. The PSOs optimal network is approximately 2x5. The GA is more successful with larger networks, whereas the PSO is more successful with smaller networks. A two-tailed z-test (α of 0.05) confirms that these differences are statistically significant.

Table 1. Number of successful trials (out of fifty) trained using GA.

Layers	1	2	3	4
1 Node/Layer	0	0	0	0
3 Nodes/Layer	0	17	44	49
5 Nodes/Layer	5	41	50	50
7 Nodes/Layer	22	48	46	41
9 Nodes/Layer	36	49	40	—

Table 2. Number of successful trials (out of fifty) trained using PSO.

Layers	1	2	3	4
1 Node/Layer	0	4	23	38
3 Nodes/Layer	17	43	49	47
5 Nodes/Layer	39	50	40	32
7 Nodes/Layer	46	46	36	19
9 Nodes/Layer	49	41	17	—

3 Conclusions and Future Work

In this paper we demonstrated that GA and PSO can be used to evolve the weights of strongly recurrent networks to produce long period, pulsed output signals from a constant valued input. Our results also show that both approaches are effective for a variety of different network topologies. Future work will include evolving a single network that can produce a variety of biologically relevant behaviors depending on the input signals.

References

1. Shepherd, G.M.: Neurobiology. Oxford University Press, New York, NY (1994)
2. Angeline, P.J., Saunders, G.M., Pollack, J.P.: An evolutionary algorithm that constructs recurrent neural networks. IEEE Transactions on Neural Networks 5 (1994) 54–65
3. Kennedy, J., Eberhart, R.: Swarm Intelligence. Morgan Kaufmann Publishers, Inc., San Francisco, CA (2001)

Adaptation and Ruggedness in an Evolvability Landscape

Terry Van Belle and David H. Ackley

Department of Computer Science
University of New Mexico
Albuquerque, New Mexico, USA
{vanbelle, ackley}@cs.unm.edu

Evolutionary processes depend on both *selection*—how fit any given individual may be, and on *evolvability*—how and how effectively new and fitter individuals are generated over time. While genetic algorithms typically represent the selection process explicitly by the fitness function and the information in the genomes, factors affecting evolvability are most often implicit in and distributed throughout the genetic algorithm itself, depending on the chosen genomic representation and genetic operators. In such cases, the genome itself has no direct control over evolvability except as determined by its fitness.

Researchers have explored mechanisms that allow the genome to affect not only fitness but also the distribution of offspring, thus opening up the potential of evolution to improve evolvability. In prior work [1] we demonstrated that effect with a simple model focusing on heritable evolvability in a changing environment.

In our current work [2], we introduce a simple evolvability model, similar in spirit to those of Evolution Strategies. In addition to genes that determine the fitness of the individual, in our model each individual contains a distinct set of ‘evolvability genes’ that determine the distribution of that individual’s potential offspring. We also present a simple dynamic environment that provides a canonical ‘evolvability opportunity’ by varying in a partially predictable manner.

That evolution might lead to improved evolvability is far from obvious, because selection operates only on an individual’s current fitness, but evolvability by definition only comes into play in subsequent generations. Two similarly-fit individuals will contribute about equally to the next generation, even if their evolvabilities vary drastically. Worse, if there is any fitness cost associated with evolvability, more evolvable individuals might get squeezed out before their advantages could pay off. The basic hope for increasing evolvability is circumstances where weak selective pressure allows diverse individuals to contribute offspring to the next generation, and then those individuals with better evolvability in the current generation will tend to produce offspring that will dominate in subsequent fitness competitions. In this way, evolvability advantages in the ancestors can lead to fitness advantages in the descendants, which then preserves the inherited evolvability mechanisms.

A common tool for imagining evolutionary processes is the fitness landscape, a function that maps the set of all genomes to a single-dimension real fitness value. Evolution is seen as the process of discovering peaks of higher fitness, while avoiding valleys of low fitness. If we can derive a scalar value that plau-

sibly captures the notion of evolvability, we can augment the fitness landscape conception with an analogous notion of an *evolvability landscape*. With our algorithm possessing variable and heritable evolvabilities, it is natural to wonder what the evolution of a population will look like on the evolvability landscape as well as the fitness landscape.

We adopt as an evolvability metric the *online fitness* of a population: The average fitness value of the best of population from the start of the run until a fixed number of generations have elapsed. The online fitness of a population with a fixed evolvability gives us the ‘height’ of the evolvability landscape at that point. In cases where evolvability is adaptive, we envision the population moving across the evolvability landscape as evolution proceeds, which in turn modifies the fitness landscape. Figures 1 and 2 show some of our results.

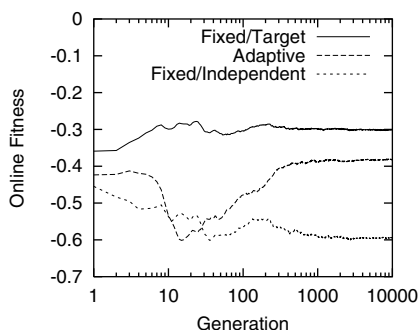


Fig. 1. *Fixed/Independent* is standard GA evolvability, in which all gene mutations are independent. *Fixed/Adaptive*, with an evolvable evolvability, does significantly better. *Fixed/Target* does best, but assumes advance knowledge of the environmental variation pattern.

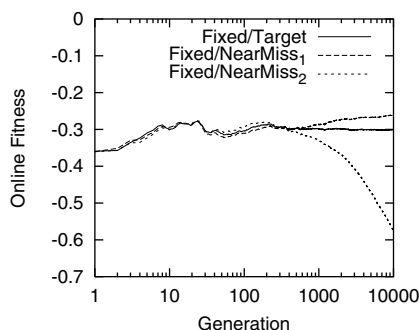


Fig. 2. Evidence of a ‘cliff’ in the evolvability landscape. Fixed evolvabilities that are close to optimal, but not exact, can produce extremely poor performance.

Acknowledgments. This research was supported in part by DARPA contract F30602-00-2-0584, and in part by NSF contract ANI 9986555.

References

- [1] Terry Van Belle and David H. Ackley. Code factoring and the evolution of evolvability. In *Proceedings of GECCO-2002*, New York City, July 2002. AAAI Press.
- [2] Terry Van Belle and David H. Ackley. Adaptation and ruggedness in an evolvability landscape. Technical Report TR-CS-2003-14, University of New Mexico, Department of Computer Science, 2003.
http://www.cs.unm.edu/colloq-bin/tech_reports.cgi?ID=TR-CS-2003-14.

Study Diploid System by a Hamiltonian Cycle Problem Algorithm

Dong Xianghui and Dai Ruwei

System Complexity Research Center
Institute of Automation, Chinese Academy of Science, Beijing 100080
xianghui.dong@mail.ia.ac.cn

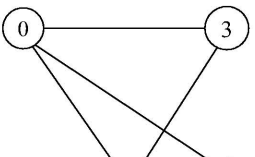
Abstract. Complex representation in Genetic Algorithms and pattern in real problems limits the effect of crossover to construct better pattern from sporadic building blocks. Instead of introducing more sophisticated operator, a diploid system was designed to divide the task into two steps: in meiosis phase, crossover was used to break two haploid of same individual into small units and remix them thoroughly. Then better phenotype was rebuilt from diploid of zygote in development phase. We introduced a new representation for Hamiltonian Cycle Problem and implemented an algorithm to test the system.

Our algorithm is different from conventional GA in several ways:

- The edges of potential solution are directly represented without coding.
- Crossover is only part of meiosis, working between diploid of same individual.
- Instead of mutation, the population size guarantees the diversity of genes.

Since Hamiltonian Cycle Problem is a NP-Complete problem, we can design a search algorithm for Non-deterministic Turing Machine.

Table 1. A graph with a Hamiltonian Cycle of (0, 3, 2, 1, 4, 5, 0), and two representation of Hamiltonian cycle

Edges In Order Of Connection			Edges In Order Of First Vertex	
Start	End		Start	End
0	3		0	3
3	2		1	4
2	1		2	1
1	4		3	2
4	5		4	5
5	0		5	0

To find the Hamiltonian Cycle, our Non-deterministic Turing Machine will:

- Check the first row. Choose a vertex from vertices connected to current first row vertex. These two vertices designate an edge.
- Process other rows in the same way. If there is a Hamiltonian Cycle and every choice is right, these n edges construct a valid cycle.
Therefore, we designed an evolutionary algorithm to simulate it approximately:
- Every individual represents a group of n edges got by a selecting procedure made by random or genetic operators.
- The fitness of an individual is the maximal length of contiguous path can extend from start in the edge group.

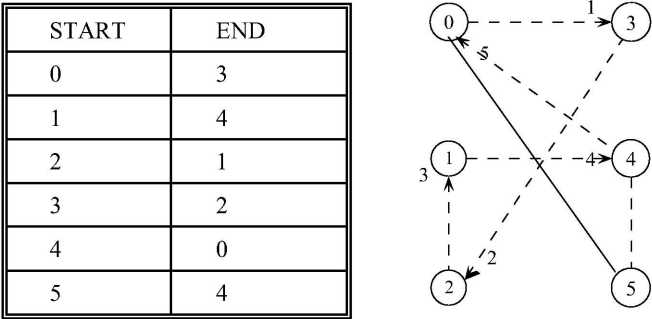


Fig. 1. Expression of genotype. Dashed edges are edges in genotype. Numbers in edges note the order of expression. The path terminated in 4-0 because of repetition

Since Hamiltonian Cycle Problem highly depend on the internal relation among vertices, it will be very hard for crossover to keep the validity of path and pattern formed at the same time. If edges are represented in path order, crossover may produce edge group with duplicate vertices; if edges are represented in the same fixed order, the low-order building blocks cannot be kept after crossover. Fortunately, the meiosis and diploid system in biology provide a solution for this problem. It can be divided into two steps:

1. Meiosis. Every chromosome got in gamete can come from either haploidy. Crossover and linkage occurred between corresponding chromosomes.
2. Diploid expression. No matter how thoroughly recombination had been conduct in meiosis, broken patterns can be recovered, and a better phenotype can be obtained with two options in every alleles. Our algorithm tests all the possible options in new searching branch and keeps the maximal contiguous path. The search space is not too much because many branches will be pruned for repeated vertex. Of course, we limited the size of searching branches pool.

It was proved that the algorithm usually solves graph with 16 vertices immediately. For larger scale (1000 ~ 5000) it had steady search capability only restrained by computing resource (mainly in space, not in time). Java codes and data are available from <http://ai.ia.ac.cn/english/people/draco/index.htm>.

Acknowledgments. The authors are very grateful to Prof. John Holland for invaluable encouragement and discussions.

A Possible Mechanism of Repressing Cheating Mutants in Myxobacteria

Ying Xiao and Winfried Just

Department of Mathematics, Ohio University, Athens, OH 45701, U.S.A.

Abstract. The formation of fruiting bodies by myxobacteria colonies involves altruistic suicide by many individual bacteria and is thus vulnerable to exploitation by cheating mutants. We report results of simulations that show how in a structured environment with patchy distribution of cheating mutants the wild type might persist.

This work was inspired by experiments on myxobacteria *Myxococcus xanthus* reported in [1]. Under adverse environmental conditions individuals in an *M. xanthus* colony aggregate densely and form a raised “fruiting body” that consists of a stalk and spores. During this process, many cells commit suicide in order to form the stalk. This “altruistic suicide” enables spore formation by other cells. When conditions become favorable again, the spores will be released and may start a new colony.

Velicer et al. studied in [1] some mutant strains that were deficient in their ability to form fruiting bodies and had lower motility but higher growth rates than wild-type bacteria. When mixed with wild-type bacteria, these mutant strains were significantly over-represented in the spores in comparison with their original frequency. Thus these mutants are cheaters in the sense that they reap the benefits of the collective action of the colony while paying a disproportionately low cost of altruistic suicide during fruiting body formation. The authors of [1] ask which mechanism insures that the wild-type behavior of altruistic suicide is evolutionarily stable against invasion by cheating mutants.

We conjecture that a clustered distribution of mutants at the time of sporulation events could be a sufficient mechanism for repressing those mutants. One possible source of such clustering could be lower motility of mutants.

A detailed description of the program written to test this conjecture, the source code, as well as all output files, can be found at the following URL: www.math.ohiou.edu/~just/Myxo/. The program simulates growth, development, and evolution of ten *M. xanthus* colonies over 500 seasons (sporulation events). Each season consists on average of 1,000 generations (cell divisions). Each colony is assumed to live on a square grid, and growth of the colony is modeled by expansion into neighboring grid cells. At any time during the simulation, each grid cell is characterized by the number of wild-type and mutant bacteria that it holds. At the end of each season, fruiting bodies are formed in regions where sufficiently many wild type bacteria are present. After each season, the program selects randomly ten fruiting bodies formed in this season and

seeds the new colonies with a mix of bacteria in the same proportions as the proportions found in the fruiting body that was chosen for reproduction.

The proportion of wild-type bacteria in excess of carrying capacity that move to neighboring grid cells in the expansion step was set to 0.024. We run ten simulations each for parameter settings where mutants in excess of carrying capacity move to neighboring grid cells at rates of 0.006, 0.008, 0.012, and 0.024 and grow 1%, 1.5%, or 2% faster than wild-type bacteria. In the following table, the column headers show the movement rates for the mutants, row headers show by how much mutants grow faster than wild type bacteria, and the numbers in the body of the table show how many of the simulations in each run of ten simulations reached the cutoff of 500 seasons without terminating due to lack of fruiting body formation.

Table 1. Number of simulations that run for 500 seasons

	0.006	0.008	0.012	0.024
1%	9	7	5	0
1.5%	6	5	2	0
2%	5	4	2	0

These results show that for many of our parameter settings, wild-type bacteria can successfully propagate in the presence of cheating mutants. Successful propagation of wild-type bacteria over many seasons is more likely the less the discrepancy in growth rates of mutants and wild type is, and the less mobile the mutants are. This can be considered as a proof of principle for our conjecture.

All our simulations in which mutants have the same motility as wild-type bacteria terminated prematurely due to lack of fruiting body formation. The authors of [2] report that motility of mutant strains that are deficient in their ability to form fruiting bodies can be (partially) restored in the laboratory. If such mutants do occur in nature, then our findings suggest that another defense mechanism is necessary for the wild-type bacteria to prevail against them.

References

1. Velicer, G. J., Kroos, L., Lenski, R. E.: Developmental cheating in the social bacterium *Myxococcus xanthus*. *Nature* **404** (2000) 598–601.
2. Velicer, G. J., Lenski, R. E., Kroos, L.: Rescue of Social Motility Lost during Evolution of *Myxococcus xanthus* in an Asocial Environment. *J. Bacteriol.* **184**(10) (2002) 2719–2727.

Tour Jeté, Pirouette: Dance Choreographing by Computers

Tina Yu¹ and Paul Johnson²

¹ ChevronTexaco Information Technology Company
6001 Bollinger Canyon Road
San Ramon, CA 94583
tiyu@chevrontexaco.com
<http://www.improvise.ws>

² Department of Political Science
University of Kansas
Lawrence, Kansas 66045
pauljohn@ukans.edu
<http://lark.cc.ku.edu/~pauljohn>

Abstract. This project is a “proof of concept” exercise intended to demonstrate the workability and usefulness of computer-generated choreography. We have developed a framework that represents dancers as individualized computer objects that can choose dance steps and move about on a rectangular dance floor. The effort begins with the creation of an agent-based model with the Swarm simulation toolkit. The individualistic behaviors of the computer agents can create a variety of dances, the movements and positions of which can be collected and animated with the Life Forms software. While there are certainly many additional elements of dance that could be integrated into this approach, the initial effort stands as evidence that interesting, useful insights into the development of dances can result from an integration of agent-based models and computerized animation of dances.

1 Introduction

Dance might be one of the most egoistic art forms ever created. This is partly due to the fact that human bodies are highly unique. Moreover, it is very difficult to record dance movements in precise details, no matter what method one uses. As a result, dances are frequently associated with the name of their choreographers, who not only create but also teach and deliver these art forms with ultimate authority. Such tight bonds between a dance and its creator gives the impression that dance is an art that can only be created by humans.

Indeed, creativity is one of the human traits that set us apart from other organisms. Random House Unabridged Dictionary defines creativity as “*the ability to transcend traditional ideas, rules, patterns, relationships or the like, and to create meaningful new ideas, forms, methods, interpretations, etc.,*” With the ability to create, humans

carry out the creation process in many different ways. One avenue is trial-and-error. It starts with an original idea and imagination. Through the process of repeated trying and learning from the failure, those that are unknown previously can be discovered and new things created.

Is creativity a quality that belongs to humans only? Do computers have the ability to create? We approach this question in two steps. First, can computers have original ideas and imagination? Second, can computers carry out the creation process?

Ideas and imagination seem to be something come and go on their own that no one has control over. Frequently, we heard artists discussing about where they *find* their ideas and what can *simulate* their imagination. What is computers' source of ideas and imagination? One answer is "randomness"; computers can be programmed to generate as many random numbers as needed. Such random numbers can be mapped into new possibilities of doing things, hence a source of ideas and imagination.

Creation process is very diverse in that different people have different approaches. For example, some dance choreographers like to work out the whole piece first and then teach them to their dancers. Others prefer working with their dancers to generate new ideas. Which style of creation process that computers can have? One answer is trial-and-error; computers can be programmed to repeat an operation as many times as needed. By applying such repetition to new/old ways of doing things, new possibilities can be discovered.

When equipped with a source of ideas and a process of creation, computers seem to become creative. This also suggests that computers might be able to create the art forms of dance. We are interested in computer-generated choreography and the possibility of incorporating that with human dancers to create a new kind of stage production. This paper describes the project and reports the progress we have made so far.

We started the project with a conversation with professional dancers and choreographers about their views of computer-generated choreography. Based on the discussion, we selected two computer tools (Swarm and Life Forms) for the project. We then implemented the "randomness" and "trial-and-error" abilities in the Swarm computer software to generate a sequence of dance steps. The music for this dance is then considered and selected. With a small degree of improvisation (according to the rhythm of the music), we put the dance sequences in animation. The initial results are then shown to a dance company's artistic director. The feedback is very encouraging, although the piece needs more work to be able to put into production. All of these lead us to conclude that computer-generated choreography can produce interesting movements that might lead to a new type of stage production.

The Swarm code: <http://lark.cc.ku.edu/~pauljohn/Swarm/MySwarmCode/Dancer>.
The Life Forms dance animation: <http://www.improvise.ws/Dance.mov.zip>.

Multiobjective Optimization Using Ideas from the Clonal Selection Principle

Nareli Cruz Cortés and Carlos A. Coello Coello

CINVESTAV-IPN
Evolutionary Computation Group
Depto. de Ingeniería Eléctrica
Sección de Computación
Av. Instituto Politécnico Nacional No. 2508
Col. San Pedro Zacatenco
México, D. F. 07300, MEXICO
nareli@computacion.cs.cinvestav.mx,
ccoello@cs.cinvestav.mx

Abstract. In this paper, we propose a new multiobjective optimization approach based on the clonal selection principle. Our approach is compared with respect to other evolutionary multiobjective optimization techniques that are representative of the state-of-the-art in the area. In our study, several test functions and metrics commonly adopted in evolutionary multiobjective optimization are used. Our results indicate that the use of an artificial immune system for multiobjective optimization is a viable alternative.

1 Introduction

Most optimization problems naturally have several objectives to be achieved (normally conflicting with each other), but in order to simplify their solution, they are treated as if they had only one (the remaining objectives are normally handled as constraints). These problems with several objectives, are called “multiobjective” or “vector” optimization problems, and were originally studied in the context of economics. However, scientists and engineers soon realized that such problems naturally arise in all areas of knowledge. Over the years, the work of a considerable number of operational researchers has produced a wide variety of techniques to deal with multiobjective optimization problems [13]. However, it was until relatively recently that researchers realized of the potential of evolutionary algorithms (EAs) and other population-based heuristics in this area [7].

The main motivation for using EAs (or any other population-based heuristics) in solving multiobjective optimization problems is because EAs deal simultaneously with a set of possible solutions (the so-called population) which allows us to find several members of the Pareto optimal set in a single run of the algorithm, instead of having to perform a series of separate runs as in the case of the traditional mathematical programming techniques [13]. Additionally, EAs are less susceptible to the shape or continuity of the Pareto front (e.g., they can easily deal with discontinuous and concave Pareto fronts), whereas these two issues are a real concern for mathematical programming techniques [7,3].

Despite the considerable amount of research on evolutionary multiobjective optimization in the last few years, there have been very few attempts to extend certain population-based heuristics (e.g., cultural algorithms and particle swarm optimization) [3]. Particularly, the efforts to extend an artificial immune system to deal with multiobjective optimization problems have been practically inexistent until very recently. In this paper, we precisely provide one of the first proposals to extend an artificial immune system to solve multiobjective optimization problems (either with or without constraints). Our proposal is based on the clonal selection principle and is validated using several test functions and metrics, following the standard methodology adopted in this area [3].

2 The Immune System

One of the main goals of the immune system is to protect the human body from the attack of foreign (harmful) organisms. The immune system is capable of distinguishing between the normal components of our organism and the foreign material that can cause us harm (e.g., bacteria). Those molecules that can be recognized by the immune system are called *antigens* that elicit an adaptive immune response.

The molecules called *antibodies* play the main role on the immune system response. The immune response is specific to a certain foreign organism (antigen). When an antigen is detected, those antibodies that best recognize an antigen will proliferate by cloning. This process is called *clonal selection principle* [5].

The new cloned cells undergo high rate somatic mutations or *hypermutation*. The main roles of that mutation process are twofold: to allow the creation of new molecular patterns for antibodies, and to maintain diversity. These mutations experienced by the clones are proportional to their affinity to the antigen. The highest affinity antibodies experiment the lowest mutation rates, whereas the lowest affinity antibodies have high mutation rates. After this mutation process ends, some clones could be dangerous for the body and should therefore be eliminated.

After these cloning and hypermutation processes finish, the immune system has improved the antibodies' affinity, which results on the antigen neutralization and elimination. At this point, the immune system must return to its normal condition, eliminating the excedent cells. However, some cells remain circulating throughout the body as memory cells. When the immune system is later attacked by the same type of antigen (or a similar one), these memory cells are activated, presenting a better and more efficient response. This second encounter with the same antigen is called *secondary response*. The algorithm proposed in this paper is based on the clonal selection principle previously described.

3 Previous Work

The first direct use of the immune system to solve multiobjective optimization problems reported in the literature is the work of Yoo and Hajela [20]. This approach uses a linear aggregating function to combine objective function and constraint information into a scalar value that is used as the fitness function of a genetic algorithm. The use of different weights allows the authors to converge to a certain (pre-specified) number of

points of the Pareto front, since they make no attempt to use any specific technique to preserve diversity. Besides the limited spread of nondominated solutions produced by the approach, it is well-known that linear aggregating functions have severe limitations for solving multiobjective problems (the main one is that they cannot generate concave portions of the Pareto front [4]). The approach of Yoo & Hajela is not compared to any other technique.

de Castro and Von Zuben [6] proposed an approach, called CLONALG, which is based on the clonal selection principle and is used to solve pattern recognition and multimodal optimization problems. This approach can be considered as the first attempt to solve multimodal optimization problems which are closely related to multiobjective optimization problems (although in multimodal optimization, the main emphasis is to preserve diversity rather than generating nondominated solutions as in multiobjective optimization).

Anchor et al. [1] adopted both lexicographic ordering and Pareto-based selection in an evolutionary programming algorithm used to detect attacks with an artificial immune system for virus and computer intrusion detection. In this case, however, the paper is more focused on the application rather than on the approach and no proper validation of the proposed algorithms is provided.

The current paper is an extension of the work published in [2]. Note however, that our current proposal has several important differences with respect to the previous one. In our previous work, we attempted to follow the clonal selection principle very closely, but our results could not be improved beyond a certain point. Thus, we decided to sacrifice some of the biological metaphor in exchange for a better performance of our algorithm. The result of these changes is the proposal presented in this paper.

4 The Proposed Approach

Our algorithm is the following:

1. The initial population is created by dividing decision variable space into a certain number of segments with respect to the desired population size. Thus, we generate an initial population with a uniform distribution of solutions such that every segment in which the decision variable space is divided has solutions. This is done to improve the search capabilities of our algorithm instead of just relying on the use of a mutation operator. Note however, that the solutions generated for the initial population are still random.
2. Initialize the secondary memory so that it is empty.
3. Determine for each individual in the population, if it is (Pareto) dominated or not. For constrained problems, determine if an individual is feasible or not.
4. Determine which are the “best antibodies”, since we will clone them adopting the following criterion:

- If the problem is unconstrained, then all the nondominated individuals are cloned.
 - If the problem is constrained, then we have two further cases: a) there are feasible individuals in the population, and b) there are no feasible individuals in the population. For case b), all the nondominated individuals are cloned. For case a), only the nondominated individuals that are feasible are cloned (nondominance is measured only with respect to other feasible individuals in this case).
5. Copy all the best antibodies (obtained from the previous step) into the secondary memory.
 6. We determine for each of the “best” antibodies the number of clones that we want to create. We wish to create the same number of clones of each antibody, and we also that the total number of clones created amounts the 60% of the total population size used. However, if the secondary memory is full, then we modify this quantity doing the following:
 - If the individual to be inserted into the secondary memory is not allowed access either because it was repeated or because it belongs to the most crowded region of objective function space, then the number of clones created is zero.
 - When we have an individual that belongs to a cell whose number of solutions contained is below average (with respect to all the occupied cells in the secondary memory), then the number of clones to be generated is duplicated.
 - When we have an individual that belongs to a cell whose number of solutions contained is above average (with respect to all the occupied cells in the adaptive grid), then the number of clones to be generated is reduced by half.
 7. We perform the cloning of the best antibodies based on the information from the previous step. Note that the population size grows after the cloning process takes place. Then, we eliminate the extra individuals giving preference (for survival) to the new clones generated.
 8. A mutation operator is applied to the clones in such a way that the number of mutated genes in each chromosomic string is equal to the number of decision variables of the problem. This is done to make sure that at least one mutation occurs per string, since otherwise we would have duplicates (the original and the cloned string would be exactly the same).
 9. We apply a non-uniform mutation operator to the “worst” antibodies (i.e., those not selected as “best antibodies” in step 4). The initial mutation rate adopted is high and it is decreased linearly over time (from 0.9 to 0.3).
 10. If the secondary memory is full, we apply crossover to a fraction of its contents (we proposed 60%). The new individuals generated that are nondominated with respect to the secondary memory will then be added to it.

11. After that cloning process ends, the population size is increased. Later on, it is necessary to reset the population size to its original value. At this point, we eliminate the excedent individuals, allowing the survival of the nondominated solutions.
12. We repeat this process from step 3 during a certain (predetermined) number of times.

Note that in the previous algorithm there is no distinction between antigen and antibody. In contrast, in this case all the individuals are considered as antibodies, and we only distinguish between “better” antibodies and “not so good” antibodies.

The reason for using an initial population with a uniform distribution of solutions over the allowable range of the decision variables is to sample the search space uniformly. This helps the mutation operator to explore the search space more efficiently.

We apply crossover to the individuals in the secondary memory once this is full so that we can reach intermediate points between them. Such information is used to improve the performance of our algorithm.

Note that despite the similarities of our approach with CLONALG, there are important differences such as the selection strategy, the mutation rate and the number of clones created by each approach. Also, note that our approach incorporates some operators taken from evolutionary algorithms (e.g., the crossover operator applied to the elements of the secondary memory (step 10 from our algorithm)). Despite that fact, the cloning process (which involves the use of a variable-size population) of our algorithm differs from the standard definition of an evolutionary algorithm.

4.1 Secondary Memory

We use a secondary or external memory as an elitist mechanism in order to maintain the best solutions found along the process. The individuals stored in this memory are all nondominated not only with respect to each other but also with respect to all of the previous individuals who attempted to enter the external memory. Therefore, the external memory stores our approximation to the true Pareto front of the problem.

In order to enforce a uniform distribution of nondominated solutions that cover the entire Pareto front of a problem, we use the adaptive grid proposed by Knowles and Corne [11] (see Figure 1).

Ideally, the size of the external memory should be infinite. However, since this is not possible in practice, we must set a limit to the number of nondominated solutions that we want to store in this secondary memory. By enforcing this limit, our external memory will get full at some point even if there are more nondominated individuals wishing to enter. When this happens, we use an additional criterion to allow a nondominated individual to enter the external memory: region density (i.e., individuals belonging to less densely populated regions are given preference).

The algorithm for the implementation of the adaptive grid is the following:

1. Divide objective function space according to the number of subdivisions set by the user.
2. For each individual in the external memory, determine the cell to which it belongs.
3. If the external memory is full, then determine which is the most crowded cell.

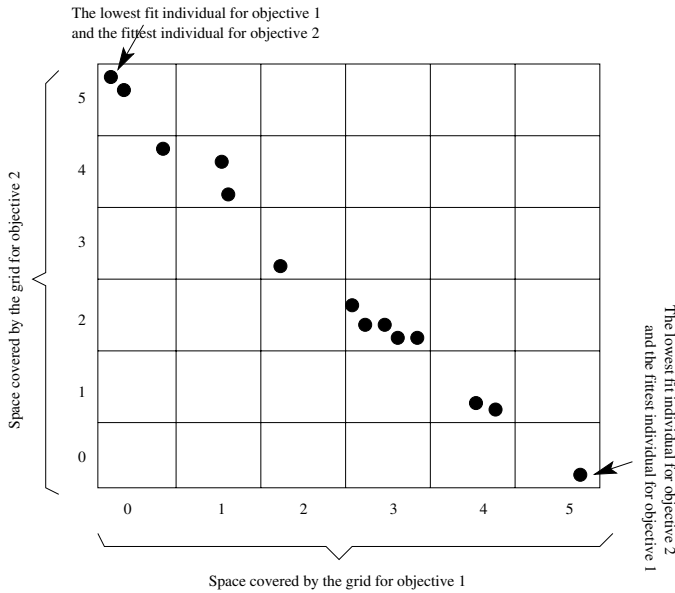


Fig. 1. An adaptive grid to handle the secondary memory

- To determine if a certain antibody is allowed to enter the external memory, do the following:
 - If it belongs to the most crowded cell, then it is not allowed to enter.
 - Otherwise, the individual is allowed to enter. For that sake, we eliminate a (randomly chosen) individual that belongs to the most crowded cell in order to have an available slot for the antibody.

5 Experiments

In order to validate our approach, we used several test functions reported in the standard evolutionary multiobjective optimization literature [18,3]. In each case, we generated the true Pareto front of the problem (i.e., the solution that we wished to achieve) by enumeration using parallel processing techniques. Then, we plotted the Pareto front generated by our algorithm, which we call the multiobjective immune system algorithm (MISA).

The results indicated below were found using the following parameters for MISA: Population size = 100, number of grid subdivisions = 25, size of the external memory = 100 (this is a value normally adopted by researchers in the specialized literature [3]). The number of iterations to be performed by the algorithm is determined by the number of fitness function evaluations required. The previous parameters produce a total of 12,000 fitness function evaluations.

MISA was compared against the NSGA-II [9] and against PAES [11]. These two algorithms were chosen because they are representative of the state-of-the-art in evolutionary multiobjective optimization and their codes are in the public domain.

The Nondominated Sorting Genetic Algorithm II (NSGA-II) [8,9] is based on the use of several layers to classify the individuals of the population, and uses elitism and a crowded comparison operator that keeps diversity without specifying any additional parameters. The NSGA-II is a revised (and more efficient) version of the NSGA [16].

The Pareto Archived Evolution Strategy (PAES) [11] consists of a (1+1) evolution strategy (i.e., a single parent that generates a single offspring) in combination with a historical archive that records some of the nondominated solutions previously found. This archive is used as a reference set against which each mutated individual is being compared.

All the approaches performed the same number of fitness function evaluations as MISA and they all adopted the same size for their external memories.

In the following examples, the NSGA-II was run using a population size of 100, a crossover rate of 0.75, tournament selection, and a mutation rate of $1/\text{vars}$, where vars = number of decision variables of the problem.

PAES was run using a mutation rate of $1/L$, where L refers to the length of the chromosomal string that encodes the decision variables. Besides the graphical comparisons performed, the three following metrics were adopted to allow a quantitative comparison of results:

- **Error Ratio (ER):** This metric was proposed by Van Veldhuizen [17] to indicate the percentage of solutions (from the nondominated vectors found so far) that are not members of the true Pareto optimal set:

$$ER = \frac{\sum_{i=1}^n e_i}{n}, \quad (1)$$

where n is the number of vectors in the current set of nondominated vectors available; $e_i = 0$ if vector i is a member of the Pareto optimal set, and $e_i = 1$ otherwise. It should then be clear that $ER = 0$ indicates an ideal behavior, since it would mean that all the vectors generated by our algorithm belong to the Pareto optimal set of the problem.

- **Spacing (S):** This metric was proposed by Schott [15] as a way of measuring the range (distance) variance of neighboring vectors in the Pareto front known. This metric is defined as:

$$S \triangleq \sqrt{\frac{1}{n-1} \sum_{i=1}^n (\bar{d} - d_i)^2}, \quad (2)$$

where $d_i = \min_j (|f_1^i(\mathbf{x}) - f_1^j(\mathbf{x})| + |f_2^i(\mathbf{x}) - f_2^j(\mathbf{x})|)$, $i, j = 1, \dots, n$, $\bar{d}$ is the mean of all d_i , and n is the number of vectors in the Pareto front found by the algorithm being evaluated. A value of zero for this metric indicates all the nondominated solutions found are equidistantly spaced.

- **Generational Distance (GD):** The concept of generational distance was introduced by Van Veldhuizen & Lamont [19] as a way of estimating how far are the elements in the Pareto front produced by our algorithm from those in the true Pareto front of the problem. This metric is defined as:

$$GD = \frac{\sqrt{\sum_{i=1}^n d_i^2}}{n} \quad (3)$$

where n is the number of nondominated vectors found by the algorithm being analyzed and d_i is the Euclidean distance (measured in objective space) between each of these and the nearest member of the true Pareto front. It should be clear that a value of $GD = 0$ indicates that all the elements generated are in the true Pareto front of the problem. Therefore, any other value will indicate how “far” we are from the global Pareto front of our problem.

In all the following examples, we performed 20 runs of each algorithm. The graphs shown in each case were generated using the average performance of each algorithm with respect to generational distance.

Example 1

Our first example is a two-objective optimization problem proposed by Schaffer [14]:

$$\text{Minimize } f_1(x) = \begin{cases} -x & \text{if } x \leq 1 \\ -2 + x & \text{if } 1 < x \leq 3 \\ 4 - x & \text{if } 3 < x \leq 4 \\ -4 + x & \text{if } x > 4 \end{cases} \quad (4)$$

$$\text{Minimize } f_2(x) = (x - 5)^2 \quad (5)$$

and $-5 \leq x \leq 10$.

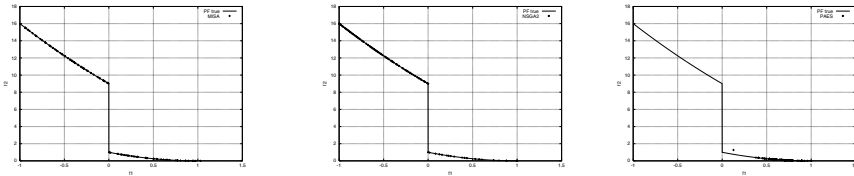


Fig. 2. Pareto front obtained by MISA (left), the NSGA-II (middle) and PAES (right) in the first example. The true Pareto front of the problem is shown as a continuous line (note that the vertical segment is NOT part of the Pareto front and is shown only to facilitate drawing the front).

The comparison of results between the true Pareto front of this example and the Pareto front produced by MISA, the NSGA-II, and PAES are shown in Figure 2. The values of the three metrics for each algorithm are presented in Tables 1 and 2.

Table 1. Spacing and Generational Distance for the first example.

	Spacing			GD		
	MISA	NSGA-II	PAES	MISA	NSGA-II	PAES
Average	0.236345	0.145288	0.268493	0.000375	0.000288	0.002377
Best	0.215840	0.039400	0.074966	0.000199	0.000246	0.000051
Worst	0.256473	0.216794	1.592858	0.001705	0.000344	0.034941
Std. Dev.	0.013523	0.079389	0.336705	0.000387	0.000022	0.007781
Median	0.093127	0.207535	0.137584	0.000387	0.000285	0.000239

In this case, MISA had the best average value with respect to generational distance. The NSGA-II had both the best average spacing and the best average error ratio. Graphically, we can see that PAES was unable to find most of the true Pareto front of the problem. MISA and the NSGA-II were able to produce most of the true Pareto front and their overall performance seems quite similar from the graphical results with a slight advantage for MISA with respect to closeness to the true Pareto front and a slight advantage for the NSGA-II with respect to uniform distribution of solutions.

Table 2. Error ratio for the first example.

	MISA	NSGA-II	PAES
Average	0.410094	0.210891	0.659406
Best	0.366337	0.178218	0.227723
Worst	0.445545	0.237624	1.000000
Std. Dev.	0.025403	0.018481	0.273242
Median	0.410892	0.207921	0.663366

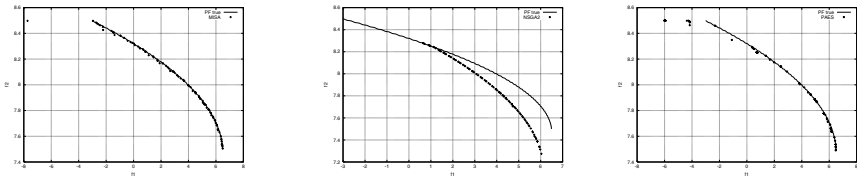


Fig. 3. Pareto front obtained by MISA (left), the NSGA-II (middle) and PAES (right) in the second example. The true Pareto front of the problem is shown as a continuous line.

Example 2

The second example was proposed by Kita [10]: Maximize $F = (f_1(x, y), f_2(x, y))$ where: $f_1(x, y) = -x^2 + y$, $f_2(x, y) = \frac{1}{2}x + y + 1$, $x, y \geq 0$, $0 \geq \frac{1}{6}x + y - \frac{13}{2}$,

$0 \geq \frac{1}{2}x + y - \frac{15}{2}$, $0 \geq 5x + y - 30$. The comparison of results between the true Pareto front of this example and the Pareto front produced by MISA, the NSGA-II and PAES are shown in Figure 3. The values of the three metrics for each algorithm are presented in Tables 3, and 4.

Table 3. Spacing and Generational Distance for the second example.

	Spacing			GD		
	MISA	NSGA-II	PAES	MISA	NSGA-II	PAES
Average	0.905722	0.815194	0.135875	0.036707	0.049669	0.095323
Best	0.783875	0.729958	0.048809	0.002740	0.004344	0.002148
Worst	1.670836	1.123444	0.222275	0.160347	0.523622	0.224462
Std. Dev.	0.237979	0.077707	0.042790	0.043617	0.123888	0.104706
Median	0.826587	0.173106	0.792552	0.019976	0.066585	0.018640

In this case, MISA had again the best average value for the generational distance. The NSGA-II had the best average error ratio and PAES had the best average spacing value. Note however from the graphical results that the NSGA-II missed most of the true Pareto front of the problem. PAES also missed some portions of the true Pareto front of the problem. Graphically, we can see that MISA found most of the true Pareto front and therefore, we argue that it had the best overall performance in this test function.

Table 4. Error ratio for the second example.

	MISA	NSGA-II	PAES
Average	0.007431	0.002703	0.005941
Best	0.000000	0.000000	0.000000
Worst	0.010000	0.009009	0.009901
Std. Dev.	0.004402	0.004236	0.004976
Median	0.009901	0.0000	0.009901

Example 3

Our third example is a two-objective optimization problem defined by Kursawe [12]:

$$\text{Minimize } f_1(\mathbf{x}) = \sum_{i=1}^{n-1} \left(-10 \exp \left(-0.2 \sqrt{x_i^2 + x_{i+1}^2} \right) \right) \quad (6)$$

$$\text{Minimize } f_2(\mathbf{x}) = \sum_{i=1}^n (|x_i|^{0.8} + 5 \sin(x_i)^3) \quad (7)$$

where: $-5 \leq x_1, x_2, x_3 \leq 5$

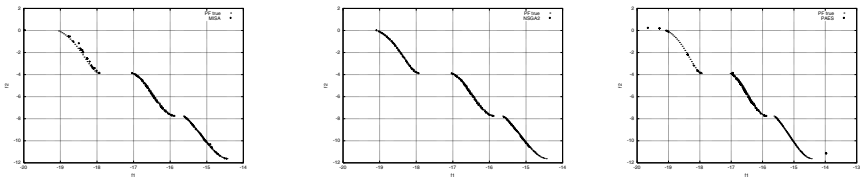


Fig. 4. Pareto front obtained by MISA (left), and the NSGA-II (middle) and PAES (right) in the third example. The true Pareto front of the problem is shown as a continuous line.

The comparison of results between the true Pareto front of this example and the Pareto front produced by MISA, the NSGA-II and PAES are shown in Figure 4. The values of the three metrics for each algorithm are presented in Tables 5 and 6.

Table 5. Spacing and Generational Distance for the third example.

	Spacing			GD		
	MISA	NSGA-II	PAES	MISA	NSGA-II	PAES
Average	3.188819	2.889901	3.019393	0.004152	0.004164	0.009341
Best	3.177936	2.705087	2.728101	0.003324	0.003069	0.002019
Worst	3.203547	3.094213	3.200678	0.005282	0.007598	0.056152
Std. Dev.	0.007210	0.123198	0.133220	0.000525	0.001178	0.013893
Median	3.186680	2.842901	3.029246	0.004205	0.003709	0.004468

For this test function, MISA had again the best average generational distance (this value was, however, only marginally better than the average value of the NSGA-II). The NSGA-II had the best average spacing value and the best average error ratio. However, by looking at the graphical results, it is clear that the NSGA-II missed the last (right lowerhand) portion of the true Pareto front, although it got a nice distribution of solutions along the rest of the front. PAES missed almost entirely two of the three parts that make the true Pareto front of this problem. Therefore, we argue in this case that MISA was practically in a tie with the NSGA-II in terms of best overall performance, since MISA covered the entire Pareto front, but the NSGA-II had a more uniform distribution of solutions.

Based on the limited set of experiments performed, we can see that MISA provides competitive results with respect to the two other algorithms against which it was compared. Although it did not always ranked first when using the three metrics adopted, in all cases it produced reasonably good approximations of the true Pareto front of each problem under study (several other test functions were adopted but not included due to space limitations), particularly with respect to the generational distance metric. Nevertheless, a more detailed statistical analysis is required to be able to derive more general conclusions.

Table 6. Error ratio for example 3

	MISA	NSGA-II	PAES
Average	0.517584	0.262872	0.372277
Best	0.386139	0.178218	0.069307
Worst	0.643564	0.396040	0.881188
Std. Dev.	0.066756	0.056875	0.211876
Median	0.504951	0.252476	0.336634

6 Conclusions and Future Work

We have introduced a new multiobjective optimization approach based on the clonal selection principle. The approach was found to be competitive with respect to other algorithms representative of the state-of-the-art in the area. Our main conclusion is that the sort of artificial immune system proposed in this paper is a viable alternative to solve multiobjective optimization problems in a relatively simple way. We also believe that, given the features of artificial immune systems, an extension of this paradigm for multiobjective optimization (such as the one proposed here) may be particularly useful to deal with dynamic functions and that is precisely part of our future research. Also, it is desirable to refine the mechanism to maintain diversity that our approach currently has, since that is its main current weakness.

Acknowledgements. We thank the comments of the anonymous reviewers that greatly helped us to improve the contents of this paper. The first author acknowledges support from CONACyT through a scholarship to pursue graduate studies at the Computer Science Section of the Electrical Engineering Department at CINVESTAV-IPN. The second author gratefully acknowledges support from CONACyT through project 34201-A.

References

1. Kevin P. Anchor, Jesse B. Zydallis, Gregg H. Gunsch, and Gary B. Lamont. Extending the Computer Defense Immune System: Network Intrusion Detection with a Multiobjective Evolutionary Programming Approach. In Jonathan Timmis and Peter J. Bentley, editors, *First International Conference on Artificial Immune Systems (ICARIS'2002)*, pages 12–21. University of Kent at Canterbury, UK, September 2002. ISBN 1-902671-32-5.
2. Carlos A. Coello Coello and Nareli Cruz Cortés. An Approach to Solve Multiobjective Optimization Problems Based on an Artificial Immune System. In Jonathan Timmis and Peter J. Bentley, editors, *First International Conference on Artificial Immune Systems (ICARIS'2002)*, pages 212–221. University of Kent at Canterbury, UK, September 2002. ISBN 1-902671-32-5.
3. Carlos A. Coello Coello, David A. Van Veldhuizen, and Gary B. Lamont. *Evolutionary Algorithms for Solving Multi-Objective Problems*. Kluwer Academic Publishers, New York, May 2002. ISBN 0-3064-6762-3.

4. Indraneel Das and John Dennis. A Closer Look at Drawbacks of Minimizing Weighted Sums of Objectives for Pareto Set Generation in Multicriteria Optimization Problems. *Structural Optimization*, 14(1):63–69, 1997.
5. Leandro N. de Castro and Jonathan Timmis. *Artificial Immune Systems: A New Computational Intelligence Approach*. Springer, London, 2002.
6. Leandro Nunes de Castro and F. J. Von Zuben. Learning and Optimization Using the Clonal Selection Principle. *IEEE Transactions on Evolutionary Computation*, 6(3):239–251, 2002.
7. Kalyanmoy Deb. *Multi-Objective Optimization using Evolutionary Algorithms*. John Wiley & Sons, Chichester, UK, 2001. ISBN 0-471-87339-X.
8. Kalyanmoy Deb, Samir Agrawal, Amrit Pratab, and T. Meyarivan. A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II. In Marc Schoenauer, Kalyanmoy Deb, Günter Rudolph, Xin Yao, Evelynne Lutton, Juan Julian Merelo, and Hans-Paul Schwefel, editors, *Proceedings of the Parallel Problem Solving from Nature VI Conference*, pages 849–858, Paris, France, 2000. Springer. Lecture Notes in Computer Science No. 1917.
9. Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and T. Meyarivan. A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, April 2002.
10. Hajime Kita, Yasuyuki Yabumoto, Naoki Mori, and Yoshikazu Nishikawa. Multi-Objective Optimization by Means of the Thermodynamical Genetic Algorithm. In Hans-Michael Voigt, Werner Ebeling, Ingo Rechenberg, and Hans-Paul Schwefel, editors, *Parallel Problem Solving from Nature—PPSN IV*, Lecture Notes in Computer Science, pages 504–512, Berlin, Germany, September 1996. Springer-Verlag.
11. Joshua D. Knowles and David W. Corne. Approximating the Nondominated Front Using the Pareto Archived Evolution Strategy. *Evolutionary Computation*, 8(2):149–172, 2000.
12. Frank Kursawe. A Variant of Evolution Strategies for Vector Optimization. In H. P. Schwefel and R. Männer, editors, *Parallel Problem Solving from Nature. 1st Workshop, PPSN I*, volume 496 of *Lecture Notes in Computer Science*, pages 193–197, Berlin, Germany, oct 1991. Springer-Verlag.
13. Kaisa M. Miettinen. *Nonlinear Multiobjective Optimization*. Kluwer Academic Publishers, Boston, Massachusetts, 1998.
14. J. David Schaffer. *Multiple Objective Optimization with Vector Evaluated Genetic Algorithms*. PhD thesis, Vanderbilt University, 1984.
15. Jason R. Schott. Fault Tolerant Design Using Single and Multicriteria Genetic Algorithm Optimization. Master's thesis, Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, Cambridge, Massachusetts, May 1995.
16. N. Srinivas and Kalyanmoy Deb. Multiobjective Optimization Using Nondominated Sorting in Genetic Algorithms. *Evolutionary Computation*, 2(3):221–248, Fall 1994.
17. David A. Van Veldhuizen. *Multiobjective Evolutionary Algorithms: Classifications, Analyses, and New Innovations*. PhD thesis, Department of Electrical and Computer Engineering. Graduate School of Engineering. Air Force Institute of Technology, Wright-Patterson AFB, Ohio, May 1999.
18. David A. Van Veldhuizen and Gary B. Lamont. MOEA Test Suite Generation, Design & Use. In Annie S. Wu, editor, *Proceedings of the 1999 Genetic and Evolutionary Computation Conference. Workshop Program*, pages 113–114, Orlando, Florida, July 1999.
19. David A. Van Veldhuizen and Gary B. Lamont. On Measuring Multiobjective Evolutionary Algorithm Performance. In *2000 Congress on Evolutionary Computation*, volume 1, pages 204–211, Piscataway, New Jersey, July 2000. IEEE Service Center.
20. J. Yoo and P. Hajela. Immune network simulations in multicriterion design. *Structural Optimization*, 18:85–94, 1999.

A Hybrid Immune Algorithm with Information Gain for the Graph Coloring Problem

Vincenzo Cutello, Giuseppe Nicosia, and Mario Pavone

University of Catania,
Department of Mathematics and Computer Science
V.le A. Doria 6, 95125 Catania, Italy
{cutello,nicosia,mpavone}@dmi.unict.it

Abstract. We present a new Immune Algorithm that incorporates a simple local search procedure to improve the overall performances to tackle the graph coloring problem instances. We characterize the algorithm and set its parameters in terms of Information Gain. Experiments will show that the IA we propose is very competitive with the best evolutionary algorithms.

Keywords: Immune Algorithm, Information Gain, Graph coloring problem, Combinatorial optimization.

1 Introduction

In the last five years we have witnessed an increasing number of algorithms, models and results in the field of Artificial Immune Systems [1,2]. Natural Immune System provide an excellent example of bottom up intelligent strategy, in which adaptation operates at the local level of cells and molecules, and useful behavior emerges at the global level, the immune humoral response. From an information processing point of view [3] the Immune System (IS) can be seen as a problem learning and solving system. The *antigen* (Ag) is the problem to solve, the *antibody* (Ab) is the generated solution. At the beginning of the *primary response* the antigen-problem is recognized by poor candidate solution. At the end of the primary response the antigen-problem is defeated-solved by good candidate solutions. Consequently the primary response corresponds to a training phase while the *secondary response* is the testing phase where we will try to solve problems similar to the original presented in the primary response [4].

Recent studies show that when one faces the Graph Coloring Problem (GCP) with evolutionary algorithms (EAs), the best results are often obtained by hybrid EAs with local search and specialized crossover [5]. In particular, the random crossover operator used in a standard genetic algorithm performs poorly for combinatorial optimization problem and, in general, the crossover operator must be designed carefully to identify important properties, building blocks, which must be transmitted from parents population to offspring population. Hence the design of a *good* crossover operator is crucial for the overall performance of the

EAs. The drawback is that it might happen to recombine good individuals from different regions of the search space, having *different symmetries*, producing poor offspring [6]. For this reason, we use an Immunological Algorithm (IA) to tackle the GCP. IAs do not have a crossover operator, and the crucial task of designing an appropriate crossover operator is avoided at once. The IA we will propose makes use of a particular mutation operator and a local search strategy without having to incorporate specific domain knowledge.

For sake of clarity, we recall some basic definitions. Given an undirected graph $G = (V, E)$ with vertex set V , edge set E and a positive integer $K \leq |V|$, the Graph Coloring Problem asks whether G is K -colorable, i.e. whether there exists a function $f : V \rightarrow \{1, 2, \dots, K\}$ such that $f(u) \neq f(v)$ whenever $\{u, v\} \in E$. The GCP is a well-known NP-complete problem [7]. Exact solutions can be found for simple or medium instances [8,9]. Coloring problems are very closely related with *cliques* [10] (complete subgraphs). The size of the maximum clique is a lower bound on the minimum number of colors needed to color a graph, $\chi(G)$. Thus, if $\omega(G)$ is the size of the maximum clique: $\chi(G) \geq \omega(G)$.

2 Immune Algorithms

We work with a simplified model of the natural immune system. We will see that the IA presented in this work is very similar to De Castro, Von Zuben's algorithm, CLONALG [11,12] and to Nicosia *et al.* immune algorithm [4,13]. We consider only two entities: Ag and B cells. Ag is the problem and the B cell receptor is the candidate solution. Formally, Ag is a set of variables that models the problem; and, B cells are defined as strings of integers of finite length $\ell = |V|$. The input is the antigen-problem, the output is basically the candidate solutions-B cells that solve-recognize the Ag.

By $P^{(t)}$ we will denote a population of d individuals of length ℓ , which represent a subset of the space of feasible solutions of length ℓ , S^ℓ , obtained at time t . The initial population of B cells, i.e. the initial set $P^{(0)}$, is created randomly. After initialization, there are three different phases.

In the *Interaction phase* the population $P^{(t)}$ is evaluated. $f(\mathbf{x}) = m$ is the fitness function value of B cell receptor $\mathbf{x}$. Hence for the GCP, the fitness function $f(\mathbf{x}) = m$ indicates that there exists a m -coloring for G , that is, a partition of vertices $V = S_1 \cup S_2 \cup \dots \cup S_m$ such that each $S_i \subseteq V$ is a subset of vertices which are pairwise not adjacent (i.e. each S_i is an *independent set*).

The *Cloning expansion phase* is composed of two steps: *cloning* and *hypermutation*. The cloning expansion events are modeled by *cloning potential* V and *mutation number* M , which depend upon f . If we exclude all the adaptive mechanisms [14] in EA's (e.g., adaptive mutation and adaptive crossover rates which are related to the fitness function values), the immune operators, contrary to standard evolutionary operators, depend upon the fitness function values [15]. Cloning potential is a truncated exponential: $V(f(\mathbf{x})) = e^{-k(\ell - f(\mathbf{x}))}$, where the parameter k determines the sharpness of the potential. The cloning operator generates the population P^{clo} . The mutation number is a simple straight line:

$M(f(\mathbf{x})) = 1 - (\ell/f(\mathbf{x}))$, and this function indicates the number of swaps between vertices in $\mathbf{x}$. The mutation operator chooses randomly $M(f(\mathbf{x}))$ times two vertices i and j in $\mathbf{x}$ and then swaps them. The hypermutation function from population P^{clo} generates the population P^{hyp} . The cell receptor mutation mechanism is modeled by the mutation number M , which is inversely proportional to the fitness function value. The cloning expansion phase triggers the growth of a new population of high-value B cells centered around a higher fitness function value.

In the *Aging phase*, after the evaluation of P^{hyp} at time t , the algorithm eliminates old B cells. Such an elimination process is stochastic, and, specifically, the probability to remove a B cell is governed by an exponential negative law with parameter τ_B , (expected mean life for the B cells): $P_{die}(\tau_B) = (1 - e^{(-\ln(2)/\tau_B)})$. Finally, the new population $P^{(t+1)}$ of d elements is produced. We can use two kinds of Aging phases: *pure aging phase* and *elitist aging phase*. In the elitist aging, when a new population for the next generation is generated, we do not allow the elimination of B cells with the best fitness function. While in the pure aging the best B cells can be eliminate as well. We observe that the exponential rate of aging, $P_{die}(\tau_B)$, and the cloning potential, $V(f(\mathbf{x}))$, are inspired by biological processes [16].

Sometimes it might be useful to apply a *birth phase* to increase the population diversity. This extra phase must be combined with an aging phase with a longer expected mean life τ_B . For the GCP we did not use the birth phase because it produced a higher number of fitness function evaluation to solutions.

Assignment colors. To assign colors, the vertices of the solution represented by a B cell are examined and assigned colors, following a deterministic scheme based on the order in which the graph vertices are visited. In details, vertices are examined according to the order given by the B cell and assigned the first color not assigned to adjacent vertices. This method is very simple. In literature there are more complicated and effective methods [5,6,10]. We do not use those methods because we want investigate the learning and solving capability of our IA. In fact, the IA described does not use specific domain knowledge and does not make use of problem-dependent local searches. Thus, our IA can be improved simply including *ad hoc* local search and immunological operators using specific domain knowledge.

2.1 Termination Condition by Information Gain

To analyze the learning process, we use the notion of *Kullback information*, also called *information gain* [17], an entropy function associated to the quantity of information the system discovers during the learning phase. To this end, we define the B cells distribution function $f_m^{(t)}$ as the ratio between the number, B_m^t , of B cells at time t with fitness function value m , (the distance m from the antigen–problem) and the total number of B cells:

$$f_m^{(t)} = \frac{B_m^t}{\sum_{m=0}^h B_m^t} = \frac{B_m^t}{d}. \quad (1)$$

It follows that the information gain can be defined as:

$$K(t, t_0) = \sum_m f_m^{(t)} \log(f_m^{(t)} / f_m^{(t_0)}). \quad (2)$$

The gain is the amount of information the system has already learned from the given Ag-problem with respect to initial distribution function (the randomly generated initial population $P^{(t_0=0)}$). Once the learning process starts, the information gain increases monotonically until it reaches a final steady state (see figure 1). This is consistent with the idea of a *maximum information-gain principle* of the form $\frac{dK}{dt} \geq 0$. Since $\frac{dK}{dt} = 0$ when the learning process ends, we use it as a termination condition for the Immune Algorithms. We will see in section 3 that the information gain is a kind of entropy function useful to understand the IA's behavior and to set the IA's parameters.

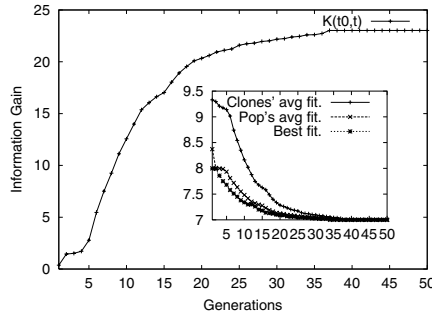


Fig. 1. Information Gain versus generations for the GCP instance QUEEN6_6.

In figure 1 we show the information gain when the IA faces the GCP instance QUEEN6_6 with vertex set $|V| = 36$, edge set $|E| = 290$ and optimal coloring 7. In particular, in the inset plot one can see the corresponding average fitness of population P^{hyp} , the average fitness of population $P^{(t+1)}$ and the best fitness value. All the values are averaged on 100 independent runs. Finally, we note that our experimental protocol can have other termination criteria, such as maximum number of evaluations or generations.

2.2 Local Search

Local search algorithms for combinatorial optimization problems generally rely on a definition of neighborhood. In our case, neighbors are generated by swapping vertex values. Every time a proposed swap reduces the number of used colors, it is accepted and we continue with the sequence of swaps, until we explore the neighborhood of all vertices. Swapping all pair of vertices is time consuming, so we use a reduced neighborhood: all $n = |V|$ vertices are tested for a swap, but only with the closer ones. We define a neighborhood with radius R . Hence

we swap all vertices only with their R nearest neighbors, to left and to right. A possible value for radius R is 5. Given the large size of neighborhood and n , we found it convenient to apply the previous local search procedure only on the population's best B cell. We note that if $R = 0$ the local search procedure is not executed. This case is used for simple GCP instances, to avoid unnecessary fitness function evaluations. The local search used is not critical to the searching process. Once a maximum number of generations has been fixed, the local search procedure increases only the success rate on a certain number of independent runs and, as drawback, it increases the average number of evaluations to solutions. However, if we omit it, the IA needs more generations, hence more fitness function evaluations, to obtain the same results of IA using local search.

Table 1. Pseudo-code of Immune Algorithm

Immune Algorithm(d, dup, τ_B, R)	
1. $t := 0$;	
2. Initialize $P^{(0)} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_d\} \in S^\ell$	
3. while ($\frac{dK}{dt} \neq 0$) do	
4. Interact(Ag, $P^{(t)}$);	/* Interaction phase */
5. $P^{clo} :=$ Cloning ($P^{(t)}, dup$);	/* First step Cloning expansion */
6. $P^{hyp} :=$ Hypermutation (P^{clo});	/* Second step Cloning expansion */
7. Evaluate (P^{hyp});	/* Compute P^{hyp} fitness function */
8. $P^{ls} :=$ Local_Search(P^{hyp}, R);	/* LS procedure */
9. $P^{(t+1)} :=$ aging($P^{hyp} \sqcup P^{(t)} \sqcup P^{ls}, \tau_B$);	/* Aging Phase */
10. $K(t, t_0) :=$ InformationGain();	/* Compute $K(t, t_0)$ */
11. $t := t + 1$;	
12. end_while	

In figure 2 we show the fitness function value dynamics. In both plots, we show the dynamics of average fitness of population P^{hyp} , $P^{(t+1)}$, and the best fitness value of population $P^{(t+1)}$. Note that the average fitness of P^{hyp} shows the diversity in the current population, when this value is equal to average fitness of population $P^{(t+1)}$, we are close a premature convergence or in the best case we are reaching a sub-optimal or optimal solution. It is possible to use the difference between P^{hyp} average fitness and $P^{(t+1)}$ average fitness, $|avgfitness(P^{hyp}) - avgfitness(P^{(t+1)})| = Pop_{div}$ as a standard to measure population diversity. When Pop_{div} rapidly decreases, this is considered as the primary reason for premature convergence. In the left plot we show the IA dynamic when we face the DSCJ250.5.COL GCP instance ($|V| = 250$ and $|E| = 15,668$). We execute the algorithm with population size $d = 500$, duplication parameter $dup = 5$, expected mean life $\tau_B = 10.0$ and neighborhood's radius $R = 5$. For this instance we use *pure aging* and obtain the optimal coloring. In the right plot

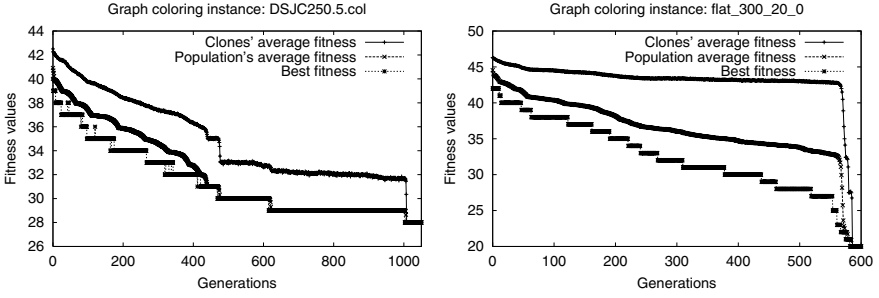


Fig. 2. Average fitness of population P^{hyp} , average fitness of population $P^{(t+1)}$, and best fitness value vs generations. Left plot: IA with pure aging phase. Right plot: IA with elitist aging

we tackle the FLAT_300_20 GCP instance ($|V| = 300$ and $|E| = 21,375$), with the following IA's parameters: $d = 1000$, $dup = 10$, $\tau_B = 10.0$ and $R = 5$. For this instance the optimal coloring is obtained using *elitist aging*. In general, with elitist aging the convergence is faster, even though it can trap the algorithm in a local optimum. Although, with pure aging the convergence is slower and the population diversity is higher, our experimental results indicate that elitist aging seems to work well. We can define the ratio $S_p = \frac{1}{dup}$ as the *selective pressure* of the algorithm: when $dup = 1$, obviously we have that $S_p = 1$ and the selective pressure is low, while increasing dup we increase the IA's selective pressure. Experimental results show that high values of d denote high clones population average fitness and, in turn, high population diversity but, also, a high computational effort during the evolution.

3 Parameters Tuning by Information Gain

To understand how to set the IA parameters, we performed some experiments it with the GCP instance QUEEN6_6. Firstly, we want to set the B cell's mean life, τ_B . We fix the population size $d = 100$, duplication parameter $dup = 2$, local search radius $R = 2$ and total generations $gen = 100$. For each experiment we performed $runs = 100$ independent runs.

3.1 B Cell's Mean Life, τ_B

In figure 3 we can see the best fitness values (left plot) and the Information Gain (right plot) with respect the following τ_B values $\{1.0, 5.0, 15.0, 25.0, 1000.0\}$. When $\tau_B = 1.0$ the B cells have a shorter mean life, only one time step, and with this value the IA performed poorly. With $\tau_B = 1.0$ the maximum information gain obtained at generation 100 is about 13. As τ_B increases, the best fitness values decreases and the Information Gain increases. The best value for τ_B is 25.0. With $\tau_B = 1000.0$, and in general when τ_B is greater than a number of fixed

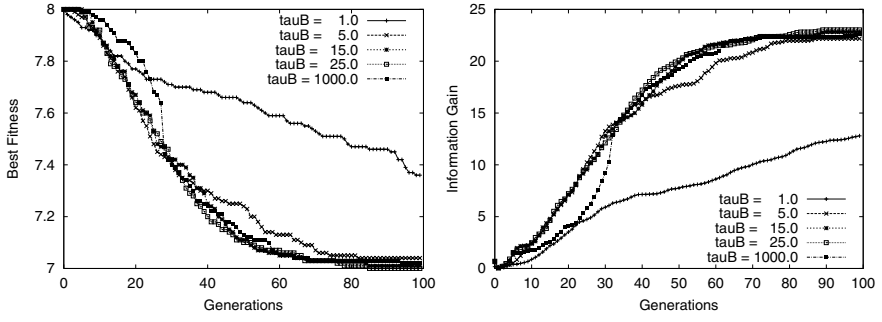


Fig. 3. Best fitness values and Information Gain vs generations.

generations gen , we can consider B cells mean life infinite and obtain a pure elitist selection scheme. In this special case, the behavior of IA shows slower convergence in the first 30 generations in both plots. For values of τ_B greater than 25.0 we obtain slightly worse results. Moreover, when $\tau_B \leq 10$ the success rate (SR) on 100 independent runs is less than 98 while when $\tau_B \geq 10$ the IA obtains a SR=100 with a lower Average number of Evaluations to Solution (AES) located when $\tau_B = 25.0$.

3.2 Duplication Parameter Dup

Now we fix $\tau_B = 25.0$ and vary dup . In fig.4 (left plot) we note that the IA obtains quickly more Information Gain at each generation with $dup = 10$, moreover it reaches faster the best fitness value with $dup = 5$. With both values of dup the

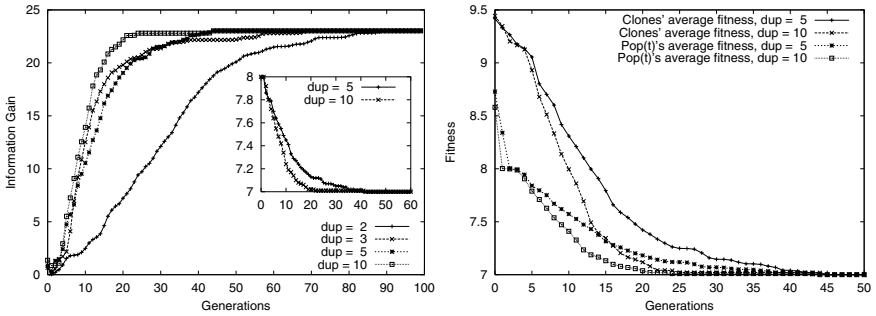


Fig. 4. Left plot, Information Gain and Best fitness value for dup . Right plot, average fitness of Clones and $Pop(t)$ for $dup \in \{5, 10\}$.

largest information gain is obtained at generation 43. Moreover, with $dup = 10$ the best fitness is obtained at generation 22, whereas with $dup = 5$ at generation 40. One may deduce that $dup = 10$ is the best value for the cloning of B cells

since we obtain faster more information gain. This is not always true. Indeed, if we observe figure 4 (right plot) we can see how the IA with $dup = 5$ obtains a larger amount of clones average fitness and hence a greater diversity. This characteristic can be useful in avoiding premature convergence and in finding more optimal solutions for a given combinatorial problem.

3.3 Dup and τ_B

In 3.1 we saw that for $dup = 2$, the best value of τ_B is 25.0. Moreover, in 3.2 experimental results show better performance for $dup = 5$. If we set $dup = 5$ and vary τ_B , we obtain the results in fig.5. We can see that for $\tau_B = 15$ we reach the maximum Information Gain at generation 40 (left plot) and more diversity (right plot). Hence, when $dup = 2$ the best value of τ_B is 25.0, i.e. on average we need 25 generations for the B cells to reach a mature state. On the other hand, when $dup = 5$ the correct value is 15.0. Thus, increasing dup the average time for the population of B cells to reach a mature state decreases.

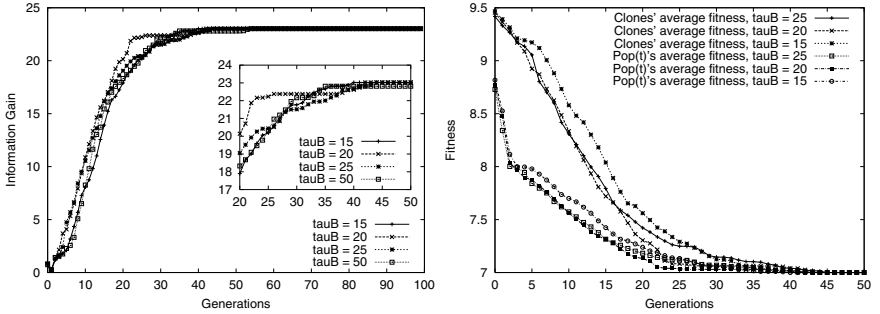


Fig. 5. Left plot Information Gain for $\tau_b \in \{15, 20, 25, 50\}$. Right plot average fitness of population P^{hyp} and population $P^{(t)}$ for $\tau_b \in \{15, 20, 25\}$

3.4 Neighborhood's Radius R, d and Dup

Local search is useful for large instances (see table 2). The cost of local search, though, is high. In figure 6 (left plot) we can see how the AES increases as the neighborhood radius increases. The plot reports two classes of experiments performed with 1000 and 10000 independent runs. In figure 6 (right plot) we show the values of parameters d and dup as functions of the Success Rate (SR). Each point has been obtained averaging 1000 independent runs. How we can see there is a certain relation between d and dup in order to reach a $SR = 100$. For the QUEEN6_6 instance, for low values for the population we need a high value of dup to reach $SR = 100$. For $d = 10$, $dup = 10$ is not sufficient to obtain the maximum SR. On the other hand, as the population number increases, we need smaller values for dup . Small values of dup are a positive factor.

Table 2. Mycielsky and Queen graph instances. We fixed $\tau_B = 25.0$, and the number of independent runs 100. *OC* denotes the Optimal Coloring.

<i>Instance G</i>	$ V $	$ E $	OC	(d,dup,R)	Best Found	AES
Myciel3	11	20	4	(10,2,0)	4	30
Myciel4	23	71	5	(10,2,0)	5	30
Myciel5	47	236	6	(10,2,0)	6	30
Queen5_5	25	320	5	(10,2,0)	5	30
Queen6_6	36	580	7	(50,5,0)	7	3750
Queen7_7	49	952	7	(60,5,0)	7	11,820
Queen8_8	64	1,456	9	(100,15,0)	9	78,520
Queen8_12	96	2,736	12	(500,30,0)	12	908,000
Queen9_9	81	1,056	10	(500,15,0)	10	445,000
School1_nsh	352	14,612	14	(1000,5,5)	15	2,750,000
School1	385	19,095	9	(1000,10,10)	14	3,350,000

We recall that *dup* is similar to the temperature in Simulated Annealing [18]. Low values of *dup* corresponds to a system that cools down slowly and has a high *EAS*.

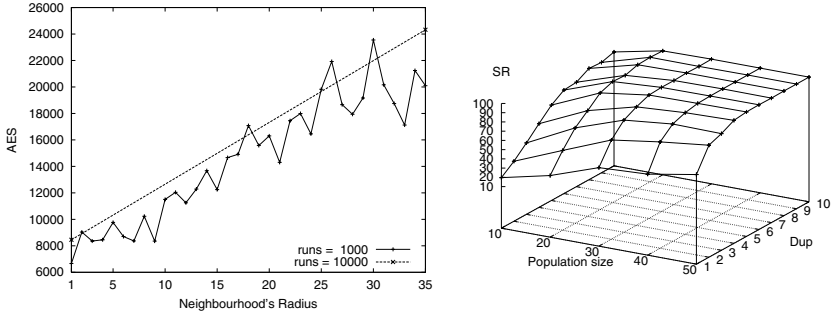


Fig. 6. Left plot: Average number of Evaluations to Solutions versus neighborhood's radius. Right plot: 3D plot of d , dup versus Success Rate (SR).

4 Results

In this section we report our experimental results. We worked with classical benchmark graph [10]: the MYCIELSKI, QUEEN, DSJC and LEIGHTON GCP instances. Results are reported in Tables 2 and 3. In these experiments the IA's best found value is always obtained $SR = 100$. For all the results presented in this section, we used *elitist aging*. In tables 4 and 5 we compare our IA with two of the best evolutionary algorithms, respectively Evolve_AO algorithm [19] and the

Table 3. Experimental results on subset instances of DSJC and Leighton graphs. We fixed $\tau_B = 15.0$, and the number of independent runs 10.

<i>Instance</i> G	$ V $	$ E $	OC	(d,dup,R)	Best Found	AES
DSJC125.1	125	736	5	(1000,5,5)	5	1,308,000
DSJC125.5	125	3,891	12	(1000,5,5)	18	1,620,000
DSJC125.9	125	6,961	30	(1000,5,10)	44	2,400,000
DSJC250.1	250	3,218	8	(400,5,5)	9	1,850,000
DSJC250.5	250	15,668	13	(500,5,5)	28	2,500,000
DSJC250.9	250	27,897	35	(1000,15,10)	74	4,250,000
le450_15a	450	8,168	15	(1000,5,5)	15	5,800,000
le450_15b	450	8,169	15	(1000,5,5)	15	6,010,000
le450_15c	450	16,680	15	(1000,15,10)	15	10,645,000
le450_15d	450	16,750	9	(1000,15,10)	16	12,970,000

HCA algorithm [5]. For all the GCP instances we ran the IA with the following parameters: $d = 1000$, $dup = 15$, $R = 30$, and $\tau_B = 20.0$. For these classes of experiments the goal is to obtain the best possible coloring, no matter the value of AES. Table 4 shows how the IA outperform the Evolve_AO algorithm, while is similar in results to HCA algorithm and better in SR values (see table 5).

Table 4. IA versus Evolve_AO Algorithm. The values are averaged on 5 independent runs.

<i>Instance</i> G	$\chi(G)$	Best-Known	Evolve_AO	IA	Difference
DSJC125.5	12	12	17.2	18.0	+ 0.8
DSJC250.5	13	13	29.1	28.0	-0.9
flat300_20_0	≤ 20	20	26.0	20.0	-6.0
flat300_26_0	≤ 26	26	31.0	27.0	-4.0
flat300_28_0	≤ 28	29	33.0	32.0	-1.0
le450_15a	15	15	15.0	15.0	0
le450_15b	15	15	15.0	15.0	0
le450_15c	15	15	16.0	15.0	-1.0
le450_15d	15	15	19.0	16.0	-3.0
mulsol.i.1	–	49	49.0	49.0	0
school1_nsh	≤ 14	14	14.0	15.0	+1.0

5 Conclusions

We have designed a new IA that incorporates a simple local search procedure to improve the overall performances to tackle the GCP instances. The IA presented has only four parameters. To set correctly these parameters we use the Information Gain function, a particular entropy function useful to understand

Table 5. IA versus Hao *et al.*'s HCA algorithm. The number of independent runs is 10.

<i>Instance G</i>	HCA's Best-Found and (SR)	IA's Best-Found and (SR)
DSJC250.5	28 (90)	28 (100)
flat300_28.0	31 (60)	32 (100)
le450_15c	15 (60)	15 (100)
le450_25c	26 (100)	25 (100)

the IA's behavior. The Information Gain measures the quantity of information that the system discovers during the learning process. We choose the parameters that maximize the information discovered and that increases moderately the information gain monotonically. To our knowledge, this is the first time that IAs, and in general the EAs, are characterized in terms of information gain. We define the average fitness of population P^{hyp} as the diversity in the current population, when this value is equal to average fitness of population $P^{(t+1)}$, we are close a premature convergence. Using a simple coloring method we have investigated the IA's learning and solving capability. The experimental results show how the proposed IA is comparable to and, in many GCP instances, outperforms the best evolutionary algorithms. Finally, the designed IA is directed to solving GCP instances although the solutions' representation and the variation operators are applicable more generally, for example Travelling Salesman Problem.

Acknowledgments. The authors wish to thank the anonymous referees for their excellent revision work. GN wishes to thank the University of Catania project "Young Researcher" for partial support and is grateful to Prof. A. M. Anile for his kind encouragement and support.

References

1. Dasgupta, D. (ed.): Artificial Immune Systems and their Applications. Springer-Verlag, Berlin Heidelberg New York (1999)
2. De Castro L.N., Timmis J.: Artificial Immune Systems: A New Computational Intelligence Paradigm. Springer-Verlag, UK (2002)
3. Forrest, S., Hofmeyr, S. A.: Immunology as Information Processing. Design Principles for Immune System & Other Distributed Autonomous Systems. Oxford Univ. Press, New York (2000)
4. Nicosia, G., Castiglione, F., Motta, S.: Pattern Recognition by primary and secondary response of an Artificial Immune System. *Theory in Biosciences* **120** (2001) 93–106
5. Galinier, P., Hao, J.: Hybrid Evolutionary Algorithms for Graph Coloring. *Journal of Combinatorial Optimization* Vol. 3 **4** (1999) 379–397
6. Marino, A., Damper, R.I.: Breaking the Symmetry of the Graph Colouring Problem with Genetic Algorithms. *Workshop Proc. of the Genetic and Evolutionary Computation Conference (GECCO'00)*. Las Vegas, NV: Morgan Kaufmann (2000)

7. Garey, M.R., Johnson, D.S.: *Computers and Intractability: a Guide to the Theory of NP-completeness*. Freeman, New York (1979)
8. Mehrotra, A., Trick, M.A.: A Column Generation Approach for Graph Coloring. *INFORMS J. on Computing* 8 (1996) 344–354
9. Caramia, M., Dell’Olmo, P.: Iterative Coloring Extension of a Maximum Clique. *Naval Research Logistics*, 48 (2001) 518–550
10. Johnson, D.S., Trick, M.A. (eds.): *Cliques, Coloring and Satisfiability: Second DIMACS Implementation Challenge*. American Mathematical Society, Providence, RI (1996)
11. De Castro, L. N., Von Zuben, F. J.: The Clonal Selection Algorithm with Engineering Applications. *Proceedings of GECCO 2000, Workshop on Artificial Immune Systems and Their Applications*, (2000) 36–37
12. De Castro, L.N., Von Zuben, F.J.: Learning and optimization using the clonal selection principle. *IEEE Trans. on Evolutionary Computation* Vol. 6 **3** (2002) 239–251
13. Nicosia, G., Castiglione, F., Motta, S.: Pattern Recognition with a Multi-Agent model of the Immune System. *Int. NAISO Symposium (ENAIIS’2001)*. Dubai, U.A.E. ICSC Academic Press, (2001) 788–794
14. Eiben, A.E., Hinterding, R., Michalewicz, Z.: Parameter control in evolutionary algorithms. *IEEE Trans. on Evolutionary Computation*, Vol. 3 **2** (1999) 124–141
15. Leung, K., Duan, Q., Xu, Z., Wong, C.W.: A New Model of Simulated Evolutionary Computation – Convergence Analysis and Specifications. *IEEE Trans. on Evolutionary Computation* Vol. 5 **1** (2001) 3–16
16. Seiden P.E., Celada F.: A Model for Simulating Cognate Recognition and Response in the Immune System. *J. Theor. Biol.* Vol. 158 (1992) 329–357
17. Nicosia, G., Cutello, V.: Multiple Learning using Immune Algorithms. *Proceedings of the 4th International Conference on Recent Advances in Soft Computing, RASC 2002*, Nottingham, UK, 12–13 December (2002)
18. Johnson, D.R., Aragon, C.R., McGeoch, L.A., Schevon, C.: Optimization by simulated annealing: An experimental evaluation; part II, graph coloring and number partitioning. *Operations Research* 39 (1991) 378–406
19. Barbosa, V.C., Assis, C.A.G., do Nascimento, J.O.: Two Novel Evolutionary Formulations of the Graph Coloring Problem. *Journal of Combinatorial Optimization* (to appear)

MILA – Multilevel Immune Learning Algorithm

Dipankar Dasgupta, Senhua Yu, and Nivedita Sumi Majumdar

Computer Science Division, University of Memphis, Memphis, TN 38152, USA
{dasgupta, senhuayu, nmajumdr}@memphis.edu

Abstract. The biological immune system is an intricate network of specialized tissues, organs, cells, and chemical molecules. T-cell-dependent humoral immune response is one of the complex immunological events, involving interaction of B cells with antigens (Ag) and their proliferation, differentiation and subsequent secretion of antibodies (Ab). Inspired by these immunological principles, we proposed a Multilevel Immune Learning Algorithm (MILA) for novel pattern recognition. It incorporates multiple detection schema, clonal expansion and dynamic detector generation mechanisms in a single framework. Different test problems are studied and experimented with MILA for performance evaluation. Preliminary results show that MILA is flexible and efficient in detecting anomalies and novelties in data patterns.

1 Introduction

The biological immune system is of great interest to computer scientists and engineers because it provides a unique and fascinating computational paradigm for solving complex problems. There exist different computational models inspired by the immune system. A brief survey of some of these models may be found elsewhere [1]. Forrest et al. [2–4] developed a *negative-selection algorithm* (NSA) for change detection based on the principles of self-nonsel self discrimination. This algorithm works on similar principles, generating detectors randomly, and eliminating the ones that detect self, so that the remaining detectors can detect any non-self. If any detector is ever matched, a change (non-self) is known to have occurred. Obviously, the first phase is analogous to the censoring process of T cells maturation in the immune system. However, the monitoring phase is logically (not biologically) derivable.

The biological immune system employs a multilevel defense against invaders through nonspecific (innate) and specific (adaptive) immunity. The problems for anomaly detection also need multiple detection mechanisms to obtain a very high detection rate with a very low false alarm rate. The major limitation of binary NSA is that it generates a higher false alarm rate when applied to anomaly detection for some data sets. To illustrate this limitation, some patterns, for example, 110, 100, 011, 001, are considered as normal samples. Based on these normal samples, 101, 111, 000, 010 become abnormal. A partial matching rule is usually used to generate a set of detectors. As described in [5], with matching threshold ($r = 2$), two strings (one represents

candidate detector, another is a pattern) match if and only if they are identical in at least 2 contiguous positions. Because the detector must fail to match any string in normal samples, for the above example, the detectors cannot be generated at all, and consequently anomalies cannot be detected; except for $r = 3$ (length of the string), which results in exact match and requires all non-self strings as detectors.

In order to alleviate these difficulties, we proposed an approach, called *Multilevel Immune Learning Algorithm (MILA)*. There are several features which distinguish this algorithm from the NSA; in particular, multilevel detection and immune memory. In this paper, we describe this approach and show the advantages of using new features of *MILA* in the application of anomaly detection.

The layout of this paper is as follows. Section 2 outlines the proposed algorithm. Section 3 briefly describes the application of MILA to anomaly detection. Section 4 reports some experimental results with different testing problems. Section 5 discusses new features of *MILA* indicated in the application of anomaly detection. Section 6 provides concluding remarks.

2 Multilevel Immune Learning Algorithm (MILA)

This approach is inspired by the interaction and processes of T cell-dependent humoral immune response. In biological immune systems, some B cells recognize antigens (foreign protein) via immunoglobulin receptors on their surface but are unable to proliferate and differentiate unless prompted by the action of lymphokines secreted by T helper cells. Moreover, in order for T helper cells to become stimulated to release lymphokines, they must also recognize specific antigens. However, while T helper cells recognize antigens via their receptors, they can only do so in the context of MHC molecules. Antigenic peptides must be extracted by several types of cells called antigen-presenting cells (APCs) through a process called “Ag presentation.” Under certain conditions, however, B-cell activation is suppressed by T suppressor cells, but specific mechanisms for such suppression are yet unknown. The activated B cells and T cells migrate to the primary follicle of the cortex in lymph nodes, where a complex interaction of the basic cell kinetic process of proliferation (cloning), mutation, selection, differentiation, and death of B-cells occurs through germinal center reaction [6] and finally secretes antibodies. These antibodies function as effectors to the humoral response by binding to antigens and facilitating their elimination. The proposed artificial immune system is an abstract of complex multistage immunological events in humoral immune response. The algorithm consists of initialization phase, recognition phase, evolutionary phase and response phase. As shown in Fig.2, the main features of each phase can be summarized as follows:

- In *initialization* phase, the detection system is “trained” by giving the knowledge of “self”. The outcome of the initialization is to generate sets of detectors, analogous to the populations of T helper cells (T_h), T suppressor cells (T_s) and B cells, which participate in T cell dependent humoral immune response.

- In *recognition* phase, B cells, together with T cells (T_h , T_s) and antigen presenting cells (APCs), form a multilevel recognition. APC is an extreme high-level detector, which acts as a default detector (based on environment) identifying visible damage signals from the system. For example, while monitoring a computer system, screen turning black, too many lining-up printing jobs and so on may provide visible signals captured by APC. Thus, APC is not defined based on particular normal behavior in input data. It is to be noted that T cells and B cells recognize antigens at different levels. The recognition of T_h is defined as a bit-level (lowest level) recognition, such as using consecutive windows of data pattern. Importantly, B cells in the immune system only recognize particular sites called *epitope* on the surface of the antigen, as shown in Fig.1. Clearly, the recognition (matching) sites are not contiguous when we stretch out the 3-dimension folding of the antigen protein. Thus, the B cell is considered as feature-level recognition at different non-contiguous (occasionally contiguous) positions of antigen strings. Accordingly, MILA can provide multilevel detection in hierarchical fashion, starting with APC detection, B-cell detection and T-cell detection. However, T_s acts as suppression and is problem dependent. As shown in fig. 2, the logical operator can be set to $\wedge$ (AND) or $\vee$ (OR) to make the system more fault-tolerant or more sensitive as desired.
- In *evolutionary* phase, the activated B cells clone to produce memory cells and plasma cells. Cloning is subject to very high mutation rates called somatic hypermutation with a selective pressure. In addition to passing *negative selection*, for each progeny of the activated B cell (parent B cell), only the clones with higher affinity are selected. This process is known as *positive selection*. The outcome of evolutionary phase is to generate high-quality detectors with specificity to the exposed antigens for future use.
- *Response* phase involves primary response to initial exposure and secondary response to the second encounter.

Accordingly, the above mechanism steps, as shown in the Fig.2, give a general description of MILA, however, based on applications and timeliness of execution, some detection phase may not be considered.

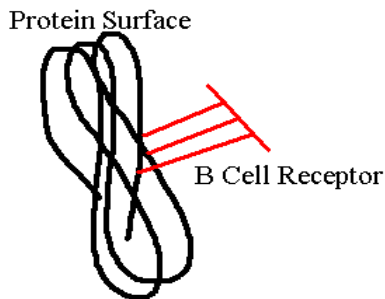


Fig. 1. B Cell Receptor Matches an antigenic protein in its surface

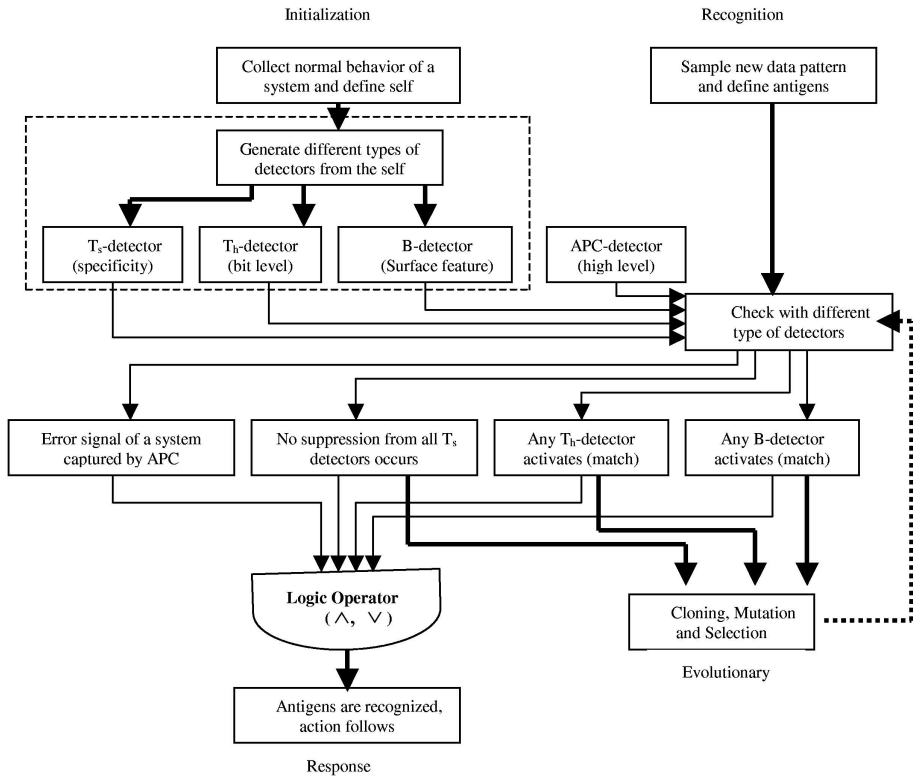


Fig. 2. Overview of Multilevel Immune Learning Algorithm (MILA)

3 Application of MILA to Anomaly Detection Problems

Detecting anomaly in a system or in a process behavior is very important in many real-world applications. For example, high-speed milling processes require continuous monitoring to assure high quality production; jet engines also require continuous monitoring to assure safe operation. It is essential to detect the occurrence of unnatural events as quickly as possible before any significant performance degradation results [5]. There are many techniques for anomaly detection, and depending on application domains, these are referred to as novelty detection, faulty detection, surprise pattern detection, etc. Among these approaches, the detection algorithm with better discrimination ability will have a higher detection rate. In particular, it can accurately discriminate the normal data and the observed data during monitoring. The decision-making systems for detection usually depend on learning the behavior of the monitored environment from a set of *normal (positive)* data. By normal, we mean usage data that have been collected during the normal operation of the system or a process. In order to evaluate the performance, MILA is applied to the anomaly detection prob-

lem. For this problem, the following assumptions are made to simplify the implementation:

- In *Initialization* phase and *Recognition* phase, T_s detectors employ more stringent threshold than T_h detectors and B detectors.
- T_s detector is regarded as a special self-detecting agent. In *Initialization* phase, T_s detector will be selected if it still matches the self-antigen under more stringent threshold, whereas in *Recognition* phase the response will be terminated when T_s detector matches a special antigen resembling self-data pattern. Similar to T_h and B cells, the activated T_s detector undergoes cloning and positive selection after being activated by a special Ag.
- APC-detectors, as shown in Fig.2, are not used in this application.
- The lower the antigenic affinity, the higher the mutation rate. From a computational perspective, the purpose of this assumption is to increase the probability of producing effective detectors.
- For each parent cloning, only ONE clone whose affinity is the highest among all clones is kept. The selected clone will be discarded if it is similar to the existing detectors. This assumption solves the problem using minimal resources without compromising the detection rate.
- Currently, the response phase is dummy as we are only dealing with anomaly detection tasks.

This application employs a distance measure (Euclidean distance) to calculate the affinity between the detector and the self/nonself data pattern along with a partial matching rule. Overall, the implementation of MILA for anomaly detection can be summarized as follows:

1. Collect *Self* data sufficient to exhibit the normal behavior of a system and choose a technique to normalize the raw data.
2. Generate different types of detectors, e.g., B, T_h , T_s detectors. T_h and B detectors should not match any of self-peptide strings according to the partial matching rule. The sliding window scheme [5] is used for T_h partial matching. The random position pick-up scheme is used for B partial matching. For example, suppose that a self string is $\langle s_1, s_2, \dots, s_L \rangle$ and the window size is chosen as 3, then the self peptide strings can be $\langle s_1, s_3, s_L \rangle$, $\langle s_2, s_4, s_9 \rangle$, $\langle s_5, s_7, s_8 \rangle$ and so on by randomly picking up the attribute at some positions. If the candidate B detector represented as $\langle m_1, m_2, m_3 \rangle$ fails to match Any self-feature indexed as $\langle 1, 3, L \rangle$ in self-data patterns, the candidate B detector is selected and represented as $\langle (1, m_1), (3, m_2), (L, m_3) \rangle$. Two important parameters, T_h threshold and B threshold, are employed to measure the matching. If the value for the distance between the T_h (or B) detector and the self string is greater than T_h (or B) threshold, then it is considered as *matching*. T_s detector, however, is selected if it can match the special self strings by employing more stringent suppressor threshold called T_s threshold.

3. When monitoring the system, the logical operator shown in Fig.1 is chosen as “AND ($\wedge$)” in this application. The unseen pattern is tested by T_h , T_s , B detector, respectively. If any T_h and B detector is ever activated (matched with current pattern) and all of the T_s detectors are not activated, a change in behavior pattern is known to have occurred and an alarm signal is generated indicating an abnormality. The same matching rule is adopted as used in generating detectors. We calculate the distance between the T_h / T_s detector and the new sample as described in [5]. B detector is actually an information vector with the information of binding sites and values of attributes in these sites. For the B detector $\langle (1, m_1), (3, m_2), (L, m_3) \rangle$ in the above example, if an Ag is represented as $\langle n_1, n_2, n_3, \dots, n_L \rangle$, then the distance is calculated only between points $\langle m_1, m_2, m_3 \rangle$ and $\langle n_1, n_3, n_L \rangle$.
4. Activated T_h , T_s , B detectors are cloned with a high mutation rate and only one clone with the highest affinity is selected. Detectors that are not activated are kept in detector sets.
5. Employ the optimized detectors generated after the detection phase to test the unseen patterns, repeat from step 3.

4 Experiments

4.1 Data Sets

We experimented with different datasets to investigate the performances of *MILA* for detecting anomalous patterns. The paper only reported results of using speech-recording time series dataset (see reference [8]) because of space limitations. We normalized the raw data (total 1025 time steps) at the range 0~1 for training the system. The testing data (total 1025 time steps) are generated that contain anomalies between 500 and 700 and some noise after 700 time steps.

4.2 Performance Measures

Using a sliding (overlapping) window of size L (in our case, $L=13$), if normal series have the values: $x_1, x_2, \dots, x_m$, *self-patterns* are generated as follows:

$$\begin{array}{cccc}
 \langle x_1, & x_2, & \dots & x_L \rangle \\
 \langle x_2, & x_3, & \dots & x_{L+1} \rangle \\
 \vdots & \vdots & \vdots & \vdots \\
 \langle x_{m-L+1}, & x_{m-L+2}, & \dots, & x_m \rangle
 \end{array}$$

Similarly, *Ag-patterns* are generated from the samples shown in Fig.4b. In this experiment, we used real-valued strings to represent Ag and Ab molecules, which is different from binary Negative Selection Algorithm [4, 5, 9] and Clone Selection Principle application [10]. Euclidean distance measure is used to model the complex

chemistry of Ag/Ab recognition as a matching rule. Two measures of effectiveness for detecting anomaly are calculated as follows:

$$\text{Detection rate} = \frac{TP}{TP + FN}$$

$$\text{False alarm rate} = \frac{FP}{TN + FP}$$

Where TP (true positives), anomalous elements identified as anomalous; TN (true negatives), normal elements identified as normal; FP (false positives), normal elements identified as anomalous; FN (false negatives), anomalous elements identified as the normal [11].

The MILA algorithm has a number of tuning parameters. Different detector thresholds that determine whether a new sample is normal or abnormal control the sensitivity of the system. Employing various strategies to change threshold values, different values for detection rate and false alarm rate are obtained that are used for plotting the ROC (Receiver Operating Characteristics) curve, which reflects tradeoff between false alarm rate and detection rate.

4.3 Experimental Results

The following test cases are studied and some results are reported in this paper:

1. For different threshold changing strategies the influence on ROC curves is studied. In this paper, we report the results obtained from three different cases: (1) changing B threshold at fixed T_h threshold (0.05 if B threshold is less than 0.16, otherwise 0.08) and T_s threshold (0.02); (2) changing B threshold at fixed T_h threshold (0.1) and T_s threshold (0.02); (3) changing T_h threshold at fixed B threshold (0.1) and T_s threshold (0.02). The results shown in Fig.3 indicate that the first case obtain a better ROC curve. Therefore, this paper uses this strategy to obtain different values for detection and false alarm rate for *MILA* based anomaly detection.
2. The comparison of performances illustrated in ROC curves between single level detection and multilevel detection (*MILA*) is studied. We experimented and compared the efficiency of anomaly detection in three cases: (1) only using T_h detectors; (2) only using B detectors; (3) combining T_h , T_s , B detectors as indicated in *MILA*. ROC curves in these cases are shown in Fig.4. Moreover, Fig.5 show how detection and false alarm rates change when threshold is modified in these three cases. Since detectors are randomly generated, different values for detection and false alarm rates are observed. Considering this issue, we run the system ten iterations to obtain the average of the values for detection and false alarm rate, as shown in Fig.4 and Fig.5.

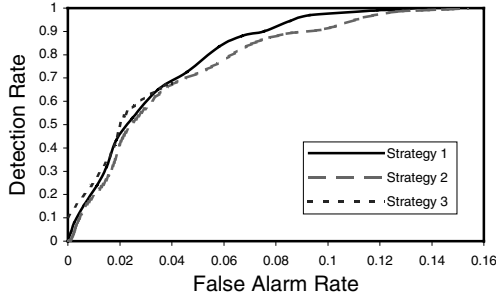


Fig. 3. ROC curves obtained by employing different thresholds changing strategy as described in the section 4.3

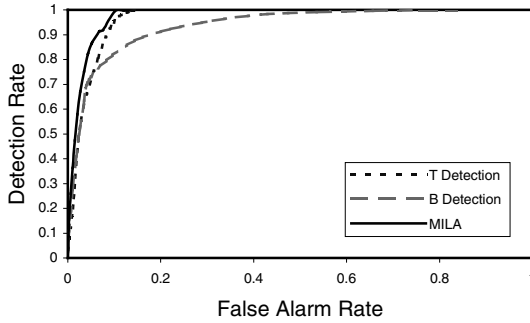


Fig. 4. Comparison of ROC curves between single level detection (e.g., T_h detection or B detection) and multilevel detection (*MILA*)

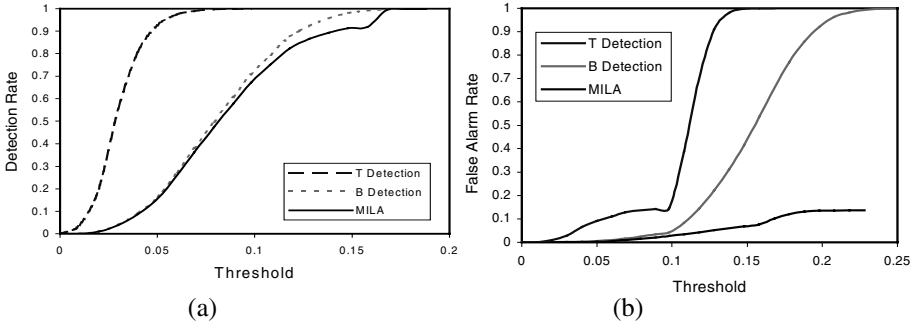


Fig. 5. Evolution of detection rate in Fig. 5(a) and false alarm rate in Fig. 5(b) based on single level detection and multilevel detection (*MILA*) with changing threshold values

3. The efficiency of the detector for detecting anomaly is studied. Once detectors, e.g., T_h detectors, T_s detectors and B detects, are generated in *Initialization* phase, we repeatedly tested the same abnormal samples for 5 iterations

with same parameter settings. Since the detector in MILA undergoes cloning, mutation and selection after Recognition phase, the elements in the detector set changes after each iteration in detecting phase, although the same abnormal samples and conditions are employed in Recognition phase. So, for each iteration, different values for detection and false alarm rate are observed, as shown in Fig. 6 through Fig.7.

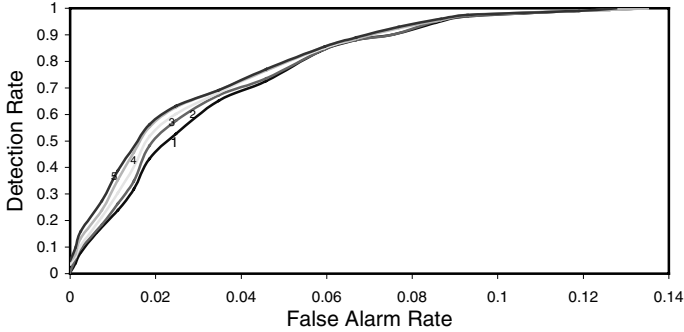


Fig. 6. ROC curves for MILA based anomaly detection in each detecting iteration. 1, 2, 3, ... in the ROC curves denote the iterations of detecting same Ag samples. For each iteration, the detector sets are those that are generated in the detect phase of previous iteration.

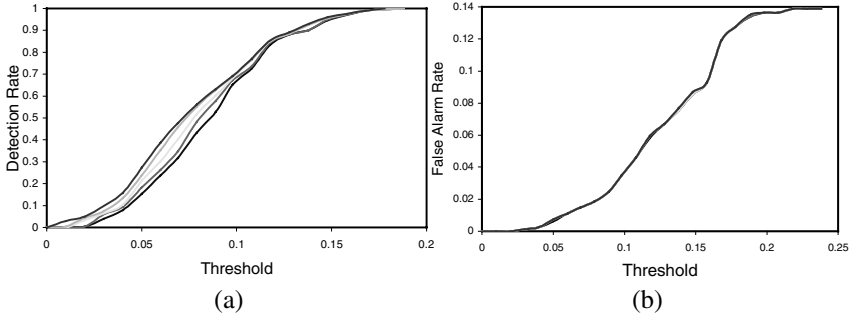


Fig. 7. Evolution of detection rate in Fig. 7(a) and false alarm rate in Fig. 7(b) for MILA based anomaly detection in each detecting iteration as described in Fig.6 when threshold is varied.

5 New Features of MILA

The algorithm presented here takes its inspiration from T-cell-dependent humoral immune response. Considering the application to anomaly detection, one of the key features of MILA is its multilevel detection; that is, multiple strategies are used to generate detectors, which are combined to detect anomalies in new samples.

Preliminary experiments show that MILA is flexible and unique. The generation and recognition of various detectors in this algorithm can be implemented in different ways depending on the application. Moreover, the efficiency of anomaly detection can

be improved by tuning threshold values for different detection scheme. Fig.3 shows this advantage of *MILA* and indicates that better performance can be obtained (shown in the ROC curves) by employing different threshold changing strategies. Compared to the Negative Selection Algorithm (*NSA*), which uses single level detection scheme, Fig.4 shows that the performance of multilevel detection of *MILA* is better. Further results shown in Fig.5 also support the superior performance of *MILA*. Specifically, when comparing multilevel detection (*MILA*) with single detection scheme (*NSA*), the varying trend for detection rate when the threshold is modified is similar as illustrated in Fig.5(a) (at least relative to false alarm rate as shown in Fig.5(b); however, the false alarm rate for multilevel detection (when the threshold is modified) is much lower.

For anomaly detection using *NSA*, the detector set remains constant once generated in the training phase. However, the detector set is dynamic for *MILA* based anomaly detection. *MILA* involves a process of cloning, mutation and selection after successful detection, and some detectors with high affinity for a given anomalous pattern will be selected. This constitutes an on-line learning and detector optimization process. The outcome is to update the detector set and affinity of those detectors that have proven themselves to be valuable by having recognized more frequently occurring anomalies. Fig.6 shows improved performance by using the optimized detector set being generated after the detection phase. This can be explained by the fact that some of the anomalous data employed in our experiment are similar, but generally anomaly is much different from normal series. Thus, when we reduce the distance between a detector and a given abnormal pattern, that is, increase the detector affinity for this pattern, the distances between this detector and other anomalies similar to the given abnormal pattern are also reduced so that those anomalies which formerly failed to be detected by this detector become detectable. However, the distances between the detector and most of the “self”, except for some “self” very similar to “non-self” (anomaly), are still exceeds allowable variation. Therefore, the number of detectors having high affinity increases with the increase in the times of detecting the antigens that are encountered before (at least in a certain range) and thus the detection rate at certain thresholds becomes higher and higher. The experimental results confirm this explanation. Under the same threshold values, Fig.7(a) shows that the detector set produced later has a higher detection rate than the previous detector set, whereas the false alarm rate is almost unchanged as shown in Fig.7(b). In the application to anomaly detection, because of the random generation of the pre-detector, the generated detector set is always different, even if the absolutely same conditions are applied. We cannot guarantee the efficiency of the initial detector set. However, *MILA* based anomaly detection can optimize the detector during on-line detection and thus we can finally obtain more efficient detectors for given samples for monitoring.

As a summary of our proposed principle and initial experiments, the following features of *MILA* have been observed on anomaly detection:

- Unites several different immune system metaphors rather than implementing the immune system metaphors in a piecemeal manner.

- Uses multilevel detection to find and patch the security hole in a large computer system as much as possible.
- MILA is more flexible than single detection scheme (e.g. Negative Selection Algorithm). The implementation for detector generation is problem dependent. More thresholds and parameters may be modified for tuning the system performance.
- Detector set in MILA is dynamic whereas detector set in Negative Selection Algorithm remains constant once it is generated in training phase. MILA involves cloning, mutation and selection after detect phase, which is similar but not equal to Clone Selection Theory. The process of cloning in MILA is targeted (not blind) cloning. Only those detectors that are activated in recognition phase can be cloned. The process of cloning, mutation and selection in MILA is actually a process of detector on-line learning and optimization. Only those clones with high affinity can be selected. This strategy ensures that both the speed and accuracy of detection become successively higher after each detecting.
- MILA is initially inspired by humoral immune response but spontaneously unites the main feature of Negative Selection Algorithm and Clone Selection Theory. It imports their merits but has its own features

6 Conclusions

In this paper, we outlined a proposed change detection algorithm inspired by the T-cell-dependent humoral immune response. This algorithm is called Multilevel Immune Learning Algorithm (*MILA*), which involves four phases: Initialization phase, Recognition phase, Evolutionary phase and Response phase. The proposed method is tested with an anomaly detection problem. *MILA* based anomaly detection is characterized by multilevel detection and on-line learning technique. Experimental results show that *MILA* based anomaly detection is flexible and the detection rate can be improved at the range of allowable false alarm rate by applying different threshold changing strategies. In comparison with single level based anomaly detection, the performance of MILA is clearly better. Experimental results show that detectors have been optimized during the on-line testing phase as well. Moreover, by using different logical operators, it is possible to make the system very sensitive to any changes or robust to noise. Reducing complexity of the algorithm, proposing appropriate suppression mechanism, implementing response phase and experimenting with different data sets are the main directions of our future work.

Acknowledgement. This work is supported by the Defense Advanced Research Projects Agency (no. F30602-00-2-0514). The authors would like to thank the source of the datasets: Keogh, E. & Folias, T. (2002). The UCR Time Series Data Mining Archive [<http://www.cs.ucr.edu/~eamonn/TSDMA/index.html>]. Riverside CA. University of California – Computer Science & Engineering Department.

References

1. Dasgupta, D., Attah-Okine, N.: Immunity-Based Systems: A Survey. In the proceedings of the *IEEE International Conference on Systems, Man and Cybernetics*, Orlando, October 12–15, 1997
2. Forrest, S., Hofmeyr, S., Somayaji, A.: Computer Immunology. *Communications of the ACM* **40**(10) (1997) pp 88–96.
3. Forrest, S., Somayaji, A., Ackley, D.: Building Diverse Computer Systems. *Proc. of the Sixth Workshop on Hot Topics in Operating Systems* (1997).
4. Forrest, S., Perelson, A. S., Allen, L., Cherukuri, R.: Self-nonself discrimination in a computer. *Proc. of the IEEE Symposium on Research in Security and Privacy*, IEEE Computer Society Press, Los Alamitos, CA, (1994) 202–212
5. Dasgupta, D., Forrest, S.: An Anomaly Detection Algorithm Inspired by the Immune System. In: Dasgupta D (eds) *Artificial Immune Systems and Their Applications*, Springer-Verlag, (1999) 262–277
6. Hollowood, K., Goodlad, J.R.: Germinal centre cell kinetics. *J. Pathol.* **185**(3) (1998) 229–33
7. Perelson, A. S., Oster, G. F.: Theoretical studies of clonal selection: Minimal antibody repertoire size and reliability of self- non-self discrimination. *J. Theor.Biol.* **81**(4) (1979) 645–670
8. Keogh, E., Folias, T.: The UCR Time Series Data Mining Archive [<http://www.cs.ucr.edu/~eamonn/TSDMA/index.html>]. Riverside CA. University of California – Computer Science & Engineering Department. (2002)
9. D'haeseleer, P., Forrest, S., Helman, P.: An immunological approach to change detection: algorithms, analysis, and implications. *Proceedings of the 1996 IEEE Symposium on Computer Security and Privacy*, IEEE Computer Society Press, Los Alamitos, CA, (1996) 110–119
10. de Castro, L. N., Von Zuben, F. J.: Learning and optimization using the clonal selection principle. *IEEE Transactions on Evolutionary Computation* **6**(3) (2002) 239–251
11. Gonzalez, F., Dasgupta, D.: Neuro-Immune and SOM-Based Approaches: A Comparison. *Proceedings of 1st International Conference on Artificial Immune Systems (ICARIS-2002)*, University of Kent at Canterbury, UK, September 9th–11th, 2002

The Effect of Binary Matching Rules in Negative Selection

Fabio González¹, Dipankar Dasgupta², and Jonatan Gómez¹

¹ Division of Computer Science, The University of Memphis, Memphis TN 38152
and Universidad Nacional de Colombia, Bogotá, Colombia

{fgonzalz, jgomez}@memphis.edu

² Division of Computer Science, The University of Memphis, Memphis TN 38152
dasgupta@memphis.edu

Abstract. Negative selection algorithm is one of the most widely used techniques in the field of artificial immune systems. It is primarily used to detect changes in data/behavior patterns by generating detectors in the complementary space (from given normal samples). The negative selection algorithm generally uses binary matching rules to generate detectors. The purpose of the paper is to show that the low-level representation of binary matching rules is unable to capture the structure of some problem spaces. The paper compares some of the binary matching rules reported in the literature and study how they behave in a simple two-dimensional real-valued space. In particular, we study the detection accuracy and the areas covered by sets of detectors generated using the negative selection algorithm.

1 Introduction

Artificial immune systems (**AIS**) is a relatively new field that tries to exploit the mechanisms present in the biological immune system (**BIS**) in order to solve computational problems. There exist many AIS works [5,8], but they can roughly be classified into two major categories: techniques inspired by the self/non-self recognition mechanism [12] and those inspired by the immune network theory [9,22].

The *negative selection* (**NS**) algorithm was proposed by Forrest and her group [12]. This algorithm is inspired by the mechanism of T-cell maturation and self tolerance in the immune system. Different variations of the algorithm have been used to solve problems of anomaly detection [4,16], fault detection [6], to detect novelties in time series [7], and even for function optimization [3].

A process that is of primary importance for the BIS is the antibody-antigen matching process, since it is the basis for the recognition and selective elimination mechanism that allows to identify foreign elements. Most of the AIS models implement this recognition process, but in different ways. Basically, antigens and antibodies are represented as strings of data that correspond to the sequence of aminoacids that constituting proteins in the BIS. The matching of two strings is determined by a function that produces a binary output (match or not-match).

The binary representation is general enough to subsume other representations; after all, any data element, whatever its type is, is represented as a sequence of bits in the memory of a computer (though, how they are treated may differ). In theory, any matching

rule defined on a high-level representation can be expressed as a binary matching rule. However, in this work, we restrict the use of the term *binary matching rule* to designate those rules that take into account the matching of individual bits representing the antibody and the antigen.

Most works on the NS algorithm have been restricted to binary matching rules like *r*-contiguous [1,10,12]. The reason is that efficient algorithms that generate detectors (antibodies or T-cell receptors) have been developed, exploiting the simplicity of the binary representation and its matching rules [10]. On the other hand, AIS approaches inspired by the immune network theory often use real vector representation for antibodies and antigens [9,22], as this representation is more suitable for applications in learning and data analysis. The matching rules used with this real-valued representation are usually based on Euclidean distance, (i.e. the smaller the antibody-antigen distance, the more affinity they have).

The NS algorithm has been applied successfully to solve different problems; however, some unsatisfactory results have also been reported [20]. As it was suggested by Balthrop et al. [2], the source of the problem is not necessarily the NS algorithm itself, but the kind of matching rule used. The same work [2] proposed a new binary matching rule, *r*-chunk matching (Equation 2 in Section 2.1), which appears to perform better than *r*-contiguous matching.

The starting point of this paper is to address the question: do the low-level representation and its matching rules affect the performance of NS in covering the non-self space? This paper provides some answers to this issue. Specifically, it shows that the low-level representation of the binary matching scheme is unable to capture the structure of even simple problem spaces. In order to justify our argument, we use some of the binary matching rules reported in the literature and study how they behave in a simple bi-dimensional real space. In particular, we study the shape of the areas covered by individual detectors and by a set of detectors generated by the NS algorithm.

2 The Negative Selection Algorithm

Forrest et al. [12] developed the NS algorithm based on the principles of self/non-self discrimination in the BIS. The algorithm can be summarized as follows (taken from [5]):

- Define self as a collection S of elements in a representation space U (also called self/non-self space), a collection that needs to be monitored.
- Generate a set R of *detectors*, each of which fails to match any string in S .
- Monitor S for changes by continually matching the detectors in R against S .

2.1 Binary Matching Rules in Negative Selection Algorithm

The previous description is very general and does not say anything about what kind of representation space is used or what the exact meaning of *matching* is. It is clear that the algorithmic problem of generating good detectors varies with the type of representation space (continuous, discrete, hybrid, etc.), the detector representation, and the process that determines the matching ability of a detector.

A binary matching rule is defined in terms of individual bit matchings of detectors and antigens represented as binary strings. In this section, some of the most widely used binary matching rules are presented.

***r*-contiguous matching.** The first version of the NS algorithm [12] used binary strings of fixed length, and the matching between detectors and new patterns is determined by a rule called *r*-contiguous matching. The binary matching process is defined as follows: given $x = x_1x_2...x_n$ and a detector $d = d_1d_2...d_n$,

$$d \text{ matches } x \equiv \exists i \leq n - r + 1 \text{ such that } x_j = d_j \text{ for } j = i, \dots, i + r - 1, \quad (1)$$

that is, the two strings match if there is a sequence of size r where all the bits are identical. The algorithm works in a generate-and-test fashion, i.e. random detectors are generated; then, they are tested for self-matching. If a detector fails to match a self string, it is retained for novel pattern detection.

Subsequently, two new algorithms based on dynamic programming were proposed [10], the *linear* and the *greedy* NS algorithm. Similar to the previous algorithm, they are also specific to binary string representation and *r*-contiguous matching. Both algorithms run in linear time and space with respect to the size of the self set, though the time and space are exponential on the size of the matching threshold, r .

***r*-chunk matching.** Another binary matching scheme called *r*-chunk matching was proposed by Balthrop et al. [1]. This matching rule subsumes *r*-contiguous matching, that is, any *r*-contiguous detector can be represented as a set of *r*-chunk detectors. The *r*-chunk matching rule is defined as follows: given a string $x = x_1x_2...x_n$ and a detector $d = (i, d_1d_2...d_m)$, with $m \leq n$ and $i \leq n - m + 1$,

$$d \text{ matches } x \equiv x_j = d_j \text{ for } j = i, \dots, i + m - 1, \quad (2)$$

where i represents the position where the *r*-chunk starts. Preliminary experiments [1] suggest that the *r*-chunk matching rule can improve the accuracy and performance of the NS algorithm.

Hamming distance matching rules. One of the first works that modeled BIS concepts in developing pattern recognition was proposed by Farmer et al. [11]. Their work proposed a computational model of the BIS based on the idiotypic network theory of Jerne [19], and compared it with the *learning classifier system* [18]. This is a binary model representing antibodies and antigens and defining a matching rule based on the Hamming distance. A Hamming distance based matching rule can be defined as follows: given a binary string $x = x_1x_2...x_n$ and a detector $d = d_1d_2...d_n$,

$$d \text{ matches } x \equiv \sum_i x_i \oplus d_i \geq r, \quad (3)$$

where $\oplus$ is the exclusive-or operator, and $0 \leq r \leq n$ is a threshold value.

Different variations of the Hamming matching rule were studied, along with other rules like r -contiguous matching, statistical matching and landscape-affinity matching [15]. The different matching rules were compared by calculating the signal-to-noise ratio and the function-value distribution of each matching function when applied to a randomly generated data set. The conclusion of the study was that the Rogers and Tanimoto (**R&T**) matching rule, a variation of the Hamming distance, produced the best performance. The R&T matching rule is defined as follows: given a binary string $x = x_1x_2\dots x_n$ and a detector $d = d_1d_2\dots d_n$,

$$d \text{ matches } x \equiv \frac{\sum_i \overline{x_i \oplus d_i}}{\sum_i x_i \oplus d_i + 2 \sum_i \overline{x_i \oplus d_i}} \geq r, \quad (4)$$

where $\oplus$ is the exclusive-or operator, and $0 \leq r \leq 1$ is a threshold value.

It is important to mention that a good detector generation scheme for this kind of rules is not available yet, other than the exhaustive generate-and-test strategy [12].

3 Analyzing the Shape of Binary Matching Rules

Usually, the self/non-self space (U) used by the NS algorithm corresponds to an abstraction of a specific problem space. Each element in the problem space (e.g. a feature vector) is mapped to a corresponding element in U (e.g. a bit string).

A matching rule defines a relation between the set of detectors¹ and U . If this relationship is mapped back to the problem space, it can be interpreted as a relation of affinity between elements in this space. In general, it is expected that elements that are matched by the same detector have some common property. So, a way to analyze the ability of a matching rule to capture this ‘affinity’ relationship in the problem space is to take the subset of U corresponding to the elements matched by a specific detector, and map this subset back to the problem space. Accordingly, this set of elements in the problem space is expected to share some common properties.

In this section, we apply the approach described above to study the binary matching rules presented in section 2.1. The problem space used corresponds to the set $[0.0, 1.0]^2$. One reason for choosing this problem space is that multiple problems in learning, pattern recognition, and anomaly detection can be easily expressed in an n -dimensional real-valued space. Also, it makes easier to visualize the *shape* of different matching rules.

All the examples and experiments in this paper use a self/non-self space composed of binary strings of length 16. An element (x, y) in the problem space is mapped to the string $b_0, \dots, b_7, b_8, \dots, b_{15}$, where the first 8 bits encode the integer value $\lfloor 255 \cdot x + 0.5 \rfloor$ and the last 8 bits encode the integer value $\lfloor 255 \cdot y + 0.5 \rfloor$. Two encoding schemes are studied: conventional binary representation and Gray encoding. Gray encoding is expected to favor binary matching rules, since the codifications of two consecutive numbers only differs by one bit.

¹ In some matching rules, the set of detectors is same as U (e.g. r -contiguous matching). In other cases, it is a different set that usually contains or extends U (e.g. r -chunk matching).

Figure 1 shows some typical shapes generated by different binary matching rules. Each figure represents the area (in the problem space) covered by one detector located at the center, (0.5,0.5) (1000000010000000 in binary notation). In the case of r -chunk matching, the detector does not correspond to an entire string representing a point on the problem space, rather, it represents a substring (chunk). Thus, we chose an r -chunk detector that matches the binary string corresponding to (0.5,0.5), ****00001000****. The area covered by a detector is drawn using the following process: the detector is matched against all the binary strings in the self/non-self space; then, all the strings that match are mapped back to the problem space; finally, the corresponding points are painted in gray color.

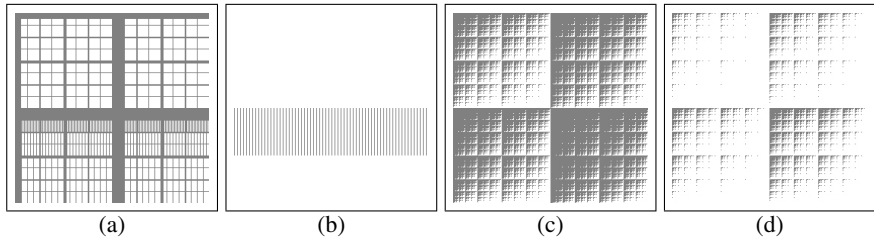


Fig. 1. Areas covered in the problem space by an individual detector using different matching rules. The detector corresponds to 1000000010000000, which is the binary representation of the point (0.5,0.5). (a) r -contiguous matching, $r = 4$, (b) r -chunk matching, $d = ****00001000****$, (c) Hamming matching, $r = 8$, (d) R&T matching, $r = 0.5$.

The shapes generated by the r -contiguous rule (Figure 1(a)) are composed by vertical and horizontal stripes that constitute a grid-like shape. The horizontal and vertical stripes correspond to sets of points having identical bits at least at r contiguous positions in the encoded space. Some of these points, however, are not close to the detector in the decoded (problem) space. The r -chunk rule generates similar, but simpler shapes (Figure 1(b)). In this case, the area covered is composed of vertical or horizontal sets of parallel strips. The orientation depends on the position of the r -chunk: if it is totally contained in the first eight bits, the strips are vertically going from top to bottom; if it is contained on the last eight bits, the strips are oriented horizontally; finally, if it covers both parts, it has the shape shown in Figure 1(b).

The area covered by Hamming and R&T matching rules has a fractal-like shape, shown in Figure 1(c) and 1(d), i.e. it exhibits self-similarity. It is composed of points that have few interconnections. There is no significant difference between the shapes generated by the R&T rule and those generated by the Hamming rule, which is not a surprise, considering the fact that the R&T rule is based on Hamming distance.

The shape of the areas covered by r -contiguous and r -chunk matching is not affected by the change in codification from binary to Gray (as shown in Figures 2(a) and 2(b)). This is not the case with the Hamming and the R&T matching rule (Figures 2(c) and

2(d)). The reason is that the Gray encoding represents consecutive values using bit strings with small Hamming distance.

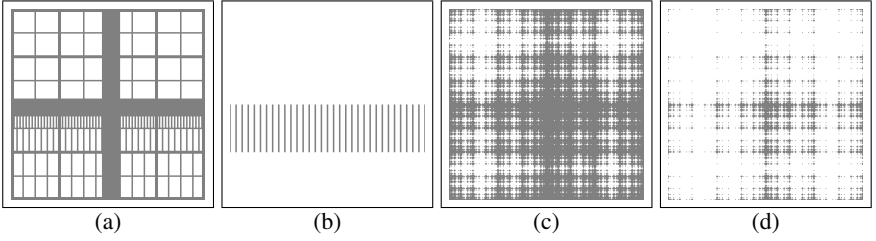


Fig. 2. Areas covered in the problem space by an individual detector using Gray encoding for the self/non-self space. The detector corresponds to 1100000011000000, which is the Gray representation of the point (0.5,0.5). (a) r -contiguous matching, $r = 4$, (b) r -chunk matching, $d = 0.5$, (c) Hamming matching, $r = 8$, (d) R&T matching, $r = 0.5$.

The different matching rules and representations generate different types of detector covering shapes. This reflects the bias introduced by each representation and the matching scheme. It is clear that the relation of proximity exhibited by these matching rules in the binary self/non-self space does not coincide with the natural relation of proximity in a real-valued, two-dimensional space. Intuitively, this seems to make the task harder of placing these detectors to cover the non-self space without covering the self set. This fact is further investigated in the next section.

4 Comparing the Performance of Binary Matching Rules

This section shows the performance of the binary matching rules (as presented in section 2.1) in the NS algorithm. A generate-and-test NS algorithm is used. Experiments are performed using two synthetic data sets shown in Figure 3.

The first data set (Figure 3(a)) was created by generating random vectors (1000) in $[0, 1]^2$ with the center in (0.5,0.5) and scaling them to a norm less than 0.1, so that the points lie within a single circular cluster. The second set (Fig. 3(b)) was extracted from the Mackey-Glass time series data set, which has been used in different works that apply AIS to anomaly detection problems [7,14,13]. The original data set has four features extracted by a sliding window. We used only the first and the fourth feature. The data set is divided in two sets (training and testing), each one with 497 samples. The training set has only normal data, and the testing set has mixed normal and abnormal data.

4.1 Experiments with the First Data Set

Figure 4 shows a typical coverage of the non-self space corresponding to a set of detectors generated by the NS algorithm with r -contiguous matching for the first data set. The non-covered areas in the non-self space are known as *holes* [17] and are due to the

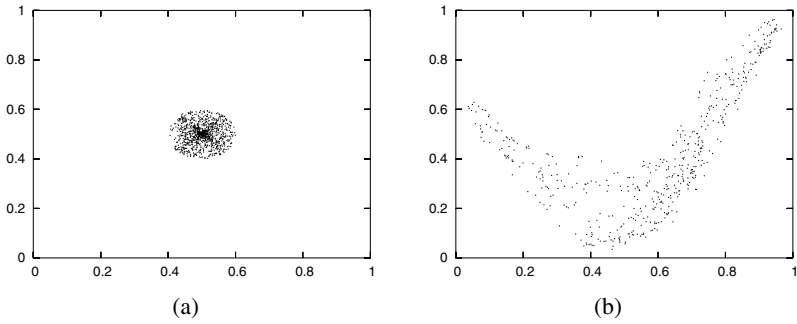


Fig. 3. Self data sets used as input to the NS algorithm shown in a two-dimensional real-valued problem space. (a) First data set composed of random points inside of a circle of radius 0.1 (b) Second data set corresponding to a section of the Mackey-Glass data set [7,14,13].

characteristics of r -contiguous matching. In some cases, these holes can be good: since they are expected to be close to self strings, the set of detectors will not detect small deviations from the self set, making the NS algorithm robust to noise. However, when we map the holes from the representation (self/non-self) space to the problem space, they are not necessarily close to the self set, as shown in Figure 4. This result is not surprising; as we saw in the previous section (section 3), the binary matching rules fail to capture the concept of *proximity* in this two-dimensional space.

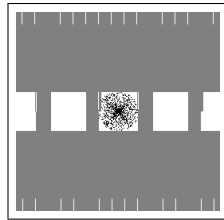


Fig. 4. Coverage of space by a set of detectors generated by NS algorithm using r -contiguous matching (with $r = 7$). Black dots represent self-set points, and gray regions represent areas covered by the generated detectors (4446).

We run the NS algorithm using different matching rules and varying the r value. Figure 5 shows the best coverage generated using standard (no Gray) binary representation. The improvement in the coverage generated by r -contiguous matching (Figure 5(a)) is due to the higher value of r ($r = 9$), which produces more specific detectors. The coverage with the r -chunk matching rule (Figure 5(b)) is more consistent with the shape of the self set because of the high specificity of r -chunk detectors. The outputs produced by the NS algorithm with Hamming and R&T matching rules are the same.

These two rules do not seem to do as well as the other matching rules (Figure 5(c)). However, by changing the encoding from binary to Gray (Figure 5(d)), the performance can be improved, since the Gray encoding changes the detector shape, as was shown in the previous section (Section 3). The change in the encoding scheme, however, does not affect the performance of the other rules for this particular data set.

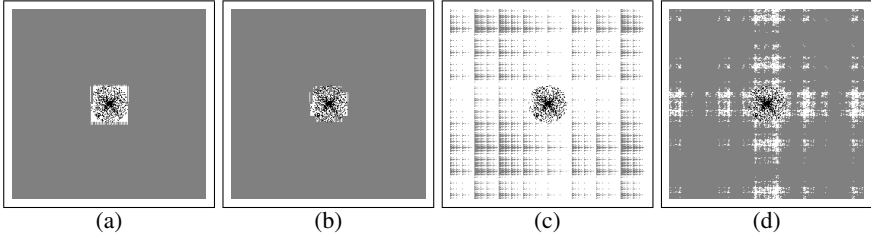


Fig. 5. Best space coverage by detectors generated with NS algorithm using different matching rules. Black dots represent self-set points, and gray regions represent areas covered by detectors. (a) r -contiguous matching, $r = 9$, binary encoding, 36,968 detectors. (b) r -chunk matching, $r = 10$, binary encoding, 6,069 detectors. (c) Hamming matching, $r = 12$, binary encoding (same as R&T matching, $r = 10/16$), 9 detectors. (d) Hamming matching, $r = 10$, Gray encoding (same as R&T matching, $r = 7/16$), 52 detectors.

The r -chunk matching rule produced the best performance in this data set, followed closely by the r -contiguous rule. This is due to the shape of the areas covered by r -chunk detectors which adapt very well to the simple structure of this self set, one localized, circular cluster of data points.

4.2 Experiments with the Second Data Set

The second data set has a more complex structure than the first one, where the data are spread in a certain pattern. The NS algorithm should be able to generalize the self set with incomplete data. The NS algorithm was run with different binary matching rules, with both encodings (binary and Gray), and varying the value parameter r (the different values are shown in Table 1). Figure 6 shows some of the best results produced. Clearly, the tested matching rules were not able to produce a good coverage of the non-self space. The r -chunk matching rule generated satisfactory coverage of the non-self space (Figure 6(b)); however, the self space was covered by some lines resulting in erroneously detecting the self as non-self (false alarms). The Hamming-based matching rules generated an even more stringent result (Figure 6(d)) that covers almost the entire self space. The parameter r , which works as a threshold, controls the detection sensitivity. A smaller value of r generates more general detectors (i.e. covering a larger area) and decreases the detection sensitivity. However, for a more complex self set, changing the value of r from 8 (Figure 6(b)) to 7 (Figure 6(c)) generates a coverage with many holes in the non-self area, and still with some portions of the self covered by detectors. So, this

problem is not with the setting of the correct value for r , but a fundamental limitation on of the binary representation that is not capable of capturing the semantics of the problem space. The performance of the Hamming-based matching rules is even worse; it produces a coverage that overlaps most of the self space (Figure 6(d)).

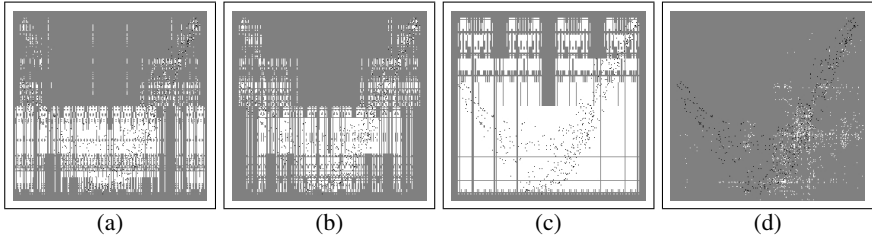


Fig. 6. Best coverage of the non-self space by detectors generated with negative selection. Different matching rules, parameter values and codings (binary and Gray) were tested. The number of detectors is reported in Table 1. (a) r -contiguous matching, $r = 9$, Gray encoding. (b) r -chunk matching, $r = 8$, Gray encoding. (c) r -chunk matching, $r = 7$, Gray encoding. (d) Hamming matching, $r = 13$, binary encoding (same as R&T matching, $r = 10/16$).

A better measure to determine the quality of the non-self space coverage with a set of detectors can be produced by matching the detectors against a test data set. The test data set is composed of both normal and abnormal elements as described in [13]. The results are measured in terms of the *detection rate* (percentage of abnormal elements correctly identified as abnormal) and the *false alarm rate* (percentage of the normal detectors wrongly identified as abnormal). An ideal set of detectors would have a detection rate close to 100%, while keeping a low false alarm rate. Table 1 accounts the results of experiments that combine different binary matching rules, different threshold or window size values (r), and two types of encoding. In general, the results are very poor. None of the configurations managed to deliver a good detection rate with a low false alarm rate. The best performance, which is far from good, is produced by the coverage depicted in Figure 6(b) (r -chunk matching, $r = 8$, Gray encoding), with a detection rate of 73.26% and a false alarm rate of 47.47%. These results are in contrast with other previously reported [7,21]; however, it is important to notice that in those experiments, the normal data in the test set is same to the normal data in the training set; so, no new normal data was presented during testing. In our case, the normal samples in the test data are, in general, different from those in the training set, though they are generated by the same process. Hence, the NS algorithm has to be able to generalize the structure of the self set in order to be able to classify correctly previously unseen normal patterns. But, is this a problem with the matching rule or a more general issue in the NS algorithm? In fact, the NS algorithm can perform very well on the same data set if the right matching rule is employed. We used a real value representation matching rule and followed the approach proposed in [14] on the second data set. The performance over the test data set

was detection rate, 94%, false alarm, 3.5%. These results are clearly superior to all the results reported in Table 1.

Table 1. Results of different matching rules in NS using the the second test data set. (r: threshold parameter, ND: number of detectors, D%: detection rate, FA%: false alarm rate). The results in bold correspond to the sets of detectors shown in Figure 6.

	r	Binary			Gray		
		ND	D%	FA %	ND	D%	FA %
<i>r</i> -contiguous	7	0			40	3.96%	1.26%
	8	343	15.84%	16.84%	361	16.83%	16.67%
	9	4531	53.46%	48.48%	4510	66.33%	48.23%
	10	16287	90.09%	77.52%	16430	90.09%	75.0%
	11	32598	95.04%	89.64%	32609	98.01%	90.4%
<i>r</i> -chunk	4	0			2	0.0%	0.75%
	5	4	0.0%	0.75%	8	0.0%	0.75%
	6	18	3.96%	4.04%	22	3.96%	2.52%
	7	98	14.85%	16.16%	118	18.81%	13.13%
	8	549	54.45%	48.98%	594	73.26%	47.47%
	9	1942	85.14%	72.97%	1959	88.11%	67.42%
	10	4807	98.01%	86.86%	4807	98.01%	86.86%
	11	9948	100%	92.92%	9948	100%	92.92%
	12	18348	100%	94.44%	18348	100%	94.44%
Hamming	12	1	0.99%	3.03%	7	10.89%	8.08%
	13	2173	99%	91.16%	3650	99.0%	91.66%
	14	29068	100%	95.2%	31166	100%	95.2%
Rogers & Tanimoto	9/16	1	0.99%	3.03%	7	10.89%	8.08%
	10/16	2173	99%	91.16%	3650	99%	91.66%
	11/16	29068	100%	95.2%	31166	100%	95.2%
	12/16	29068	100%	95.2%	31166	100%	95.2%

5 Conclusions

In this paper, we discussed different binary matching rules used in the negative selection (NS) algorithm. The primary applications of NS have been in the field of change (or anomaly) detection, where the detectors are generated in the complement space which can detect changes in data patterns. The main component of NS is the choice of a matching rule, which determines the similarity between two patterns in order to classify self/non-self (normal/abnormal) samples. There exists a number of matching rules and encoding schemes for the NS algorithm. This paper examines the properties (in terms of coverage and detection rate) of each binary matching rule for different encoding schemes.

Experimental results showed that the studied binary matching rules cannot produce a good generalization of the self space, which results in a poor coverage of the non-

self space. The reason is that the affinity relation implemented by the matching rule at the representation level (self/non-self) space cannot capture the affinity relationship at the problem space. This phenomenon is observed in our experiments with a simple real-valued two-dimensional problem space.

The main conclusion of this paper is that the matching rule for NS algorithm needs to be chosen in such a way that it accurately represents the data proximity in the problem space. Another factor to take into account is the type of application. For instance, in change detection applications (integrity of software or data files), where the complete knowledge of the self space is available, the generalization of the data may not be necessary. In contrast, in anomaly detection applications, like those in computer security where a normal behavior model needs to be build using available samples in a training set, it is crucial to count on matching rules that can capture the semantics of the problem space [4,20].

Other types of representation and detection schemes for the NS algorithm have been proposed by different researchers [4,13,15,21,23]; however, they have not been studied as extensively as binary schemes. The findings in this paper provide motivation to further explore matching rules for different representations. Particularly, our effort is directed to investigate methods to generate good sets of detectors in real valued spaces. This type of representation also opens the possibility to integrate NS with other AIS techniques like those inspired by the immune memory mechanism [9,22].

Acknowledgments. This work was funded by the Defense Advanced Research Projects Agency (no. F30602-00-2-0514) and National Science Foundation (grant no. IIS-0104251). The authors would like to thank Leandro N. de Castro and the anonymous reviewers for their valuable corrections and suggestions to improve the quality of the paper.

References

1. J. Balthrop, F. Esponda, S. Forrest, and M. Glickman. Coverage and generalization in an artificial immune system. In *GECCO 2002: Proceedings of the Genetic and Evolutionary Computation Conference*, pages 3–10, New York, 9-13 July 2002. Morgan Kaufmann Publishers.
2. J. Balthrop, S. Forrest, and M. R. Glickman. Revisiting lisy: Parameters and normal behavior. In *Proceedings of the 2002 Congress on Evolutionary Computation CEC2002*, pages 1045–1050. IEEE Press, 2002.
3. C. A. C. Coello and N. C. Cortes. A parallel implementation of the artificial immune system to handle constraints in genetic algorithms: preliminary results. In *Proceedings of the 2002 Congress on Evolutionary Computation CEC2002*, pages 819–824, Honolulu, Hawaii, 2002.
4. D. Dagupta and F. González. An immunity-based technique to characterize intrusions in computer networks. *IEEE Transactions on Evolutionary Computation*, 6(3):281–291, June 2002.
5. D. Dasgupta. An overview of artificial immune systems and their applications. In D. Dasgupta, editor, *Artificial immune systems and their applications*, pages pp 3–23. Springer-Verlag, Inc., 1999.

6. D. Dasgupta and S. Forrest. Tool breakage detection in milling operations using a negative-selection algorithm. Technical Report CS95-5, Department of Computer Science, University of New Mexico, 1995.
7. D. Dasgupta and S. Forrest. Novelty detection in time series data using ideas from immunology. In *Proceedings of the International Conference on Intelligent Systems*, pages 82–87, June 1996.
8. L. N. de Castro and J. Timmis. *Artificial Immune Systems: A New Computational Approach*. Springer-Verlag, London, UK, 2002.
9. L. N. de Castro and F. J. Von Zuben. An evolutionary immune network for data clustering. *Brazilian Symposium on Artificial Neural Networks (IEEE SBRN'00)*, pages 84–89, 2000.
10. P. D'haeseleer, S. Forrest, and P. Helman. An immunological approach to change detection: algorithms, analysis and implications. In *Proceedings of the 1996 IEEE Symposium on Computer Security and Privacy*, pages 110–119, Oakland, CA, 1996.
11. J. D. Farmer, N. H. Packard, and A. S. Perelson. The immune system, adaptation, and machine learning. *Physica D*, 22:187–204, 1986.
12. S. Forrest, A. Perelson, L. Allen, and R. Cherukuri. Self-nonsself discrimination in a computer. In *Proc. IEEE Symp. on Research in Security and Privacy*, pages 202–212, 1994.
13. F. González and D. Dagupta. Neuro-immune and self-organizing map approaches to anomaly detection: A comparison. In *Proceedings of the 1st International Conference on Artificial Immune Systems*, pages 203–211, Canterbury, UK, Sept. 2002.
14. F. González, D. Dasgupta, and R. Kozma. Combining negative selection and classification techniques for anomaly detection. In *Proceedings of the 2002 Congress on Evolutionary Computation CEC2002*, pages 705–710, Honolulu, HI, May 2002. IEEE.
15. P. Harmer, G. Williams, P.D. and Gnusch, and G. Lamont. An Artificial Immune System Architecture for Computer Security Applications. *IEEE Transactions on Evolutionary Computation*, 6(3):252–280, June 2002.
16. S. Hofmeyr and S. Forrest. Architecture for an artificial immune system. *Evolutionary Computation*, 8(4):443–473, 2000.
17. S. A. Hofmeyr. An interpretative introduction to the immune system. In I. Cohen and L. Segel, editors, *Design principles for the immune system and other distributed autonomous systems*. Oxford University Press, 2000.
18. J. H. Holland, K. J. Holyoak, R. E. Nisbett, and P. R. Thagard. *Induction: Processes of Inference, Learning, and Discovery*. MIT Press, Cambridge, 1986.
19. N. K. Jerne. Towards a network theory of the immune system. *Ann. Immunol. (Inst. Pasteur)*, 125C:373–389, 1974.
20. J. Kim and P. Bentley. An evaluation of negative selection in an artificial immune system for network intrusion detection. In *GECCO 2001: Proceedings of the Genetic and Evolutionary Computation Conference*, pages 1330–1337, San Francisco, California, USA, 2001. Morgan Kaufmann.
21. S. Singh. Anomaly detection using negative selection based on the r-contiguous matching rule. In *Proceedings of the 1st International Conference on Artificial Immune Systems (ICARIS)*, pages 99–106, Canterbury, UK, sep 2002.
22. J. Timmis and M. J. Neal. A resource limited artificial immune system for data analysis. In *Research and development in intelligent systems XVII, proceedings of ES2000*, pages 19–32, Cambridge, UK, 2000.
23. P. D. Williams, K. P. Anchor, J. L. Bebo, G. H. Gunsch, and G. D. Lamont. CDIS: Towards a computer immune system for detecting network intrusions. *Lecture Notes in Computer Science*, 2212:117–133, 2001.

Immune Inspired Somatic Contiguous Hypermutation for Function Optimisation

Johnny Kelsey and Jon Timmis

Computing Laboratory, University of Kent
Canterbury, Kent. CT2 7NF. UK
{jk34,jt6}@kent.ac.uk

Abstract. When considering function optimisation, there is a trade off between quality of solutions and the number of evaluations it takes to find that solution. Hybrid genetic algorithms have been widely used for function optimisation and have been shown to perform extremely well on these tasks. This paper presents a novel algorithm inspired by the mammalian immune system, combined with a unique mutation mechanism. Results are presented for the optimisation of twelve functions, ranging in dimensionality from one to twenty. Results show that the immune inspired algorithm performs significantly fewer evaluations when compared to a hybrid genetic algorithm, whilst not sacrificing quality of the solution obtained.

1 Introduction

The problem of function optimisation has been of interest to computer scientists for decades. Function optimisation can be characterised as, given an arbitrary function, how can the maximum (or minimum) value of the function be found. Such problems can present a very large search space, particularly when dealing with higher-dimensional functions. Genetic algorithms (GAs) though not initially designed for such a purpose, however, they soon began to grow in favour with researchers for this task. Whilst the standard GA performs well in terms of finding solutions, it is typical that for more complex problems, some form of hybridisation of the GA is performed: typically, an extra search mechanism is employed as part of the hybridisation, for example hill climbing, to help the GA perform a more effective local search near the optimum [10].

In recent years, interest has been growing in the use of other biologically inspired models: in particular the immune system, as witnessed by the emergence of the field of Artificial Immune Systems (AIS). AIS can be defined as adaptive systems inspired by theoretical immunology and observed immune functions and principles, which are applied to problem solving [5]. This insight into the immune system has led to an ever increasing body of research in a wide variety of domains. To review the whole area would be outside the scope of this paper, but work pertinent to this paper is work on function optimisation [4], extended with an immune network approach in [6] and applied to multi-modal optimisation.

Other germane and significant papers include [19], where work there is considered on multi-objective optimisation. However, work proposed in this paper varies significantly in terms of population evolution and mutation mechanisms employed.

This paper presents initial work into the investigation of immune inspired algorithms for function optimisation. A novel mutation mechanism has been developed, loosely inspired by the mutation mechanism found in B-cell receptors in the immune system. This coupled with evolutionary pressure observed in the immune system, leads to the development of a novel algorithm for function optimisation. Experiments with twelve different functions have shown the algorithm to perform significantly fewer evaluations when compared to a standard hybrid GA, whilst maintaining high accuracy on the solutions found.

This paper first outlines a hybrid genetic algorithm which might typically be used for function optimisation. Then there follows a short discussion on immune inspired algorithms which outlines the basis of the theoretical framework underpinning AIS. The focus of the paper then turns to the novel B-cell algorithm, followed with the presentation and initial analysis of the first empirical results obtained. Conclusions are drawn and future research directions are explored.

2 Hybrid Genetic Algorithms

Hybrid genetic algorithms (HGAs) have, over the last decade, become almost standard tools for function optimisation and combinatorial analysis: according to Goldberg et. al., real-world business and engineering applications are typically undertaken with some form of hybridisation between the GA and a specialised search [10]. The reason for this is that HGAs generally have an improved performance, as has been demonstrated in such diverse areas as vehicle routing [2] and multiple protein sequence alignment [16]).

As an example, within a HGA a population P is given as candidates to optimised an objective function $g(x)$. Each member of the population can be thought of as a vector $\underline{v}$ of bit strings of length $l = 64$ (to represent double-precision floating point numbers, although this does not have to be the case) where $\underline{v} \in P$ and P is the population. Hybrid genetic algorithms employ an extra operator working in conjunction with crossover and mutation which improves the fitness of the population. This can come in many different guises: sometimes it is specific to the particular problem domain; when dealing with numerical function optimisation, the HGA is likely to employ a variant of local search. The basic procedure of a HGA is given in figure 1. The local search mechanism functions by examining the neighbourhood of the fitness individuals within a given landscape of the population. This allows for a more specific search around possible solutions that results in a faster convergence rate to a possible solution. The local search typically operates as described in figure 2. Notice that there are two distinct mutation rates utilised: the standard genetic algorithm typically uses a very low level of mutation, and the local search function $h(x)$ uses a much higher one, so

we have $\delta \ll \rho$. This process outlined in figure 2. This method of hybridising a GA is adopted as the model for the HGA used in this paper.

1. *Initialisation*: create an initial random population (P) of individuals $\underline{v}$;
 - a) *Fitness evaluation*: $\forall \underline{v} \in P$: evaluate fitness of $P(\underline{v})$ with objective function $g(x)$;
 - b) *Diversity*
 - i. *Selection and crossover*: Select n number of fittest individuals and with probability p perform crossover between selected individuals;
 - ii. *Mutation*: subject t number of individuals of the population to a low level of mutation with an equally low probability;
 - c) *Utilise hybrid function*: subject s members of the population to a hybrid search technique $h(x)$; if a higher-fitness member results, return this to the population;
 - d) *Cycle*: repeat from step (a) until a certain stopping criterion is met.

Fig. 1. Generic Hybrid GA algorithm

1. *Select*: copy $\underline{v}$ to $\underline{v}'$;
2. *Explore neighbourhood*: apply mutation to $\underline{v}'$ with probability ρ ;
 - a) *Generate number of mutations*: Subject $\underline{v}'$ to mutation: $N_{mut} = f(\rho)$;
 - b) *Generate mutation sites*: for N_{mut} , randomly select sites on $\underline{v}'$ and perturb bit string;
3. *Fitness Evaluation*: if $g(\underline{v}') > g(\underline{v})$, replace $\underline{v}$ so that $\underline{v}' \in P$;

Fig. 2. Example of local search mechanism for a HGA

3 Artificial Immune Systems

There has been a growing interest in the use of the biological immune system as a source of inspiration to the development of computational systems [5]. The natural immune system protects our bodies from infection and this is achieved by a complex interaction of white blood cells called B Cells and T Cells. Essentially, AIS is concerned with the use of immune system components and processes as inspiration to construct computational systems. This insight into the natural immune system has led to an increasing body of work in a wide variety of domains. Much of this work emerged from early work in theoretical immunology [13], [8] and where mathematical models of immune system process were developed in an attempt to better understand the function of the immune system. This acted as a mini-catalyst for computer scientists, examples being work on on computer

security [9] and virus detection [14]. Researchers realised that, although the computer security metaphor was a natural first choice for AIS, there are many other potential application areas that could be explored such as machine learning [18], scheduling [12] and optimisation [4].

Recent work in [5] has proposed a framework for the construction of AIS. This framework can be described in three layers. The first layer is one of representation of the system, this is termed shape space and defines the components for the system. A typical shape space for a system may be binary, where elements within each component can take either a zero or one value. The second layer is one of affinity measures: this allows for the measurement of the *goodness* of the component when measured against the problem. In terms of optimisation, this would be in terms of how well the values in the component performed with respect to the function being optimised. Finally, immune algorithms control the interactions of these components in terms of population evolution and dynamics. Such basic algorithms include negative selection, clonal selection and immune network models. These can be utilised as *building blocks* for AIS and augmented and adapted as desired. At present, clonal selection based algorithms have been typically used to build AIS for optimisation. This is the approach adopted in this paper. Work in this paper can be considered as an augmentation to the framework in the area of immune algorithms, rather than offering anything new in terms of representation and affinity measures.

3.1 An Immune Algorithm for Optimisation

Pertinent to work in this paper is work in [4]. Here the authors proposed an algorithm inspired by the workings of the immune system, in a process known as clonal selection. There are other examples of immune inspired optimisation such as [11], however these will not be discussed here. The reader is directed to [5] for a full review of these techniques. Clonal selection is the process by which the immune system is said to respond to invading organisms (pathogens, which then become antigens). The process is conceptually simple: the immune system is made up of cells known as T-cells and B-cells (all of which have receptors on them which are capable of recognising antigens, via a binding mechanism analogous to a lock and key). When an antigen enters the host, receptors on B-cells and T-cells attach themselves to the antigens. These cells become stimulated through this interaction, with B-cells receiving stimulation from T-cells that attach themselves to similar antigen. Once a certain level of stimulation is reached, B-cells begin to clone at a rate proportional to their affinity to the antigen. These clones undergo a process of affinity maturation: this is achieved by the mutation of the clones at a high rate (known as somatic hypermutation) and selection of the *strongest* cells, some of which are retained as memory cells. At the end of each iteration, a certain number of random individuals are inserted into the population, to maintain an element of diversity.

Results reported for CLONALG (CLONal ALGORITHM), which captures the above process, seem to indicate that it performs well on function optimisation [4]. However, from the paper it was hard to extract an exact number of evaluations

and solutions found, as these were not presented other than in graphical form. Additionally, a detailed comparison between alternative techniques was never undertaken, so it has proved difficult to fully assess the potential of the algorithm.

The work presented in this paper (undertaken independently of and contemporaneously to the above work) is a variation of clonal selection, which applies a novel mutation operator and a different selection mechanism, which has been found to greatly improve on optimisation performance on a number of functions.

4 The B-Cell Algorithm

This paper proposes a novel algorithm, called the B-cell algorithm (BCA), which is also inspired by the clonal selection process. An important feature of the BCA is its use of a unique mutation operator, known as *contiguous somatic hypermutation*. Evidence for this in the immunological literature is sparse, but such examples are [17], [15]. Here the authors argue that mutation occurs in *clusters* of regions within cells: this is analogous to contiguous regions. However, in the spirit of biologically inspired computing, it is not necessary for the underlying biological theory to be proven, as computer scientists are interested in taking inspiration from these theories to help improve on current solutions. As will be shown the BCA is different to both CLONALG and HGAs in a number of ways. The BCA and motivation for the algorithm will now be discussed.

The representation employed in the BCA is one of a N-dimensional vector of 64-bit strings (as in the HGA above), known as Binary Shape Space within AIS, which represents bit-encoded double-precision numbers. These vectors are considered to be the B-cells within the system. Each B-cell within the population are evaluated by the objective function, $g(x)$. More formally, the B-cells are defined as a vector $\underline{v} \in P$ of bit strings of length $l = 64$ where P is the population. Empirical evidence indicates that an efficient population size for many functions is low in contrast with genetic algorithms; a typical size would be $\#P \in [3..5]$. The BCA *can* find solutions with higher P , but it converges more rapidly to the solution (using less evaluations of $g(x)$) with a smaller value for P . Results were obtained regarding this observation, but are not presented in this paper.

After evaluation by the objective function, a B-cell ($\underline{v}$) is cloned to produce a *clonal pool*, C . It should be noted that there exists a clonal pool C for *each* B-cell within the population and also that all the adaptation takes place within C . The size of C is typically the same size as the population P (but this does not have to be the case). Therefore, if P was of size 4 then each B-cell would produce 4 clones. In order to maintain diversity within the search, one clone is selected at random and each element in vector undergo a random change, subject to a certain probability. This is akin to the *metadynamics* of the immune system, a technique also employed in CLONALG, but here a separate random clone is produced, rather than utilising an existing one. Each B-cell $\underline{v}' \in C$ is then subjected to a novel contiguous somatic hypermutation mechanism. The precise form of this mutation operator will be explored in more detail below.

The BCA uses a distance function as its stopping criterion for the empirical results presented below: when it is within a certain prescribed distance from the optimum, the algorithm is considered to have converged. The BCA is outlined in figure 4.

1. *Initialisation*: create an initial random population of individuals P ;
2. *Main loop*: $\forall \underline{v} \in P$:
 - a) *Affinity Evaluation*: evaluate $g(\underline{v})$;
 - b) *Clonal Selection and Expansion*:
 - i. *Clone each B-cell*: clone $\underline{v}$ and place in clonal pool C ;
 - ii. *Metadynamics*: randomly select a clone $\underline{c} \in C$; randomise the vector;
 - iii. *Contiguous mutation*: $\forall \underline{c} \in C$, apply the contiguous somatic hypermutation operator;
 - iv. *Affinity Evaluation*: evaluate each clone by applying $g(\underline{v})$; if a clone has higher affinity than its parent B-cell $\underline{v}$, then $\underline{v} = \underline{c}$;
3. *Cycle*: repeat from step (2) until a certain stopping criterion is met.

Fig. 3. Outline of the B-Cell Algorithm

The unusual feature of the BCA is the form of the mutation operator. This operates by subjecting *contiguous* regions of the vector to mutation. The biological motivation for this is as follows: when mutation occurs on B-cell receptors, it focuses on complementarity determining regions, which are small regions on the receptor. These are sites that are primarily responsible for detecting and binding to their targets. In essence a more focused search is undertaken. This is in contrast to the method employed by CLONALG and the local search function $h(x)$, whereby although multiple mutations take place, they are uniformly distributed across the vector, rather than being targeted at a contiguous region (see figure 4). Contrastingly, as also shown in figure 4, the contiguous mutation operator, rather than selecting multiple random sites for mutation, a random site (or *hotspot*) is chosen within the vector, along with a random length; the vector is then subjected to mutation from the hotspot onwards, until the length of the contiguous region has been reached.

5 Results

Both the HGA and BCA were tested on a number of functions ranging in complexity from one to twenty dimensions, taken from [1] and [7]. It was not possible to obtain results for all functions for the CLONALG, but results for certain functions were taken from [4] for comparative purposes. In total twelve functions were tested. The parameters for the HGA were derived according to standard heuristics, with a crossover rate of 0.6 and a mutation rate of 0.001: the local search function $h(x)$ incorporated a mutation rate of $\delta \in \{2, 3, 4, 5\}$ per vector. The BCA had a clonal pool size equal to the population size. It should be noted

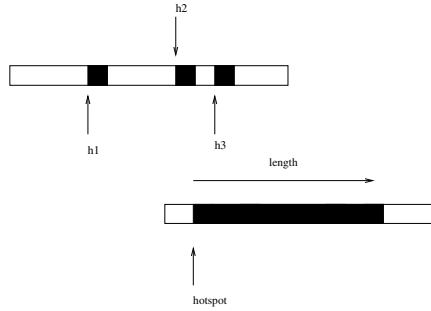


Fig. 4. Multiple-point and contiguous mutation

that all vectors consisted of bit strings of length 64 (i.e double-precision floating point numbers) and no Gray encoding was used on either the HGA or BCA.

Each experiment was run for 50 iterations and the results averaged over the runs. The functions to be optimised are given in table 1. Some of the functions may seem quite simple e.g. **f1**, **f9** with one and two dimensions respectively. However, **f12** is of twenty dimensions. An interesting characteristic of function **f11** is the presence of a second best minimum away from the global minimum. Function **f12** has a product term introducing an interdependency between the variables; this is intended to disrupt optimisation techniques that work on one function variable at a time [7].

5.1 Overview of Results

When monitoring the performance of the algorithms, two measures were employed: these were the quality of the solution found, and the number of evaluations taken to find the solution. The number of evaluations of the objective function is a measure adopted in many papers for assessing the performance of an algorithm; in case the algorithm does not converge on the optimum, the distance measure can give an estimate of how proximity to the solution. Table 2 provides a set of results averaged over 50 runs for the optimised functions. It is noteworthy that the results presented are for a population size of only 4 individuals, in order to allow for direct comparisons to be made; it should also be noted that results were obtained for population sizes ranging from 4 to 40 for both algorithms. It was found that the performance difference between the two algorithms was similar as the population size was increased. As the population sizes increased for both algorithms, the number of evaluations increased, with occasional effect on the quality of the result obtained (in terms of quality of solution found). As can be seen from table 2 both the hybrid GA and BCA perform well in finding the optimal solutions for the majority of functions. Notable exceptions are **f7** and **f9** where neither algorithm found a minimal value. In terms of the metric for quality of solutions then there seems little to distinguish the

Table 1. Functions to be Optimised

Function ID	Function	Parameters
f1	$f(x) = 2(x - 0.75)^2 + \sin(5\pi x - 0.4\pi)$ - 0.125	$0 \leq x \leq 1$
f2 (Camelback)	$f(x, y) = (4 - 2.1x^2 + \frac{x^4}{3})x^2 +$ $xy + (-4 + 4y^2)y^2$	$-3 \leq x \leq 3$ and $-2 \leq y \leq 2$
f3	$f(x) = -\sum_{j=1}^5 [j \sin((j+1)x + j)]$	$-10 \leq x \leq 10$
f4 (Branin)	$f(x, y) = a(y - bx^2 + cx - d)^2 +$ $h(1 - f) \cos(x) + h$	$a = 1, b = \frac{5.1}{4\pi^2}, c = \frac{5}{\pi},$ $d = 6, f = \frac{1}{8\pi}, h = 10$ $-5 \leq x \leq 10, 0 \leq y \leq 15$ $0 \leq y \leq 15$
f5 (Pshubert 1)	$f(x, y) = \sum_{j=1}^5 j \cos[(j+1)x + j]$	$-10 \leq x \leq 10$ and $-10 \leq y \leq 10$ and $\beta = 0.5$
f6 (Pshubert 2)	$\sum_{j=1}^5 j \cos[(j+1)y + j]$ $+ \beta[(x + 1.4513)^2 + (y + 0.80032)^2]$	as above but $\beta = 1$
f7	$f(x, y) = x \sin(4\pi x) - y \sin(4\pi y\pi) + 1$	$-10 \leq x \leq 10$ and $-10 \leq y \leq 10$
f8	$y = \sin^6(5\pi x)$	$-10 \leq x \leq 10$ and $-10 \leq y \leq 10$
f9 (quartic)	$f(x, y) = \frac{x^4}{4} - \frac{x^2}{2} + \frac{x}{10} + \frac{y^2}{2}$	$-10 \leq x \leq 10$ and $-10 \leq y \leq 10$
f10 (Shubert)	$f(x, y) = \sum_{j=1}^5 j \cos[(j+1)x + j]$ $\sum_{j=1}^5 j \cos[(j+1)y + j]$	$-10 \leq x \leq 10$ and $-10 \leq y \leq 10$
f11 (Schwefel)	$f(\vec{x}) = 418.9829n -$ $\sum_{i=1}^n x_i \sin(\sqrt{ x_i })$	$-512.03 \leq x_i \leq 511.97, n = 3.$
f12 (Griewangk)	$f(\vec{x}) = 1 + \sum_{i=1}^n \frac{x_i^2}{4000} -$ $\prod \cos(\frac{x_i}{\sqrt{i}})$	$n = 20$ and $-600 \leq x_i \leq 600$

two algorithms. This at least confirms that the BCA is performing sensibly on the functions. However, when the number of evaluations are taken into account, then a different picture emerges. These are highlighted in the table 2 and are presented as a compression rate, so the lower the rate, the fewer the number of evaluations the BCA algorithm performs when compared to the HGA. As can be seen from the table, for the majority of the functions reported, the BCA performed significantly fewer evaluations on the objective function than the HGA, but without compromising quality of the solution.

Table 2. Averaged results over 50 runs, for a population size of 4. Standard deviations are given where it was non-zero

f(x)	Min.	Minimum Found		No. Eval. of g(x)		Compression Rate
		BCA	HGA	BCA	HGA	
f1	-1.12	-1.08($\pm .49$)	-1.12	1452	6801	21.35
f2	-1.03	-1.03	-0.99($\pm .29$)	3016	12658	23.81
f3	-12.03	-12.03	-12.03	1219	3709	32.87
f4	0.40	0.40	0.40	4921	30583	16.09
f5	-186.73	-186.73	-186.73	46433	78490	59.16
f6	-186.73	-186.73	-186.73	42636	76358	55.84
f7	1	0.92	0.92($\pm .03$)	333	870	38.28
f8	1	1.00	1.00	132	484	27.27
f9	-0.35	-0.91	-0.99($\pm .29$)	2862	15894	18.01
f10	-186.73	-186.73	-186	14654	52581	27.87
f11	0	0.04	0.04	67483	131147	51.46
f12	1	1	1	44093	80062	55.07

The difference between the number of evaluations is striking. The BCA takes fewer evaluations to converge on the optimum in every case, as the percentage difference in number of evaluations illustrates. On average, it would appear that the BCA performs at least half as many evaluations as the HGA. Further experiments need to be done in comparison with other techniques, in order to further gauge evaluation performance. This is outside the scope of this paper, but is earmarked for future research.

Clearly, the BCA is not performing like the HGA. When compared to the CLONALG results, it should be noted that CLONALG also found optimal solutions for **f7** but the number of evaluations was not available.

5.2 Why Does the BCA Have Fewer Evaluations?

The question of why the BCA converges on a solution with relatively few evaluations of the objective function is one which has not yet been fully explored as part of this work, but is clearly a major avenue for investigation. It is possible that the performance of this algorithm is problem dependant (as is the case with GA's) and that the mutation operator is specifically well suited to the nature of the data representation.

It is possible that the responsibility for rapid convergence lies with the contiguous somatic hypermutation operator. Consider a fitness landscape with a number of local optima and one global optimum. Now consider a B-cell that is trapped on a local optimum; a purely local search mechanism would be unable to extricate the B-cell, since that would mean first moving to a point of lower fitness. If the mutation regime were limited to a small number of *point mutations*, it would only be able to explore its immediate neighbourhood in the fitness landscape, and so it is unlikely that it would be able to escape the local optimum.

However, the random length utilised by the contiguous somatic hypermutation operator means that it is possible for the B-cell to explore a much wider area of the fitness landscape than just its immediate neighbourhood. The B-cell may be able to jump off of a local optimum and onto the slopes of the global optimum. In much the same way, the contiguous somatic hypermutation operator can also function in a more narrow sense, analogous to local search, exploring local points in the fitness space, depending on the value of length.

Despite their intuitive appeal, these are far from formal arguments; more work will need to be undertaken to verify this hypothesis.

5.3 Differences between HGA, BCA, and CLONALG

It is important to identify, at least at a conceptual level, differences in these approaches. It should be noted that, although the BCA is clearly an evolutionary algorithm, the authors do not consider it to be a genetic or hybrid genetic algorithm: a canonical GA employs a deliberately low mutation rate, and emphasises crossover as the primary operator. Similarly, the authors do not consider the BCA to be a memetic algorithm, despite superficial similarities. It is noted that a more rigorous analysis of differences is required, but that has been earmarked for future research. It is the aim of this section to merely highlight conceptual differences for the reader. Table 3 summarises the main similarities and differences. However, it is worth expanding on these slightly.

Table 3. Summarising the main similarities and differences between BCA, HGA and CLONALG

Algorithm	Diversity	Selection	Population
BCA	Somatic Contiguous mutation	Replacement	Introduction of random B-cell. Fixed size.
HGA	Point mutation, crossover and local search	Replacement	Fixed size.
CLONALG	Affinity proportional somatic mutation	Replacement by n fittest clones	Introduction of random cells, flexible population fixed size memory population

Two major differences are the mutation mechanisms and the frequency of mutation that is employed. Both BCA and CLONALG have high levels of mutation, when compared to the HGA. However, the BCA mutates a contiguous region of the vector, whereas the other two select multiple random points in the vector space. As hypothesised above, this may give the BCA a more focused search, which helps the algorithm to converge with fewer evaluations. It is also noteworthy that neither AIS algorithms employ crossover, as this does not occur within the immune system.

The replacement of individuals within the population also varies between algorithms. Within both the HGA and BCA, when a new clone has been evaluated and is found to be better than an existing member of the population, the existing member is simply replaced with the new clone. Alternatively, in CLONALG a number n of the memory set are replaced, rather than just one. However, it should be noted that within the HGA the concept of a clone does not exist, as crossover rather than cloning is employed. This means that within the BCA there is a certain amount of enhanced parallelism, since copies of the cloned B-cell have a chance to explore the immediate neighbourhood within the vector space, by providing extra coverage of the neighbourhood. In contrast, it is again hypothesised that the HGA loses this extra parallelism through the crossover mechanism.

6 Conclusions and Future Work

This work has presented an algorithm inspired by how the immune system creates and matures B-cells, called the B-cell algorithm. A striking feature of the B-cell algorithm is its performance in comparison to a hybrid genetic algorithm. A unique aspect of the BCA is its use of a contiguous hypermutation operator, which, it has been hypothesised, is responsible for its enhanced performance. A first test would be to use this operator in a standard GA to assess the performance gain (or not) that the operator brings. This will allow for useful conclusions to be drawn about the nature of the mutation operator. A second useful direction for future work would be to further test the BCA against other algorithms and widen the scope and type of functions tested; another would be to test its inherent ability to optimise multimodal functions. It has been noted that CLONALG is suitable for multimodal optimisation [4] as an inherent property of the algorithm; it would be worthwhile evaluating if this is the case for the BCA. Perhaps the most illuminating piece of work would be to test the hypothesis regarding the effect of the contiguous hypermutation operator on convergence of the algorithm.

References

1. Andre, J., Siarry, P. and Dognon, T. An improvement of the standard genetic algorithm fighting premature convergence in continuous optimisation. *Advances in Engineering Software*. **32**. p. 49–60, 2001.
2. Berger, J., Sassi, J and Salois, M. A Hybrid Genetic Algorithm for the Vehicle Routing Problem with Time Windows and Itinerary Constraints, Proceedings of the Genetic and Evolutionary Computation Conference, 1999, 1, 44–51, Orlando, Florida, USA, Morgan Kaufmann. 1-55860-611-4,
3. Burke E.K., Elliman D.G. and Weare R.F., A hybrid genetic algorithm for highly constrained timetabling problems, 6th International Conference on Genetic Algorithms (ICGA'95, Pittsburgh, USA, 15th-19th July 1995), Morgan Kaufmann, San Francisco, CA, USA, pages 605–610, 1995

4. de Castro L. Von Zuben F. Clonal selection principle for learning and optimisation. *IEEE Transactions on Evolutionary Computation*. 2002.
5. de Castro L and Timmis J. Artificial immune systems: a new computational intelligence approach Springer-Verlag. ISBN 1-85233-594-7. 2002
6. de Castro L and Timmis J. An artificial immune network for multimodal optimisation In *2002 Congress on Evolutionary Computation. Part of the 2002 IEEE World Congress on Computational Intelligence*, pages 699–704, Honolulu, Hawaii, USA, May 2002. IEEE.
7. Eiben, A and van Kemenade, C. Performance of multi-parent crossover operators on numerical function optimization problems Technical Report TR-9533, Leiden University, 1995.
8. Farmer, J.D., Packard, N.H., and Perelson, A. The Immune System, Adaptation and Machine Learning. *Physica*, 1986. **22**(D): p. 187-204
9. Forrest S., Hofmeyr S. and Somayaji S. *Computer Immunology*. Communications of the ACM. 40(10). pages 88–96. 1997
10. Goldberg, D. and Voessner, S. Optimizing global-local search hybrids, *Proceedings of the Genetic and Evolutionary Computation Conference*, 1, 13–17, Morgan Kaufmann, Orlando, Florida, USA, 1-55860-611-4, 220–228, 1999.
11. Hajela, P. and Yoo, J. Immune network modelling in design optimisation. In *New Ideas in Optimisation*. D. Corne, M. Dorigo and F. Glover (eds), McGraw-Hill. pp. 203–215, 1999.
12. Hart, E. and Ross, P. The evolution and analysis of a potential antibody library for use in job-shop scheduling. In *New Ideas in Optimisation*. Corne, D., Dorigo, M. and Glover, F.(eds), p. 185–202, 1999.
13. Jerne, N.K. Towards a network theory of the immune system. *Annals of Immunology*, 1974. **125C**: p. 373–389.
14. Kephart, J. A biologically inspired immune system for computers. *Artificial Life IV. 4th International Workshop on the Synthesis and Simulation of Living Systems*. MIT Press, 1994.
15. Lamlum, H., et. al. The type of somatic mutation at APC in familial adenomatous polyposis is determined by the site of the germline mutation: a new facet to Knudson's 'two-hit' hypothesis. *Nature Medicine*, 1999, **5**: pages 1071–1075.
16. Nguyen, H. Yoshihara, I., Yamamori, M. and Yasunaga, M. A parallel hybrid genetic algorithm for multiple protein sequence alignment, *Proceedings of the 2002 Congress on Evolutionary Computation CEC2002*, 309–314, 2002, IEEE Press.
17. Rosin-Arbesfeld, R., Townsley, F. and Bienz, M. The APC tumour suppressor has a nuclear export function. *Letters to nature*, 2000, **406**: pages 1009–1012.
18. Timmis, J. and Neal, M. A resource limited artificial immune system for data analysis. *Knowledge Based Systems*. **14**(3-4): p. 121–130, 2001.
19. Coello, C. Coello and Cruz Cortes, N. An approach to solve multiobjective optimization problems based on an artificial immune system, *Proceedings of the 1st International Conference on Artificial Immune Systems (ICARIS)* 1, 212–221, 2002

A Scalable Artificial Immune System Model for Dynamic Unsupervised Learning

Olfa Nasraoui¹, Fabio Gonzalez², Cesar Cardona¹, Carlos Rojas¹, and
Dipankar Dasgupta²

¹ Department of Electrical and Computer Engineering, The University of Memphis
Memphis, TN 38152

{onasraou, ccardona, crojas}@memphis.edu

² Division of Computer Sciences, The University of Memphis
Memphis, TN 38152

{fgonzalz, ddasgupt}@memphis.edu

Abstract. Artificial Immune System (AIS) models offer a promising approach to data analysis and pattern recognition. However, in order to achieve a desired learning capability (for example detecting all clusters in a data set), current models require the *storage and manipulation* of a large network of B Cells (with a number often exceeding the number of data points in addition to all the pairwise links between these B Cells). Hence, current AIS models are far from being scalable, which makes them of limited use, even for medium size data sets.

We propose a new scalable AIS learning approach that exhibits superior learning abilities, while at the same time, requiring *modest* memory and computational costs. Like the natural immune system, the strongest advantage of immune based learning compared to current approaches is expected to be its ease of adaptation in dynamic environments. We illustrate the ability of the proposed approach in detecting clusters in noisy data.

Keywords. Artificial immune systems, scalability, clustering, evolutionary computation, dynamic learning

1 Introduction

Natural organisms exhibit powerful learning and processing abilities that allow them to survive and proliferate generation after generation in ever changing and challenging environments. The natural immune system is a powerful defense system that exhibits many signs of cognitive learning and intelligence [1,2]. Several Artificial Immune System (AIS) models [3,4] have been proposed for data analysis and pattern recognition. However, in order to achieve a desired learning capability (for example detecting all clusters in a data set), current models require the *storage and manipulation* of a large network of B Cells (with a number of B Cells often exceeding the number of data points, and for network based models, all the pairwise links between these B Cells). Hence, current AIS models are far from being scalable, which makes them of limited use, even for medium size data sets. In this paper, we propose a new AIS learning approach for

clustering, that addresses the shortcomings of current AIS models. Our approach exhibits improved learning abilities and *modest* complexity. The rest of the paper is organized as follows. In Section 2, we review some current artificial immune system models that have been used for clustering. In Section 3, we present a new dynamic AIS model and learning algorithm designed to address the challenges of Data Mining. In Section 4, we illustrate using the proposed Dynamic AIS model for robust cluster detection. Finally, in Section 5, we present our conclusions.

2 Artificial Immune System Models

Artificial Immune Systems have been investigated and practical applications developed notably by [5,6,7,3,8,1,4,9,10,11,12,13,14]. The immune system (lymphocyte elements) can behave as an alternative biological model of intelligent machines, in contrast to the conventional model of the neural system (neurons). Of particular relevance to our work, is the Artificial Immune Network (AIN) model. In their attempt to apply immune system metaphors to machine learning, Hunt and Cooke based their model [3] on Jerne's Immune Network theory [15]. The system consisted of a network of B cells used to create antibody strings that can be used for DNA classification. The resource limited AIN (RLAINE) model [9] brought improvements for more general data analysis. It consisted of a set of ARBs (Artificial Recognition Balls), each consisting of several identical B cells, a set of antigen training data, links between ARBs, and cloning operations. Each ARB represents a single n -dimensional data item that could be matched by Euclidean distance to an antigen or to another ARB in the network. A link was created if the affinity (distance) between 2 ARBs was below a Network Affinity Threshold parameter, NAT , defined as the average distance between all data items in the training set. Other immune network models have been proposed, notably by De Castro and Von Zuben[4].

It is common for the ARB population to grow at a prolific rate in AINE [3,16], as well as other derivatives of AINE, though to a lesser extent [9,11]. It is also common for the ARB population to converge rather prematurely to a state where a few ARBs matching a small number of antigens overtake the entire population. Hence, any enhancement that can reduce the size of this repertoire while still maintaining a reasonable approximation/representation of the antigen population (data) can be considered a significant step in immune system based data mining.

3 Proposed Artificial Immune System Model

In all existing artificial immune network models, the number of ARBs can easily reach the same size as the training data, and even exceed it. Hence, storing and handling the network links between all ARB pairs makes this approach unscalable. We propose to reduce the storage and computational requirements related to the network structure.

3.1 A Dynamic Artificial B-Cell Model Based on Robust Weights: The D-W-B-Cell Model

In a dynamic environment, the antigens are presented to the immune network one at a time, with the stimulation and scale measures re-updated with each presentation. It is

more convenient to think of the antigen index, j , as monotonically increasing with time. That is, the antigens are presented in the following chronological order: $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N$. The Dynamic Weighted B-Cell (*D-W-B-cell*) represents an influence zone over the domain of discourse consisting of the training data set. However, since data is dynamic in nature, and has a temporal aspect, data that is more current will have higher influence compared to data that is less current/older. Quantitatively, the influence zone is defined in terms of a weight function that decreases not only with distance from the antigen/data location to the D-W-B-cell prototype / best exemplar as in [11], but also with the time since the antigen has been presented to the immune network. It is convenient to think of time as an additional dimension that is added to the D-W-B-Cell compared to the classical B-Cell, traditionally statically defined in antigen space only. For the i^{th} D-W-B-cell, DWB_i , we define the following weight/membership function after J antigens have been presented:

$$w_{ij} = w_i (d_{ij}^2) = e^{-\left(\frac{d_{ij}^2}{2\sigma_i^2} + \frac{(J-j)}{\tau}\right)} \quad (1)$$

where d_{ij}^2 is the distance from antigen $\mathbf{x}_j$ (j^{th} antigen encountered by the immune network) to D-W-B-cell, DWB_i . The stimulation level, after J antigens have been presented to DWB_i , is defined as the density of the antigen population around DWB_i :

$$s_{ai,j} = \frac{\sum_{j=1}^J w_{ij}}{\sigma_i^2}, \quad (2)$$

The scale update equations are found by setting $\frac{\partial s_{ai,j}}{\partial \sigma_i^2} = 0$, and deriving incremental update equations, to obtain the following approximate incremental equations for stimulation and scale, after J antigens have been presented to DWB_i .

$$s_{ai,J} = \frac{e^{-\frac{1}{\tau}} W_{i,J-1} + w_{iJ}}{\sigma_{i,J}^2}, \quad (3)$$

$$\sigma_{i,J}^2 = \frac{e^{-\frac{1}{\tau}} \sigma_{i,J-1}^2 W_{i,J-1} + w_{iJ} d_{iJ}^2}{2 \left(e^{-\frac{1}{\tau}} W_{i,J-1} + w_{iJ} \right)}. \quad (4)$$

where $W_{i,J-1} = \sum_{j=1}^{J-1} w_{ij}$ is the sum of the contributions from the $(J-1)$ previous antigens, $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_{J-1}$, to D-W-B-Cell i , and $\sigma_{i,J-1}^2$ is its previous scale value.

3.2 Dynamic Stimulation and Suppression

We propose incorporating a dynamic stimulation factor, $\alpha(t)$, in the computation of the D-W-B-cell stimulation level. The static version of this factor is a classical way to simulate memory in an immune network by adding a compensation term that depends on other D-W-B-cells in the network [3]. In other words, a group of intra-stimulated D-W-B-cells can self-sustain themselves in the immune network, even after the antigen that caused their creation disappears from the environment. However, we need to put a limit on the time span of this memory so that truly outdated patterns do not impose an

additional superfluous (computational and storage) burden on the immune network. We propose to do this by an annealing schedule on the stimulation factor. This is done by allowing each group of D-W-B-cells to have their own stimulation coefficient, and to have this stimulation coefficient decrease with the age of the sub-net). In the absence of a recent antigen that succeeds in stimulating a given subnet, the age of the D-W-B-cell increases by 1 with each antigen presented to the immune system. However, if a new antigen succeeds in stimulating a given subnet, then the age calculation is modified by refreshing the age back to zero. This makes extremely old sub-nets die gradually, if not re-stimulated by more recent relevant antigens. Incorporating a dynamic suppression factor in the computation of the D-W-B-cell stimulation level is also a more sensible way to take into account internal interactions. The suppression factor is not intended for memory management, but rather to control the proliferation and redundancy of the D-W-B-cell population. In order to understand the combined effect of the proposed stimulation and suppression mechanism, we consider the following two extreme cases: (i) When there is positive suppression (competition), but no stimulation. This results in good population control and no redundancy. However, there is no memory, and the immune network will forget past encounters. (ii) When there is positive stimulation, but no suppression, there is good memory but no competition. This will cause the proliferation of the D-W-B-cell population or maximum redundancy. Hence, there is a natural tradeoff between redundancy/memory and competition/reduced costs.

3.3 Organization and Compression of the Immune Network

We define *external interactions* as those occurring between an antigen (external agent) and the D-W-B-cell in the immune network. We define *internal interactions* as those occurring between one D-W-B-cell and all other D-W-B-cells in the immune network. Figure 1(a) illustrates internal (relative to D-W-B-cell_k) and external interactions (caused by an external agent called "Antigen"). Note that the number of possible interactions is immense, and this is a serious bottleneck in the face of all existing immune network based learning techniques [3,9,11]. Suppose that the immune network is compressed by clustering the D-W-B-cells using a linear complexity approach such as K Means. Then the immune network can be divided into several *subnetworks* that form a parsimonious view of the entire network. For global low resolution interactions, such as the ones between D-W-B-cells that are very different, only the *inter-subnetwork interactions* are germane. For higher resolution interactions such as the ones between similar D-W-B-cells, we can drill down inside the corresponding subnetwork and afford to consider all the *intra-subnetwork interactions*. Similarly, the external interactions can be compressed by considering interactions between the antigen and the subnetworks instead of all the D-W-B-cells in the immune network. Note that the centroid of the D-W-B-cells in a given subnetwork/cluster is used to summarize this subnetwork, and hence to compute the distance values that contribute in the internal and external interaction terms. This divide and conquer strategy can have significant impact on the number of interactions that need to be processed in the immune network. Assuming that the network is divided into roughly K equal sized subnetworks, then the number of internal interactions in an immune network of N_B D-W-B-cells, can drop from N_B^2 in the uncompressed net-

work, to $\left(\frac{N_B}{K}\right)^2$ *intra-subnetwork interactions* and $K - 1$ *inter-subnetwork interactions* in the compressed immune network. This clearly can approach linear complexity as $K \rightarrow \sqrt{N_B}$. Figure 1(c) illustrates the reduced internal (relative to D-W-B-cell_k) interactions in a compressed immune network. Similarly the number of external interactions relative to each antigen can drop from N_B in the uncompressed network to K in the compressed network. Figure 1(b) illustrates the reduced external (relative to external agent "Antigen") interactions. Furthermore, the compression rate can be modulated by choosing the appropriate number of clusters, $K \approx \sqrt{N_B}$, when clustering the D-W-B-cell population, to maintain linear complexity, $O(N_B)$.

Sufficient summary statistics for each cluster of D-W-B-cells are computed, and can later be used as approximations in lieu of repeating the computation of the entire suppression/stimulation sum. The summary statistics are in the form of average dissimilarity within the group, cardinality of the group (number of D-W-B-cells in the group), and density of the group.

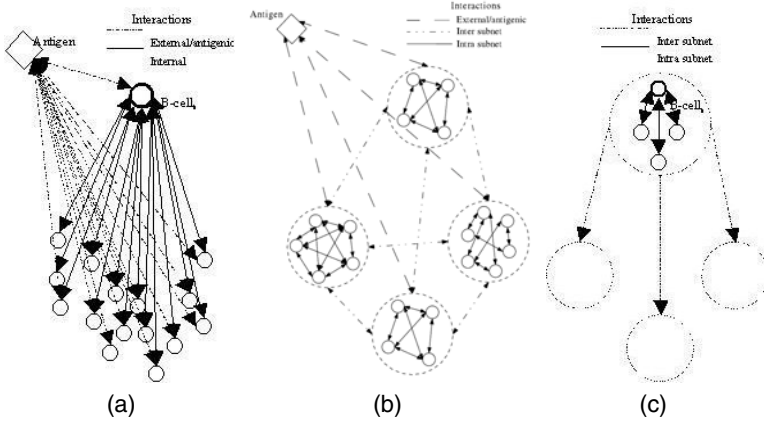


Fig. 1. Immune network interactions: (a) without compression, (b) with compression, (c) Internal Immune network interactions with compression

3.4 Effect of the Network Compression on Interaction Terms

The D-W-B-cell specific computations can be replaced by subnet computations in a compressed immune network. The stimulation and scale values become

$$s_i = s_{a_i, J} + \alpha(t) \frac{\sum_{l=1}^{N_B^i} w_{il}}{\sigma_{i, J}^2} - \beta(t) \frac{\sum_{l=1}^{N_B^i} w_{il}}{\sigma_{i, J}^2}, \quad (5)$$

where $s_{ai,J}$ is the pure antigen stimulation given by (3) for D-W-B-cell $_i$; and N_B^i is the number of B-cells in the subnetwork that is closest to the J^{th} antigen. This will modify the D-W-B-cell scale update equations to become

$$\sigma_{i,J}^2 = \frac{1}{2} \frac{e^{-\frac{1}{\tau}} \sigma_{i,J-1}^2 W_{i,J-1} + w_{iJ} d_{iJ}^2 + \alpha(t) \sum_{l=1}^{N_B^i} w_{il} d_{il}^2 - \beta \sum_{l=1}^{N_B^i} w_{il} d_{il}^2}{2 \left(e^{-\frac{1}{\tau}} W_{i,J-1} + w_{iJ} \right) + \alpha(t) \sum_{l=1}^{N_B^i} w_{il} - \beta \sum_{l=1}^{N_B^i} w_{il}}. \quad (6)$$

3.5 Cloning in the Dynamic Immune System

The D-W-B-cells are cloned (i.e, duplicated together with all their intrinsic properties such as scale value) in proportion to their stimulation levels relative to the average stimulation in the immune network. However, to avoid preliminary proliferation of good B-Cells, and to encourage a diverse repertoire, new B-Cells do not clone before they are mature (their age, t_i exceeds a lower limit t_{min}). They are also not removed from the immune network regardless of their stimulation level. Similarly, B-cells with age $t_i > t_{max}$ are frozen, or prevented from cloning, to give a fair chance to newer B-Cells. This means that

$$N_{clones_i} = K_{clone} \frac{s_i}{\sum_{k=1}^{N_{D-W-B-cell}} s_k} \text{ if } t_{min} \leq t_i \leq t_{max}. \quad (7)$$

3.6 Learning New Antigens and Relation to Outlier Detection

Somatic hypermutation is a powerfull natural exploration mechanism in the immune system, that allows it to learn how to respond to new antigens that have never been seen before. However, from a *computational* point of view, this is a very costly operation since its complexity is exponential in the number of features. Therefore, we model this operation in the artificial immune system model by an instant antigen duplication whenever an antigen is encountered that fails to activate the entire immune network. A new antigen, x_j is said to activate the i^{th} B-Cell, if its contribution to this B-Cell, w_{ij} exceeds a minimum threshold w_{min} . Antigen duplication is a simplified rendition of the action of a special class of cells called *dendritic* cells whose main purpose is to *teach other immune cells such as B-cells to recognize new antigens*. Dendritic cells (which have long been mistaken to be part of the nervous system), and their role in the immune system, have only recently been understood. We refer to this new antigen duplication, a *dendritic injection*, since it essentially injects new information in the immune system.

3.7 Proposed Scalable Immune Learning Algorithm for Clustering Evolving Data

Scalable Immune Based Clustering for Evolving Data

```

Fix the maximal population size  $N_B$ ;
Initialize D-W-B-cell population and  $\sigma_i^2 = \sigma_{init}$  using the first batch of the input anti-
gens/data;
Compress immune network into  $K$  subnets using 2-3 iterations of  $K$  Means;
Repeat for each incoming antigen  $\mathbf{x}_j$  {
  Present antigen to each subnet centroid in network and determine the closest subnet;
  IF antigen activates closest subnet Then {
    Present antigen to each D-W-B-cell,  $D\text{-}W\text{-}B\text{-}cell_i$ , in closest immune subnet;
    Refresh this D-W-B-cell's age ( $t = 0$ ) and update  $w_{ij}$  using (1);
    Update the compressed immune network subnets incrementally;
  }
  ELSE
    Create by dendritic injection a new D-W-B-cell =  $\mathbf{x}_j$  and  $\sigma_i^2 = \sigma_{init}$ ;
  Repeat for each D-W-B-cell $i$  in closest subnet only {
    Increment age ( $t$ ) for D-W-B-cell $i$ ;
    Compute D-W-B-cell $i$ 's stimulation level using (5);
    Update D-W-B-cell $i$ 's  $\sigma_i^2$  using (6);
  }
  Clone and mutate D-W-B-cells;
  IF population size  $> N_B$  Then
    Kill worst excess D-W-B-cells, or leave only subnetwork representatives of oldest subnet-
    works in main memory;
    Compress immune network periodically (after every  $T$  antigens), into  $K$  subnets using 2-3
    iterations of  $K$  Means;
}

```

3.8 Comparison to Other Immune Based Clustering Techniques

Because of paucity of space, we review only some of the most recent and most related methods. The Fuzzy AIS [11] uses a richer knowledge representation for B-cells as provided by fuzzy memberships that not only model different areas of the same cluster differently, but are also robust to noise and outliers, and allow a dynamic estimation of scale unlike all other approaches. The Fuzzy AIS obtains better results than [9] with a reduced immune network size. However, its batch style processing required storing the entire data set and all intra-network interaction affinities.

The Self Stabilizing AIS (SSAIS) algorithm [12] maintains stable immune networks that do not proliferate uncontrollably like in previous versions. However, a single NAT threshold is not realistic for data with clusters of varying size and separation, and SSAIS is rather slow in adapting to new/emerging patterns/clusters. Even though SSAIS does not require storage of the entire data set, it still stores and handles interactions between all the cells in the immune network. Because the size of this network is comparable to that of the data set, this approach is not scalable.

The approach in [13] relies exclusively on the antigen input and not on any internal stimulation or suppression. Hence the immune network has no memory, and would not be

able to adapt in an incremental scenario. Also, the requirement to store the entire dataset (batch style) and the intense computations of all pairwise distances to get the initial NAT value, make this approach unscalable. Furthermore, a single NAT value and a drastic winner-takes-all pruning strategy may impact diversity and robustness on complex and noisy data sets.

In [14], an approach is presented that exploits the analogy between immunology and sparse distributed memories. The scope of this approach is different from most other AIS based methods for clustering because it is based on binary strings, and clusters represent different schemas. This approach is scalable, since it has linear complexity, and works in an incremental fashion. Also, the gradual influence of data inputs to all clusters avoids undesirable winner-take-all effects of most other techniques.

Finally, the aiNet algorithm [4] evolves a population of antibodies using clonal selection, hypermutation and apoptosis, and then uses a computationally expensive graph theoretic technique to organize the population into a network of clusters.

Table 1 summarizes the characteristics of several immune based approaches to clustering, in addition to the K Means algorithm. The last row lists typical values reported in the experimental results in these papers. Note that all immune based techniques, as well as most evolutionary type clustering techniques are expected to benefit from insensitivity to initial conditions (*reliability*) by virtue of being population based. Also, techniques that require storage of the entire data set or a network of immune cells with a size that is comparable to that of the data set in main memory, are not scalable in memory. The criterion Density/distance/Partition/ refers to whether a density type of fitness/stimulation measure is used or one that is based on distance/error. Unlike Distance and Partitioning based methods, Density type methods directly seek dense areas of the data space, and can find more good clusters, while being robust to noise.

Table 1. Comparison of proposed Scalable Immune Learning Approach with Other Immune Based Approaches for Clustering and K Means

Approach →	<i>Proposed AIS</i>	<i>Fuzzy AIS</i> [11]	<i>RLAINE</i> [9]	<i>SSAIS</i> [12]	<i>Wierzchon</i> [13]	<i>SOSDM</i> [14]	<i>aiNet</i> [4]	<i>K Means</i>
Reliability/Insensitivity to initialization	yes	yes	yes	yes	yes	yes	yes	no
Robustness to noise	yes	yes	no	no	no	moderately	no	no
Scalability in time (linear)	yes	no	no	no	no	yes	no	yes
Scalability in space (memory)	yes	no	no	no	no	yes	no	no
Maintains Diversity	yes	yes	no	yes	not clear	yes	yes	N/A
does not requires No. Clusters	yes	yes	yes	yes	yes	yes	yes	no
Quickly Adapts to New Patterns	yes	no	no	no	no	yes	yes	no
Robust Individualized Scale Estimation	yes	yes	no	no	no	no	no	no
Density/Distance/Partition based?	Density	Density	Distance	Distance	Distance/ Partition	Distance/ Partition	Distance/ Partition	Distance/ Partition
batch/incremental: passes(size of data) (for incremental: required passes over <i>entire</i> data set data to learn <i>new</i> cluster)	incremental: 1(2000)	batch: 39 (600) (Fig. 2 in [11])	batch: 20 (150) (Fig. 5(b) in [9])	incremental: 10,000(25) (Fig. 10 in [12])	batch: 15 (100) (Fig. 5 in [12])	incremental:) 1-10(40) (Fig. 5 in [13])	batch: 10 (50) (Fig. 6 in [4])	batch: typically a few passes

4 Experimental Results

Clean and noisy 2-dimensional sets, with roughly 1000 to 2000 points, and between 3 and 5 clusters, are used to illustrate the performance of the proposed immune based approach. The implementation parameters were as follow: The first 0.02% of the data are used to create an initial network. The initial value for the scale was $\sigma_{init} = 0.0025$ (an upper radial bound derived based on the range of normalized values in $[0, 1]$). B-cells were only allowed to clone past the age of $t_{min} = 2$, and the cloning coefficient was 0.97. The maximum B-cell population size was 30 (an extremely small number considering the size of the data), the mutation rate was 0.01, $\tau = 1.5$, and the compression rate, K varied between 1 and 7. The network compression was performed after every $T = 40$ antigens have been processed. The evolution of the D-W-B-cell population for 3 noisy clusters, after a *single* pass over the antigens, presented in random order, is shown in Figure 2, superimposed on the original data set. The results for the same data set, but with antigens presented in the order of the clusters is shown in Figure 3, with the results of RLAIN [9] in Fig. 3 (d). This scenario is the most difficult (worst) case for single-pass learning, as it truly tests the ability of the system to memorize the old patterns, adapt to new patterns, and still avoid excessive proliferation. Unlike the proposed approach, RLAIN is unable to adapt to new patterns, given the same amount of resources. Similar experiments are shown for a data set of five clusters in Figure 4 and 5. Since this is an unsupervised clustering problem, it is not important that a cluster is modeled by one or several D-W-B-cells. In fact, merging same-cluster cells is trivial since we have not only their location estimates, but also their individual robust scale estimates. Finally, we illustrate the effect of the compression of the immune network by showing the final D-W-B-cell population for different compression rates corresponding to $K = 1, 3, 5$ on the data set with 3 clusters, in Fig. 6. In the last case ($K = 5$), the immune interactions have been practically reduced from quadratic to linear complexity by using $K \approx \sqrt{N_B}$. It is worth mentioning that despite the dramatic reduction in complexity, the results are virtually indistinguishable in terms of quality. The effect of compression is further illustrated for the data set with 5 clusters, in Fig. 7. The antigens were presented in the most challenging order (one cluster at a time), and in a single pass. In each case, the proposed immune learning approach succeeds in detecting dense areas after a single pass, while remaining robust to noise.

5 Conclusion

We have introduced a new robust and adaptive model for immune cells, and a scalable immune learning process. The D-W-B-cell, modeled by a robust weight function, defines a gradual influence region in the antigen, antibody, and time domains. This is expected to condition the search space. The proposed immune learning approach succeeds in detecting dense areas/clusters, while remaining robust to noise, and with a very modest D-W-B-cell population size. Most existing methods work with B-cell population sizes often exceeding the size of the data set, and can suffer from premature loss of good detected immune cells. The proposed approach is favorable from the points of view of scalability, as well as quality of learning. Quality comes in the form of diversity

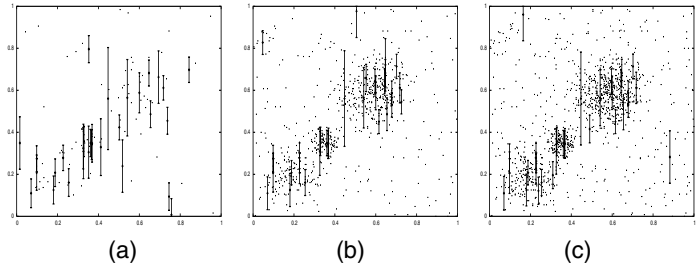


Fig. 2. Single Pass Results on a Noisy antigen set presented one at a time in random order: Location of D-W-cells and estimated scales for data set with 3 clusters after processing (a) 100 antigens, (b) 700 antigens, and (c) all 1133 antigens

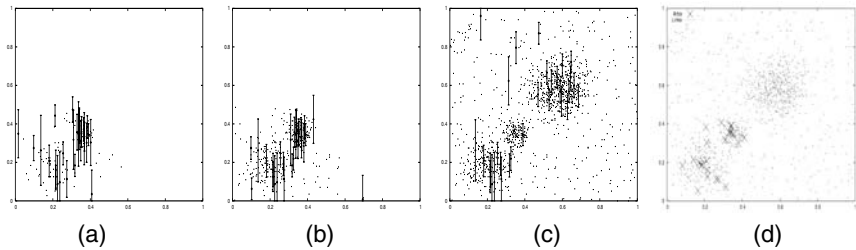


Fig. 3. Single Pass Results on a Noisy antigen set presented one at a time in the same order as clusters, (a, b, c): Location of D-W-B-cells and estimated scales for data set with 3 clusters after processing (a) 100 antigens, (b) 300 antigens, and (c) all 1133 antigens, (d) RLAINÉ's ARB locations after presenting all 1133 antigens

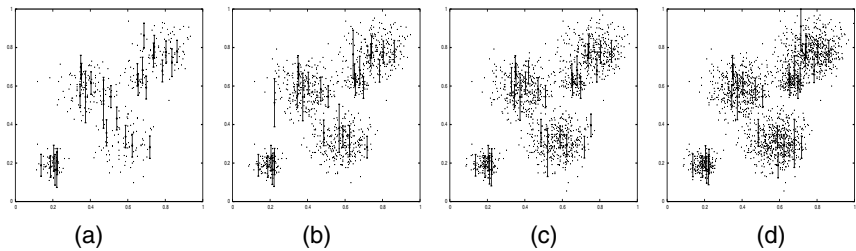


Fig. 4. Single Pass Results on a Noisy antigen set presented one at a time in random order: Location of D-W-B-cells and estimated scales for data set with 5 clusters after processing (a) 400 antigens, (b) 1000 antigens, and (c) 1300 antigens, (d) all 1937 antigens

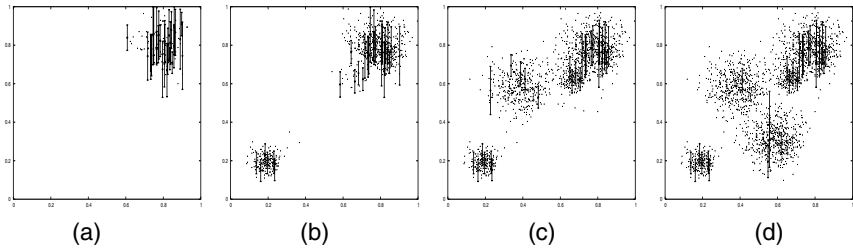


Fig. 5. Single Pass Results on a Noisy antigen set presented one at a time in the same order as clusters: Location of D-W-B-cells and estimated scales for data set with 5 clusters after processing (a) 100 antigens, (b) 700 antigens, and (c) 1300 antigens, (d) all 1937 antigens

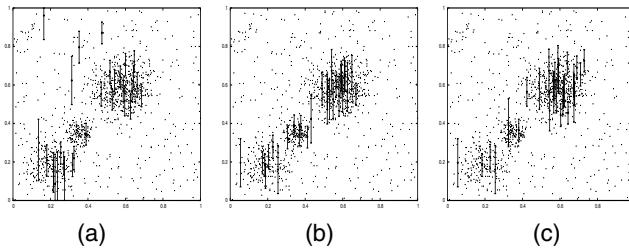


Fig. 6. Effect of Compression rate on Immune Network: Location of D-W-B-cells and estimated scales for data set with 3 clusters (a) $K = 1$, (b) $K = 3$, (c) $K = 5$

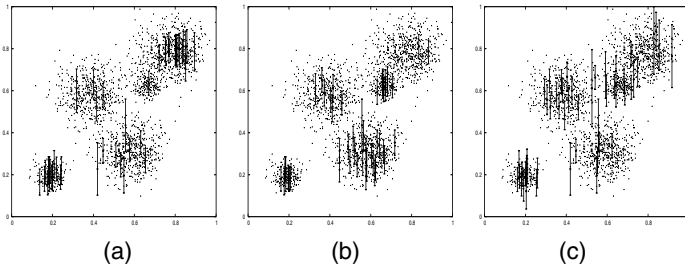


Fig. 7. Effect of Compression rate on Immune Network: Location of D-W-B-cells and estimated scales for data set with 5 clusters (a) $K = 3$, (b) $K = 5$, (c) $K = 7$

and continuous adaptation as new patterns emerge. We are currently investigating the use of our scalable immune learning approach to extract patterns from evolving Web clickstream and text data for Web data mining applications.

Acknowledgment. This work is partially supported by a National Science Foundation CAREER Award IIS-0133948 to Olfa Nasraoui and support from Universidad Nacional de Colombia for Fabio Gonzalez.

References

1. D. Dasgupta, *Artificial Immune Systems and Their Applications*, Springer Verlag, 1999.
2. I. Cohen, *Tending Adam's Garden*, Academic Press, 2000.
3. J. Hunt and D. Cooke, "An adaptative, distributed learning system, based on immune system," in *IEEE International Conference on Systems, Man and Cybernetics*, Los Alamitos, CA, 1995, pp. 2494–2499.
4. L. N. De Castro and F. J. Von Zuben, "An evolutionary immune network for data clustering," in *IEEE Brazilian Symposium on Artificial Neural Networks*, Rio de Janeiro, 2000, pp. 84–89.
5. J.D. Farmer and N.H. Packard, "The immune system, adaptation and machine learning," *Physica*, vol. 22, pp. 187–204, 1986.
6. F.J. Varela H. Bersini, "The immune recruitment mechanism: a selective evolutionary strategy," in *Fourth International Conference on Genetic Algorithms*, San Mateo, CA, 1991, pp. 520–526.
7. S. Forrest, A. S. Perelson, L. Allen, and R. Cherukuri, "Self-nonself discrimination in a computer," in *IEEE Symposium on Research in Security and Privacy*, Los Alamitos, CA, 1994.
8. D. Dasgupta and S. Forrest, "Novelty detection in time series data using ideas from immunology," in *5th International Conference on Intelligent Systems*, Reno, Nevada, 1996.
9. J. Timmis and M. Neal, "A resource limited artificial immune system for data analysis," *Knowledge Based Systems*, vol. 14, no. 3, pp. 121–130, 2001.
10. T. Knight and J. Timmis, "Aine: An immunological approach to data mining," in *IEEE International Conference on Data Mining*, San Jose, CA, 2001, pp. 297–304.
11. O. Nasraoui, D. Dasgupta, and F. Gonzalez, "An artificial immune system approach to robust data mining," in *Genetic and Evolutionary Computation Conference (GECCO) Late breaking papers*, New York, NY, 2002, pp. 356–363.
12. M. Neal, "An artificial immune system for continuous analysis of time-varying data," in *1st International Conference on Artificial Immune Systems*, Canterbury, UK, 2002, pp. 76–85.
13. Wierczon and U. Kuzelewska, "Stable clusters formation in an artificial immune system," in *1st International Conference on AIS*, Canterbury, UK, 2002, pp. 68–75.
14. E. Hart and P. Ross, "Exploiting the analogy between immunology and spares distributed memories: A system for clustering non-stationary data," in *1st International Conference on Artificial Immune Systems*, Canterbury, UK, 2002, pp. 49–58.
15. N. K. Jerne, "The immune system," *Scientific American*, vol. 229, no. 1, pp. 52–60, 1973.
16. J. Timmis, M. Neal, and J. Hunt, "An artificial immune system for data analysis," *Biosystems*, vol. 55, no. 1, pp. 143–150, 2000.

Developing an Immunity to Spam

Terri Oda and Tony White

Carleton University
terri@zone12.com,
arpwhite@scs.carleton.ca

Abstract. Immune systems protect animals from pathogens, so why not apply a similar model to protect computers? Several researchers have investigated the use of an artificial immune system to protect computers from viruses and others have looked at using such a system to detect unauthorized computer intrusions. This paper describes the use of an artificial immune system for another kind of protection: protection from unsolicited email, or spam.

1 Introduction

The word “spam” is used to denote the electronic equivalent of junk mail. This typically includes advertisements (unsolicited commercial email or UCE) or other messages sent in bulk to many recipients (unsolicited bulk email or UBE). Although spam may also include viruses, typically the term is used to refer to the less destructive classes of email.

In small quantities, spam is simply an annoyance but easily discarded. In larger quantities, however, it can be time-consuming and costly. Unlike traditional junk mail, where the cost is borne by the sender, spam creates further costs for the recipient and for the service providers used to transmit mail. To make matters worse, it is difficult to detect all spam with the simple rule-based filters commonly available.

Spam is similar to computer viruses because it keeps mutating in response to the latest “immune system” response. If we don’t find a technological solution to spam, it will disable Internet email as a useful medium, just as viruses threatened to disable the PC revolution. [1]

Although many people would consider this statement a little over-dramatic, there is definitely real need for methods of controlling spam (unsolicited email).

This paper will look at a new mechanism for controlling spam: an artificial immune system (AIS). The authors of this paper have found no other research involving creation of a spam-detector based on the function of the mammalian immune system, although the immune system model has been applied to the similar problem of virus detection [2].

2 The Immune System

To understand how an artificial immune system functions, we need to consider the mammalian immune system upon which it is based. This is only a very general overview and simplification of the workings of the immune system which uses information from several sources [3], [4]. A more complete and accurate description of the immune system can be found in many biology texts.

In essence, the job of an immune system is to distinguish between self and potentially harmful non-self elements.

The harmful non-self elements of particular interest are the *pathogens*. These include viruses (e.g. Herpes simplex), bacteria (e.g. *E. coli*), multi-cellular parasites (e.g. Malaria) and fungi. From the point of view of the immune system, there are several features that can be used to identify a pathogen: the cell surface, and soluble proteins called *antigens*.

In order to better protect the body, an immune system has many layers of defence: the skin, physiological defences, the innate immune system and the acquired immune system. All of these layers are important in building a full viral defence system, but since the acquired immune system is the one that this spam immune system seeks to emulate, it is the only one that we will describe in more detail.

2.1 The Acquired Immune System

The acquired immune system is comprised mainly of *lymphocytes*, which are types of white blood cells that detect and destroy pathogens. The lymphocytes detect pathogens by binding to them.

There are around 10^{16} possible varieties of antigen, but the immune system has only 10^8 different antibody types in its *repertoire* at any given time. To increase the number of different antigens that the immune system can detect, the lymphocytes bind only approximately to the pathogens. By using this approximate binding, the immune system can respond to new pathogens as well as pathogens that are similar to those already encountered. The higher affinity the surface protein receptors (called *antibodies*) have for a given pathogen, the more likely that lymphocyte will bind to it. Lymphocytes are only activated when the bond reaches a threshold level, that may be different for different lymphocytes.

Creating the detectors. In order to create lymphocytes, the body uses a “library” of genes that are combined randomly to produce different antibodies. Lymphocytes are fairly short-lived, living less than 10 days, usually closer to 2 or 3. They are constantly replaced, with something on the order of 100 million new lymphocytes created daily.

Avoiding Auto-immune Reactions. An *auto-immune* reaction is one where the immune system attacks itself. Obviously this is not desirable, but if lymphocytes are created randomly, why doesn’t the immune system detect self?

This is done by self-tolerization. In the thymus, where one class of lymphocytes matures, any lymphocyte that detects self will either be killed or simply not selected. These specially self-tolerized lymphocytes (known as T-helper cells) must then bind to a pathogen before the immune system can take any destructive action. This then *activates* the other lymphocytes (known as B-cells).

Finding the Best Fit. (Affinity maturation) Once lymphocytes have been activated, they undergo cloning with hypermutation. In hypermutation, the mutation rate is 10^9 times normal. Three types of mutations occur:

- point mutations,
- short deletions,
- and insertion of random gene sequences.

From the collection of mutated lymphocytes, those that bind most closely to the pathogen are selected. This hypermutation is thought to make the coverage of the antigen repertoire more complete. The end result is that a few of these mutated cells will have increased affinity for the given antigen.

3 Spam as the Common Cold

Receiving spam is generally less disastrous than receiving an email virus. To continue the immune system analogy, one might say spam is like the common cold of the virus world – it is more of an inconvenience than a major infection, and most people just deal with it. Unfortunately, like the common cold, spam also has so many variants that it is very difficult to detect reliably, and there are people working behind the scenes so the “mutations” are intelligently designed to work around existing defences.

Our immune systems do not detect and destroy every infection before it has a chance to make us feel miserable. They do learn from experience, though, remembering structures so that future responses to pathogens can be faster. Although fighting spam may always be a difficult battle, it seems logical to fight an adaptive “pathogen” with an adaptive system.

We are going to consider spam as a pathogen, or rather a vast set of varied pathogens with similar results, like the common cold. Although one could say that spam has a “surface” of headers, we will use the entire message (headers and body) as the antigen that can be matched.

4 Building a Defence

4.1 Layers Revisited

Like the mammalian immune system, a digital immune system can benefit from layers of defence [5]. The layers of spam defence can be divided into two broad categories: social and technological. The proposed spam system is a technological defence, and would probably be expected to work alongside other defence strategies. Some well-known defences are outlined below.

Social Defences. Many people are attempting to control spam through social methods, such as suing senders of spam [6], legislation prohibiting the sending of spam [7], or more grassroots methods [8].

Technological Defences. To defend against spam, people will attempt to make it difficult for spam senders to obtain their real email address, or use clever filtering methods. These include two of particular interest for this paper:

SpamAssassin [9] uses a large set of heuristic rules.

Bayesian/Probabilistic Filtering [10] [11] uses “tokens” that are rated depending on how often they appear in spam or in real mail. Probabilistic filters are actually the closest to the proposed spam immune system, since they learn from input.

Some solutions, such as the Mail Abuse Prevention System (MAPS) Realtime Blackhole List (RBL) fall into both the social and the technological realms. RBL provides a solution to spam through blocking mail from networks known to be friendly or neutral to spam senders [12]. This helps from a technical perspective, but also from a social perspective since users, discovering that their mail is being blocked, will often petition their service providers to change their attitudes.

4.2 Regular Expressions as Antibodies

Like real lymphocytes, our digital lymphocytes have receptors that can bind to more than one email message. This is done by using *regular expressions* (patterns that match a variety of strings) as antibodies. This allows use of a smaller gene library than would otherwise be necessary, since we do not need to have all possible email patterns available. This has the added advantage that, given a carefully-chosen library, a digital immune system could be able to detect spam with only minimal training.

The library of gene sequences is represented by a library of regular expressions that are combined randomly to produce other regular expressions. Individual “genes” can be taken from a variety of sources:

- a set of heuristic filters (such as those used by SpamAssassin)
- an entire dictionary
- several entire dictionaries for different languages
- a set of strings used in code, such as HTML and Javascript, that appears in some messages
- a list of email addresses and URLs of known spam senders
- a list of words chosen by a trained or partially-trained Bayesian Filter

The combining itself can be done as a simple concatenation, or with wild-cards placed between each “gene” to produce antibodies that match more general patterns.

Unfortunately, though this covers the one-to-many matching of antibodies to antigens, there is no clear way to choose which of our regular expression antibodies has the *best* match, since regular expressions are handled in a binary (matches/does not match) way. Although an arbitrary “best match” function could be applied, it is probably just as logical to treat all the matching antibodies equally.

4.3 Weights as Memory

Theories have proposed that there may be a longer-lived lymphocyte, called a memory B-cell, that allows the immune system to remember previous infections. In a digital immune system, it is simple enough to create a special subclass of lymphocytes that is very long-lived, but doing this may not give the desired behaviour.

While a biological immune system has access to all possible self-proteins, a spam immune system cannot be completely sure that a given lymphocyte will not match legitimate messages in the future. Suppose the user of the spam immune system buys a printer for the first time. Previously, any message with the phrase “inkjet cartridges” was spam (e.g. “CHEAP INKJET CARTRIDGES ONLINE – BUY NOW!!!”), but she now emails a friend to discuss finding a store with the best price for replacement cartridges. If her spam immune system had long-lived memory B-cells, these would continue to match not only spam, but also the legitimate responses from her friend that contain that phrase.

In order to avoid this, we need a slightly more adaptive memory system in that it can *unlearn* as well as learn things. A simple way to model this is to use weights for each lymphocyte.

In the mammalian immune system, pathogens are detected partially because many lymphocytes will bind to a single pathogen. This could easily be duplicated, but matching multiple copies of a regular expression antibody is needlessly computationally intensive. As such, we use the weights as a representation of the number of lymphocytes that would bind to a given pathogen.

When a lymphocyte matches a message that the user has designated as spam, the lymphocyte’s weight is then incremented (e.g. by a set amount or a multiple of current weight). Similarly, when a lymphocyte matches something that the user indicates is not spam, then the weight is decremented.

Although the lymphocyte weights can be said to represent numbers of lymphocytes, it is important to note that these weights can be negative, representing lymphocytes which, effectively, detect self. Taking a cue from SpamAssassin, we use the sum of the positive and negative weights as the final weight of the message. If the final weight is larger than a chosen threshold, it can be declared as spam. (Similarly, messages with weights smaller than a chosen threshold can be designated non-spam.)

The system can be set to learn on its own from existing lymphocytes. If a new lymphocyte matches a message that the immune system has designated spam, then the weight of the new lymphocyte could be incremented. This increment would probably be less than it would have been with a human-confirmed spam message, since it is less certain to be correct. Similarly, if it matches a message designated as non-spam, its weight is decremented.

When a false positive or negative is detected, the user can force the system to re-evaluate the message and update all the lymphocytes that match that message. These incorrect choices are handled using larger increments and decrements so that the automatic increment or decrement is overridden by new weightings

based on the correction. Thus, the human feedback can override the adaptive learning process if necessary.

In this way, we create an adaptive system that learns from a combination of human input and automated learning.

An Algorithm for Aging and Cell Death. Lymphocytes “die” (or rather, are deleted) if they fall below a given weight and a given age (e.g. a given number of days or a given number of messages tested). This simulates not only the short lifespan of real lymphocytes, but also the negative selection found in the biological immune system.

We benefit here from being less directly related to the real world. Since there is no good way to be absolutely sure that a given lymphocyte will not react to the wrong messages, co-stimulation by lymphocytes that are guaranteed not to match legitimate messages would be difficult. Attempting to simulate this behaviour might even be counter-productive with a changing “self.” For this prototype, we chose to keep the negatively-weighted, self-detecting lymphocytes in this prototype to help balance the system without co-stimulation as it occurs in nature. Thus, cell death occurs only if the absolute value of the weight falls below a threshold. It should be possible to create a system which “kills” off the self-matching lymphocytes as the self changes, but this was not attempted for this prototype.

How legitimate is removing those with weights with small absolute values?

Consider an antibody that never matches any messages (e.g. **antidisestablishmentarianism.* aperient.* kakistocracy**). It will have a weight of 0, and there is no harm in removing it since it does not affect detection. Even a lymphocyte with a small absolute weight is not terribly useful, since small absolute weights mean that the lymphocyte has only a small effect on the final total. It is not a useful indicator of spam or non-spam, and keeping it does not benefit the system.

A simple algorithm for artificial lymphocyte death would be:

```

if (cell is past “expiry date”) {
  decrement weight magnitude
  if (abs(cell weight) < threshold) {
    kill cell
  } else {
    increment expiry date
  }
}

```

The decrement of the weight is to simulate forgetfulness, so that if a lymphocyte has not had a match in a very long time, it can eventually be recycled. This decrement should be very small or could even be none, depending on how strong a memory is desired.

4.4 Mutations?

Since we have no algorithm defined to say that one regular expression is a better match than another, we cannot use mutation easily to find matches that are more accurate. Despite this, there could still be a benefit to mutating the antibodies of a digital immune system, since it would be possible (although perhaps unlikely) that some of the new antibodies created would match *more* spam, even if there was no clear way to define a *better* match with the current message. Mutations could be useful for catching words that spam senders have hyphenated, misspelled intentionally, or otherwise altered to avoid other filters. At the very least, mutations would have a higher chance of matching with similar messages than lymphocytes created by random combinations from the gene library.

Mutations could occur in two ways:

1. They could be completely random, in which case some of the mutated regular expressions will not parse correctly and will not be usable.
2. They could be mutated according to a scheme similar to that of Automatically Defined Functions (ADF) in genetic programming [13]. This would leave the syntax intact so that the result is a legitimate regular expression.

It would be simpler to write code that would do random mutations, but then harder to check the syntax of the mutated regular expressions if we wanted to avoid program crashing when lymphocytes with invalid antibodies try to bind to a message. These lymphocytes would simply die through negative selection during the hypermutation process, since they are not capable of matching with anything. Conversely, it would be harder to code the second type, but it would not require any further syntax-checking.

Another variation on mutation is an adaptive library. In some cases, no lymphocytes will match a given message. If this message is tagged as spam by the user, then the system will be unable to “learn” more about the message because no weights will be updated. To avoid this situation, the system could generate new gene sequences based upon the message. These could be “tokens” as described by Graham [11], or random sections of the email. These new sequences, now entered into the gene pool, will be able to match and learn about future messages.

5 Prototype Implementation

Our implementation has been done in Perl because of its great flexibility when it comes to working with strings.

The gene library and lymphocytes are stored in simple text files. Figure 1 shows the contents of a short library file. In the library, each line is a regular expression. Each “gene” is on a separate line.

Figure 2 shows the contents of a short lymphocytes file. For the lymphocytes, each line contains the weight, the cell expiry date and the antibody regular expression. The format uses the string “###” (that does not occur in the library)

```

remove.{1,15}subject
Bill.{0,10}1618.{0,10}TITLE.{0,10}(III|\#3)
check or money order
\s+href=['"]?www\.
money mak(?:ing|er)
(?:100%|completely|totally|absolutely) (?-i:F)ree

```

Fig. 1. Sample Library Entries

```

-5###1040659390###result of
10###1040659390###\<BODY.*bgcolor="#?[^f]
-98###1040659390###(?:\$|US\$|usd?).?\d{2,3}(?:\.\d)?
52###1040661344###trial version of CommuniGate.*opt-in
33###1040661344###click .{0,30}(?:here|below)

```

Fig. 2. Sample Lymphocytes

as a separator, so each lymphocyte definition string has <weight>###<expiry time>###<regular expression antibody>.

Our AIS spam detector has three phases:

1. Generation of Lymphocytes
2. Application of Lymphocytes
3. Culling of Lymphocytes

Figure 3 gives an overview of the life cycle of a spam lymphocyte. The generation and culling scripts are run on a regular schedule, and the application script is called with appropriate arguments to increment, decrement, or simply detect, as appropriate. For example, if the target message has been tagged as spam by the user, the application script is called with a large increment so that all the lymphocytes that match that message are incremented. The target message could fall into six potential classes: Tagged as spam or legitimate by the user, tagged as spam or legitimate by the immune system, or tagged as a false positive or a false negative by the user.

5.1 Generation

This script reads all the “genes” in the library and combines them randomly to produce the number of antibodies specified as the first argument.

The antibodies are created by choosing a gene randomly, then effectively flipping a coin to see if another gene should be appended to the resulting regular expression antibody. The probability of appending to an antibody can be changed relatively easily. For the purposes of testing, we chose 50% as the likelihood for each gene addition. (Although an antibody using fewer genes will tend to match more frequently, we wished to see if some longer antibodies matched spam more exclusively.) To add another gene, another gene is chosen

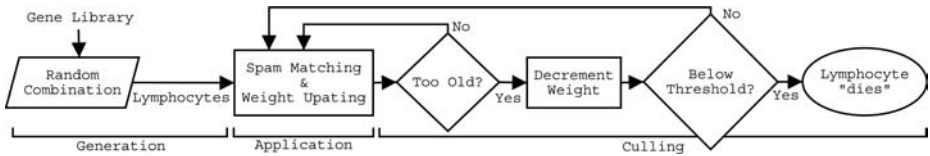


Fig. 3. Life cycle of a digital lymphocyte

randomly and the original string is concatenated with the string .* (match any number of characters) followed by the new gene sequence. This intermediary .* is used to increase the number of possible matches.

The lymphocyte is then given a weight of 0 and an expiry date. The resulting lymphocytes are written to disk in the order of generation.

There is no attempt made to avoid duplicates since doing so could be costly and having duplicates should not overly adversely affect results, only giving some antibodies a stronger effect by duplicating. This effect should be balanced by other antibodies.

5.2 Application

The application script reads in all the lymphocytes and applies them one by one to each message in the file specified as first argument. The total weighted sums are outputted for each message, along with the Subject: and From: lines to give some context.

The antibodies are applied in a case-insensitive manner. This was chosen so that more matches would be found, although a quick comparison using the same lymphocytes showed that matching in a case-sensitive manner did not make a significant difference in the weightings.

This script takes a second argument that indicates the increment or decrement to be applied to each lymphocyte that matches messages in this file. The updated lymphocytes are then re-written to the lymphocytes file.

5.3 Culling

The culling script reads in all the lymphocytes and kills off cells based on the algorithm described earlier in this paper. Ideally, the culling script would call the generation script to replace those lymphocytes removed, but for the moment this is done manually.

6 Results

6.1 Initial Parameters and Data

We initially used a set of 150 regular expression "genes" as a library. Many of these were simplified SpamAssassin heuristics. SpamAssassin contains over 700

very complex heuristics, but our gene library contained only pieces of those, typically pieces that matched only a few shorter phrases. These heuristics were used to reduce the training time required for the lymphocytes, since many people would prefer a solution which required little training time. (If individual users were not interested in training their own artificial immune systems at all, it could be possible give them spam-trained lymphocytes, much like users currently download known virus patterns for their computer virus scanners.)

To test and train the lymphocytes, a set of mailbox files (in mbox format) were used with messages compiled from personal mail, Grant Taylor’s collection of spam [14], and mailing list archives of lists known to contain both legitimate posts and spam.

In a quick test with 100 lymphocytes generated, it was soon apparent that either this size of library was insufficient to detect the selection of spam being used to train the lymphocytes or more lymphocytes needed to be generated. Over 40% of the messages known to be spam were not detected by any of the lymphocytes, and thus the immune system could not learn about these messages. This may indicate a need for a process akin to affinity maturation where, knowing a message to be spam, the digital immune system produces lymphocytes that match it.

With a larger library of around 300 genes and set of around 1000 lymphocytes generated, the tests take much longer to run, so only a few iterations were run after the initial testing phase.

Increment and decrement values were chosen so that a false positive or negative was given a more significant weight than either training values or automated learning values. Because missed messages are more irritating to users, we wanted to encourage faster learning from these messages so that users would not have to keep reporting similar missed messages. The following values were used:

Determined by	User (training)	AIS (automated)	User (false positive/negative)
Spam/non-spam weight adjustment	5/-5	1/-1	10/-10

6.2 Output

The initial spam training gave weights ranging from 0 to over 10^5 . As such, some lymphocytes had a much larger ability than others to affect the final sum. Some of these were balanced after initial “self-tolerization” training against legitimate messages.

After the initial training phase of over 1600 spam messages and 1000 legitimate messages was done, we forced a cell death on lymphocytes with absolute weights less than 4. Out of the 1000 lymphocytes originally generated, less than 100 were saved, leading credence to the idea that a larger lymphocyte population would be necessary to do proper tests.

These remaining 100 lymphocytes were used as a quick test against a heterogeneous (spam and non-spam) mailbox with approximately 1200 messages.

The results were very promising. We set the threshold at 10, but even with this small threshold, only two false positives occurred (where a legitimate message was found as spam).

- One was a message sent by a random stranger and was vastly different in content and style from the messages used initially to set the weights.
- The other was a message from a current spam control system. It contained a listing of the Subject: and From: lines of the messages blocked that day.

Although the second false positive seems reasonably explicable, this behaviour is problematic for layering spam defences. Some sort of special exception may have to be found so that messages of this sort from a reporting system will not be caught. This could be through manual addition of an overriding lymphocyte (one with a very large negative weight), for example. Training the system normally should be possible, but could neutralise otherwise-useful lymphocytes.

Similarly, the spam messages not tagged by the system were unusual spams, unlike those that had been seen earlier by the system.

There were some negatively-valued messages, but most of the messages were given weights of 0, as would be expected given that the library was drawn from spam-matching regular expressions used by SpamAssassin. If a more broad-based library were used, then there would be more negatively-weighted lymphocytes, and thus it would be possible to have more negatively-valued messages.

Further tests after more iterations and re-running of false positives yielded similar results to those initial ones: approximately 1% of the messages scanned were false positives. Overall, the 1000 lymphocyte system was correctly identifying approximately 90% of the spam messages.

7 Conclusions

This model has some potential: even with empirically-chosen values for increments and decrements and a relatively small library of regular expressions, emergent behaviour has been observed. It is detecting spam with very few false positives in the test mailboxes. Due to the small size of the library space, it is still missing approximately 10% of the spam messages entirely. While false negatives are less damaging than false positives [15], they are still an indicator of a detector or library space that is still too small to cover the range of potential spam messages.

This initial 90% accuracy is not as good as Graham's probabilistic filter, which can work at over 99% accuracy [11]. However, while we looked at 1000 lymphocytes, Graham's system recognises over 23,000 tokens, and his algorithm has had significantly more time to mature.

7.1 Future Work

Several directions for future research are planned. A mutation model is to be included, perhaps with a function to determine better matching or preferential

weightings of lymphocytes (e.g. One could use the length of the string matched by a given regular expression as an indicator of better matching[16]). Larger and alternative gene libraries are to be created and tested. A variation self-tolerization could be used to generate only lymphocytes which match spam rather than relying upon weight-balancing through self-detecting lymphocytes. Finally, alternative weighting schemes are to be designed and compared to the performance of the algorithm described in this paper.

References

1. Hellweg, E.: What price spam? Business 2.0 (1999)
2. White, S.R., Swimmer, M., Pring, E.J., Arnold, W.C., Chess, D.M., Morar, J.F.: Anatomy of a commercial-grade immune system. Technical report, IBM Thomas J. Watson Research Center (2002)
3. Nunn, I.: Immunocomputing: The natural immune system and its computational metaphors. (2002)
<http://www.scs.carleton.ca/~arpwhite/courses/95590Y/notes/Immunocomputing.ppt>.
4. Hofmeyr, S.A.: An overview of the immune system (1997)
<http://www.cs.unm.edu/~immsec/html-imm/immune-system.html>.
5. Sakkis, G., Androutsopoulos, I., Paliouras, G., Karkaletsis, V., Spyropoulos, C.D., Stamatopoulos, P.: Stacking classifiers for anti-spam filtering of E-mail. In: Proceedings of EMNLP-01, 6th Conference on Empirical Methods in Natural Language Processing, Pittsburgh, US, Association for Computational Linguistics, Morristown, US (2001)
6. Jesdanun, A.: AOL wins \$7 million spam lawsuit. Salon.com (2002)
http://www.salon.com/tech/wire/2002/12/17/aol_spam/.
7. Email, C.A.U.: Pending legislation (2002) <http://www.cauce.org/legislation>.
8. Wendland, M.: Internet spammer can't take what he dishes out. Detroit Free Press (2002)
9. SpamAssassin: Spamassassin website (2002) <http://spamassassin.org/>.
10. Sahami, M., Dumais, S., Heckerman, D., Horvitz, E.: A bayesian approach to filtering junk E-mail. In: Learning for Text Categorization: Papers from the 1998 Workshop, Madison, Wisconsin, AAAI Technical Report WS-98-05 (1998)
11. Graham, P.: A plan for spam (2002) <http://www.paulgraham.com/spam.html>.
12. Vixie, P.: MAPS RBL rationale (2000)
<http://mail-abuse.org/rbl/rationale.html>.
13. Poli, R.: Introduction to evolutionary computation: Automatically defined functions in genetic programming (1996)
http://www.cs.bham.ac.uk/~rmp/slide_book/node7.html.
14. Taylor, G.: A collection of spam (2002)
<http://www2.picante.com:81/~{}gtaylor/download/spam.tar.gz>.
15. Androutsopoulos, I., Koutsias, J., Chandrinou, K.V., Paliouras, G., Spyropoulos, C.D.: An evaluation of naive bayesian anti-spam filtering. In Potamias, G., Moustakis, V., van Someren, M., eds.: Proceedings of the Workshop on Machine Learning in the New Information Age, 11th European Conference on Machine Learning (ECML 2000), Barcelona, Spain (2000) 9–17
16. O'Byrne, K.: Private communication (2002)

A Novel Immune Anomaly Detection Technique Based on Negative Selection

F. Niño, D. Gómez, and R. Vejar

Department of Computer Science
National University of Colombia
Bogotá, Colombia

lfnino@ing.unal.edu.co, {djgomezb, ravejaru}@unal.edu.co

1 Introduction

In this paper, a novel immune based anomaly detection technique is proposed. Specifically an Artificial Immune System (AIS), based on negative selection, is used to detect some particular kind of computer attacks (Denial of Service attacks) on a typical LAN environment.

2 AIS Architecture

In this work, the general model of an AIS consists of the following phases: data collection, data pre-processing, learning (training), post-processing of the detectors and the detection phase. The learning and detection phases are described next.

2.1 Learning Phase

Here, a representation of the environment, the antigens (negative samples) and the detectors (antibodies) is chosen. Besides, a matching rule between detectors and antigens is specified; an antigen matches a detector if the distance between them is less than a threshold value. The goal of the training process of the AIS is to find an optimum covering of the self space with a suitable set of antibodies (hyper-spheres), which are generated based on the set of sample antigens. In intrusion detection, such set of antigens will correspond to DoS attacks.

- **Generation of Antibodies.** The goal is to produce a new population of antibodies in the self space from the current population. The radius of a new antibody depends on the distance from its center to its nearest antigen. The generation of new antibodies is determined by its overlapping with existing detectors. The overlapping between two antibodies is a measure of the intersection between their corresponding hyper-spheres. Hence, the goal is to make such overlapping measure as small as possible.
- **Selection of Antibodies.** The best antibodies are selected according to the following criteria: the fittest antibodies are determined according to their radii

and their overlapping with other antibodies. Detectors with larger radius are more likely to be selected. Besides, if its overlapping measure is greater than a specified threshold, then such antibody will be discarded. Otherwise, it will become part of the next generation.

- **Cloning Antibodies.** A new set of antibodies is generated from the existing antibodies. A clone of an antibody is generated by making a copy of such antibody and moving its center at random; its radius is adjusted based on its distance to the nearest antigen.
- **Post-processing Antibodies.** The goal is to cover small regions of the self space that may remain uncovered after the learning process, and that may produce false negatives. The radius of a new antibody is adjusted using the distance to the nearest antibody.

2.2 Detection Phase

A set of unseen patterns is presented to the AIS to be classified as either corresponding to normal traffic or to DoS attacks. The degree of abnormality of a pattern is proportional to the distance between the input pattern and the set of detectors. If the distance from a detector to an input pattern is smaller than a threshold then such pattern belongs to the self set. Otherwise, it will be considered as abnormal traffic. Four fields of the header information were used as input to the AIS, namely, *source and target port, packet size and protocol ID*. This data was used to completely characterize the information that would allow distinguishing packets belonging to DoS attacks and normal traffic.

3 Experimental Results

Network traffic was collected and used during the AIS learning process. The captured data was divided into two sets: the training set, which consisted of DoS attacks, and the test set that consisted of both, normal traffic data and DoS attacks. An input pattern consisted of a sequence of several consecutive packets. The AIS was able to detect all the attacks in the training set (it consisted of 4516 patterns), 99% of the attacks were detected in the test set, and the AIS performed well in classifying normal traffic. However, some patterns corresponding to normal traffic were misclassified as possible attacks. In the best experimental results, the AIS correctly classified 88% of normal traffic.

4 Conclusions

In this paper, a novel general anomaly detection technique based on immunology was developed. One main advantage of the AIS is that it starts with a small number of detectors and a new set of antibodies is generated through an iterative process that improves the covering of the self space. The number of detectors generated during the training process was smaller than the size of the input data set because the learning process allows the detectors to have variable radius and it is possible to cover the self

space with a small number of detectors. The post-processing of the antibodies improves the performance of the AIS. In future work, the immune technique may be applied to detect other types of intrusions in computer systems, or to solve any other anomaly detection problem.

References

1. R. Bolaños and C. Cadena. Intrusion detection in Linux using neural networks (in spanish). Computer Science Thesis. National University of Colombia. Bogotá, Colombia 2002.
2. D. Dasgupta (Editor), Artificial Immune Systems and Their Applications, Publisher: Springer-Verlag, Inc. Berlin, January 1999.
3. D. Dasgupta and F. Nino. Comparison of Negative and Positive Selection Algorithms in Novel Pattern Detection, In the Proceedings of the IEEE International Conference on Systems, Man and Cybernetics (SMC), Nashville, 2000.
4. IBM. Denial of service attacks: Understanding network Vulnerabilities in www.ibm.com. 2002.

Visualization of Topic Distribution Based on Immune Network Model

Yasufumi Takama

Tokyo Metropolitan Institute of Technology
6-6 Asahigaoka, Hino, Tokyo 191-0065 JAPAN
PREST, Japan Science and Technology Corporation, JAPAN
ytakama@cc.tmit.ac.jp

Abstract. A method is presented that applies the properties of immune system for visualizing the topic distribution over a document set, as well as the topic stream through a sequence of document sets.

1 Introduction

As the Web provides us with new topics constantly, it is difficult for us to grasp the trends or change of topics. In particular, there exists the information, which cannot be found from a single document but from a sequence of document sets. We are developing a Web information visualization method that can find the topic distribution over the document set(s)[1,2]. As the fundamental method, a **plastic clustering method**[2] has been proposed for generating a topic-sensitive keyword map. One of the characteristic features of the plastic clustering method is the generation of **keyword map** as well as document clustering. The keyword's activation value is calculated based on the **immune network model**, which is also useful as the visualization metaphor to improve the understandability of the keyword map. The model of **memory cell** is also incorporated, so that it can find the topical relation among different document sets.

2 Examples of Extracted Topic Stream

The experiment shown here is performed on the sequences of Yahoo! Japan online news article sets that were issued from 17 to 21 September 2001. Fig. 1 shows the landmarks extracted from the sequence. The landmarks extracted by both methods (with/without memory cell) are indicated with dotted texture. In Fig. 1, only one landmark is reused more than once, among 18 landmarks (5.56%) when no memory cell is used. On the contrary, 3 landmarks are reused among 17 landmarks (17.6%) with using memory cell.

As the news articles within the sequence was issued just after the tragedy in N.Y., many articles contain the related topics. Two landmarks, 'Performance', 'Multiple', which are reused more than once, concern the topics related to this N.Y. tragedy. That is, 'Multiple' literally concerns the simultaneous multiple terrorist attacks occurred in New York. After that disaster, many concerts, events,

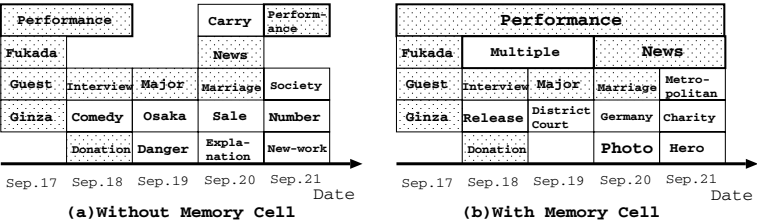


Fig. 1. Landmarks Extracted from Online News Sequence

etc., were performed/held for the purpose of charity, donation, or encouragement for the victims. The landmark ‘Performance’ mainly refer to this kind of topics.

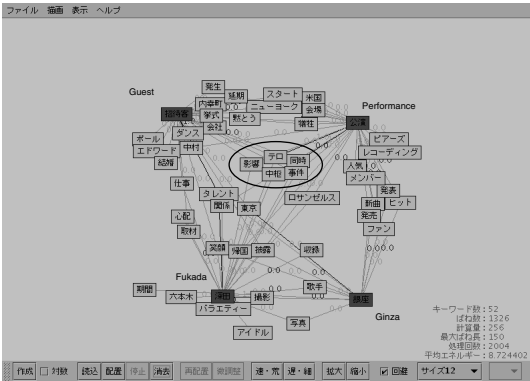


Fig. 2. Keyword Map Generated From Document Set of 17 Sep. 2001.

3 Conclusion

A method for extracting topic distribution over document space based on immune network model is presented. We are now developing the keyword map-based visualization tool that can visualize the extracted topic distribution as shown in Fig.2.

References

1. Takama, Y. and Hirota, K., “Web Information Visualization Method Employing Immune Network Model for Finding Topic Stream from Document-Set Sequence,” *J. of New Generation Computing*, Vol. 21, No. 1, pp. 49–59, 2003.

2. Takama, Y., Hirota, K., “Web Information Visualization Based on Immune Network Metaphor,” *J. of Japan Society for Fuzzy Theory and Systems*, 14, 5, pp. 472–481, 2002.

Spatial Formal Immune Network

Alexander O. Tarakanov

St. Petersburg Institute for Informatics, Russian Academy of Sciences
14-line 39, St. Petersburg 199178, Russia
sasha_tar@hotmail.com

Abstract. A notion of Spatial Formal Immune Network is proposed for pattern recognition applications.

In our previous works [4], [6], we have proposed a rigorous mathematical notion of Formal Immune Network (FIN) inspired by N. Jerne's network theory of the immune system [3]. That FIN has been introduced as Integer valued (IFIN), where B-cells are coded by integers and formed ordered sequences (populations). Such IFIN provides a discrete mathematical model of immune response, which properties described by a row of theorems. However, IFIN is insufficient for real-world applications with multi-dimensional and real valued data, like those in surveillance of the natural plague or information security.

The present work makes an attempt to solve the above problem by introducing Spatial FIN (SFIN) as an expanded FIN for pattern recognition applications.

Define SFIN as a tuple

$$SFIN = \langle R^N, m, w, \Delta h, B\text{-cells} \rangle,$$

where

$$B\text{-cell} = \langle S, P \rangle,$$

R^N is N -dimensional Euclidean space considered as the *shape space* of SFIN, according to [1];

$P \in R^N$ is N -dimensional vector (point of the shape space) considered as the Formal Protein (FP) as well as the *receptor* of the B-cell (see, e.g., [6]);

m is the number of the nearest neighbors of any B-cell;

w is the Euclidean distance considered as the *binding energy* between FPs:

$$w: R^N \times R^N \rightarrow R^1, w_{ij} = |P_i - P_j|;$$

h is real valued *mutation step*: $h \in R^1$;

S is the state indicator of B-cell: $S = \{lm, stm, del\}$, where *lm* is long-term (memory) cell, *stm* is short-term (memory) cell, and *del* is deleted cell.

The behavior of the SFIN is determined by the following rules.

All B-cells update their states simultaneously in a discrete time $t = 0, 1, 2, \dots$.

Let $\{B_i\}$ is current set (*population*) of B-cells: $S_i \neq del, i = 1, \dots, n$.

Any B-cell B_i has no more than m nearest neighbors, which correspond to the m nearest points in the shape space (rule B_neighbor).

Any B-cells B_i, B_j are *close* if $w_{ij} < h$.

Any B-cell B_i updates its state as follows.

Short-term cell B_j becomes deleted if it is close to any long-term cell B_i (rule B_delete).

In case of the deletion, consider that long-term cell B_i *recognizes* short-term cell B_j .

Short-term cell makes m copies (*proliferates*) by the mutation step to the directions of the nearest neighbors if it isn't close to any cell (rule B_prolif).

Short-term cell becomes long-term cell if it is close to any short-term cell but isn't close to any long-term cell (rule B_interf).

Long-term cell *drifts* by the mutation step to the direction of the nearest long-term cell if the cells are close (rule B_drift).

Two long-term cells B_i, B_j *fuse* if they are identical (rule B_fusion).

If any new FP (*antigen*) P_A intrudes to the SFIN, then new short-term cell (*antigen presenting cell*) appears, which expose the intruder as the receptor (rule A_pres).

The following conclusions can be done based on simple theoretical examples of immune response of SFIN, as well as on the applications of SFIN to predict risk of the plague infection and to detect intrusions in computer networks.

SFIN differs essentially from immune algorithms proposed by [2]. The algorithms represent special case of genetic algorithms, because immune cells are coded by bit strings and mutate by random rules. Unlike these features, cells of SFIN are coded by real-valued vectors and their behavior is deterministic.

Actually, SFIN represents a modification of hybrid cellular automata, which have been applied successfully to the modeling of rather complex dynamical process [5].

References

1. De Boer, R.J., Segel, L.A., Perelson, A.S.: Pattern Formation in One and Two-Dimensional Shape Space Models of the Immune System. *J. Theoret. Biol.* 155 (1992) 295–333
2. De Castro, L.N., Timmis, J.: *Artificial Immune Systems: A New Computational Intelligence Paradigm*. Springer-Verlag, Berlin Heidelberg New York (2002)
3. Jerne, N.K.: Toward a Network Theory of the Immune System. *Ann. Immunol.* 125C Paris (1974) 373–389
4. Tarakanov, A.O.: *Information Security with Formal Immune Networks*. Lecture Notes in Computer Science, Vol. 2052. Springer-Verlag, Berlin Heidelberg New York (2001) 115–126
5. Tarakanov, A., Adamatzky, A.: Virtual Clothing in Hybrid Cellular Automata. *Kybernetes (Int. J. of Systems & Cybernetics)* 31 (7/8) (2002) 1059–1072
6. Tarakanov, A.O., Skormin, V.A., Sokolova, S.P.: *Immunocomputing: Principles and Applications*. Springer-Verlag, Berlin Heidelberg New York (2003)

Focusing versus Intransitivity

Geometrical Aspects of Co-evolution

Anthony Bucci* and Jordan B. Pollack

Brandeis University, Waltham MA 02454, USA
{abucci,pollack}@cs.brandeis.edu
<http://demo.cs.brandeis.edu/>

Abstract. Recently, a minimal domain dubbed *the numbers game* has been proposed to illustrate well-known issues in co-evolutionary dynamics. The domain permits controlled introduction of features like intransitivity, allowing researchers to understand failings of a co-evolutionary algorithm in terms of the domain. In this paper, we show theoretically that a large class of co-evolution problems closely resemble this minimal domain. In particular, all the problems in this class can be embedded into an ordered, n -dimensional Euclidean space, and so can be construed as greater-than games. Thus, conclusions derived using the numbers game are more widely applicable than might be presumed. In light of this observation, we present a simple algorithm aimed at remedying focusing problems and relativism in the numbers game. With it we show empirically that, contrary to expectations, focusing in *transitive* games can be more troublesome for co-evolutionary algorithms than intransitivity. Practitioners should therefore be just as wary of focusing issues in application domains.

1 Introduction

[1] discusses a minimal substrate which can be used to illustrate several issues plaguing co-evolutionary dynamics. Individuals in this substrate are simply tuples of numbers. Because of the simplicity of the individuals, we are able to see more clearly what goes wrong when an algorithm fails to work as hoped. The authors explored two variants of the numbers game, a transitive and an intransitive one. The intransitive numbers game proved problematic for a conventional, fitness proportionate co-evolutionary algorithm. One issue which arose in the numbers game experiments which is of particular interest to us is the problem of overspecialization. Individuals had multiple dimensions on which they could vary. It was observed that some individuals would *focus* on one dimension at the expense of another. We will refer to this issue as *the focusing problem*. A detailed discussion of this problem can be found in [2].

In this paper, we will show that a large class of co-evolutionary domains, even intransitive ones, can be viewed as n -dimensional, transitive numbers games,

* Corresponding author

for some unknown dimension n . This observation raises an important question: where have the intransitivities of the original domain gone? As we will see, the mathematics of *Pareto co-evolution* [3],[4] turns intransitive cycles into sets of non-dominated individuals [5]; what remains then are the transitive relations among individuals.

Our main mathematical tool will be the partial order decomposition theorem. This theorem states that any finite partially ordered set (poset) can be decomposed into an intersection of n linear orders, for some number n called the *partial order dimension*. This dimension n is meaningful geometrically: one corollary of the theorem is that an n -dimensional poset can be embedded monotonically into $\mathbb{R}^n$. As shown in [5], co-evolution problems expressible with a payoff function $p : S \times T \rightarrow R$ (R preordered) can be regarded as preorderings on the set S of candidate solutions. Modulo technicalities caused by preordering, such co-evolution problems can therefore also be viewed as suborders of $\mathbb{R}^n$. Or, to put it simply: as n -dimensional transitive numbers games.

While this result is purely theoretical, our empirical results suggest that we may reasonably approximate the mathematics in algorithms, at least for the problems we have tried. In other words, a naïve implementation of Pareto co-evolution can effectively solve intransitive problems.

Surprisingly, we also find that a co-evolutionary algorithm with a diversity maintenance mechanism can tackle the intransitive numbers game, similarly calling into question the importance of intransitivity in assessing co-evolutionary algorithm failure. Indeed, with a *transitive* variant of the numbers game designed to emphasize the focusing problem, co-evolution with diversity maintenance fails, and the full power of Pareto co-evolution is required to solve the problem. The conclusion we draw from these results is that focusing, rather than intransitivity, is the more important domain feature to consider in some applications.

We will be considering co-evolution problems which can be expressed with a payoff function of form $p : S \times T \rightarrow R$, where R is assumed to be partially ordered. For simplicity we will assume S , T and R are finite sets; though this assumption is not strictly necessary, it does simplify arguments.¹ We make no further assumptions about structure on R . This set might consist of numbers, or it might contain symbolic values like *lose* and *win*.

This paper is organized as follows. In section 2, we set up and state the partial order decomposition theorem. Making use of the mathematical notation and terminology established in [5], we apply this theorem to co-evolution problems and prove an important corollary, the preorder decomposition theorem, which makes up the backbone of our claim that co-evolution problems have significant geometric aspects. In section 3 we describe our experiments.

¹ What we really need for most results is that the induced preorder on S be finite-dimensional, which is true of many infinite preorders too.

2 Orders and Co-evolution

In this section we set up and state the poset decomposition theorem. Our presentation borrows from [6], which should be consulted for details and proofs. We next prove a corollary which we call the preorder decomposition theorem. Finally, we apply our results to co-evolution problems of form $p : S \times T \rightarrow R$, showing that any such problem can be embedded into Euclidean n -space for some unknown n . Finally, we work through an example to illustrate the concepts.

2.1 Poset Decomposition

We will write $\leq$ for orders. We may subscript like $\leq_R$ to emphasize we are in the poset R . Recall that a *monotone* function between two posets Q and R is a function $f : Q \rightarrow R$ such that whenever $q_1 \leq_Q q_2$, then $f(q_1) \leq_R f(q_2)$. Such a function thus preserves order relations. Recall also that f is *injective* if, whenever $q_1 \neq q_2$ then $f(q_1) \neq f(q_2)$. f is an *embedding of Q into R* if it is both monotone and injective. An embedding f essentially realizes Q as being part of R .

In this section, assume all posets are finite. We begin with the notion of linear extension:

Definition 1 (Linear Extension). *A linear extension of R is a total ordering L of the elements of R which is consistent with R 's order. In other words, if S is the underlying set of both R and L , the identity function $\mathbf{1}_S : S \rightarrow S$ is monotone with respect to R in the domain and L in the range.*

Example 2. Let R have elements $\{a, b, c\}$ and relations $a \leq c, b \leq c$ (i.e., a and b are incomparable). Then one linear extension of R puts these elements in order $a \leq b \leq c$; call it L_1 . R has a second linear extension L_2 putting the elements in order $b \leq a \leq c$. In light of this example, we have the following:

Definition 3 (Linear Realizer). *A linear realizer of a (finite) poset R is a set $\{L_1, \dots, L_n\}$ of linear extensions of R such that $\bigcap_{i=1}^n L_i = R$. The intersection means that the only comparisons in $\bigcap_{i=1}^n L_i$ are the ones which are in all the L_i ; all other pairs of elements are incomparable.*

Example 4. In example 2, L_1 and L_2 constitute a linear realizer $\{L_1, L_2\}$ of R . To see this, notice that $a \leq c$ and $b \leq c$ in both L_1 and L_2 , whereas $a \leq b$ in L_1 while $b \leq a$ in L_2 . Thus, in $L_1 \cap L_2$, $a \leq c$ and $b \leq c$ whereas a and b are incomparable. These relations are exactly the ones in R ; hence, $R = L_1 \cap L_2$.

We have been leading up to the following fundamental fact about posets which we state without proof.²

² See [6] for details. The crux of the proof is to show that R has at least one finite linear realizer.

Theorem 5 (Poset Decomposition Theorem). *Every finite poset has a minimal realizer.*

The “minimal” in the name “minimal realizer” means the linear realizer contains a minimum number of linear extensions. This minimum, call it n , is the *dimension* of R ; alternately, R is called an n -dimensional poset. For instance, the poset in examples 2 and 4 is two-dimensional. The justification for using the word “dimension” is via the following two lemmas:

Lemma 6. *Every linear extension L of R gives rise to an embedding $x : R \rightarrow \mathbb{N}$.*

Proof. A linear extension of R is essentially a choice for putting the elements of R into a list. If $R = \{s_1, \dots, s_m\}$, then L will be $s_{\sigma(1)} \leq s_{\sigma(2)} \leq \dots \leq s_{\sigma(m)}$, σ being some permutation of $1, \dots, m$. Let us reindex R by defining $t_i = s_{\sigma(i)}$. Then, the mapping $x : R \rightarrow \mathbb{N}$ defined by $t_i \mapsto i$, is monotonic by construction. It is also injective, since we only index distinct elements of R . $\square$

Lemma 7 (Embedding Lemma). *Every linear realizer $\{L_1, \dots, L_n\}$ of R gives rise to an embedding $\phi : R \rightarrow \mathbb{N}^n$.*

Proof. By lemma 6, each L_i gives rise to an injective, monotone function $x_i : R \rightarrow \mathbb{N}$. These functions define a sort of coordinate system for R . Define the map $\phi : R \rightarrow \mathbb{N}^n$ by $s \mapsto (x_1(s), \dots, x_n(s))$ for all $s \in R$. Each coordinate x_i of ϕ is injective and monotone; thus ϕ itself is too. $\square$

Remark 8. In particular, if R is an n -dimensional poset, it has a minimal realizer $\{L_1, \dots, L_n\}$. By lemma 7, there is thus an embedding $\phi : R \rightarrow \mathbb{N}^n$. $\mathbb{N}^n$ embeds into $\mathbb{R}^n$, and so we see that ϕ can be regarded as embedding R into ordered, n -dimensional Euclidean space. n is minimal in this case, so R *cannot* be embedded into m -dimensional space for some smaller m . Thus the name “ n -dimensional poset.”

2.2 Applications to Co-evolution

To complete the picture, we need to see how to apply the results of the previous section to co-evolution problems. First, let us recall some important definitions and notation from [5]. For any function $f : S \times T \rightarrow R$, where S is a set and R is a poset, write S_f for the preordering induced on S by pullback, and write the order on S_f as $\leq_f$. This definition means that $s_1 \leq_f s_2$ exactly when $f(s_1) \leq_R f(s_2)$.³

Any function of form $S \times T \rightarrow R$ can be curried to a function of form $S \rightarrow [T \rightarrow R]$, where $[T \rightarrow R]$ stands for the set of all functions from T to R . If $p : S \times T \rightarrow R$, write $\lambda t.p$ for the corresponding curried function.

Consequently, starting from a co-evolution problem $p : S \times T \rightarrow R$, with R a poset, there is a corresponding *preorder* structure on the set S , namely

³ We can interpret this definition in terms of fitness functions: $s_1 \leq_f s_2$ if s_2 ’s fitness is at least as high as s_1 ’s.

$S_{\lambda t.p}$. The basic idea is that two candidate solutions s_1 and s_2 lie in the relation $s_1 \leq_{\lambda t.p} s_2$ exactly when s_2 's array of outcomes *covers* s_1 's. In other words, s_2 does at least as well as s_1 does against every possible opponent. We refer the reader to [5] for details and examples.

The poset decomposition theorem applies to partial orders, not preorders, so we need to adjust it slightly. Our approach is to observe that every preorder R comes with an equivalence relation defined: $s_1 \sim s_2$ if and only if $s_1 \leq_R s_2$ and $s_2 \leq_R s_1$. One way to think about this relation is in the context of an objective function $f : S \rightarrow \mathbb{R}$. $s_1 \sim s_2$ exactly when the individuals s_1 and s_2 have the same fitness. In a multi-objective context, $s_1 \sim s_2$ when s_1 and s_2 have the same objective vector. Given this equivalence relation, we can then prove the following:

Lemma 9. *Let R be a (finite) preorder. There is a canonical partial order $Q = R/\sim$ and a surjective, monotone function $\pi : R \rightarrow Q$ (called the projection) such that $\pi(s_1) = \pi(s_2)$ if and only if $s_1 \sim s_2$, for all $s_1, s_2 \in R$. Q is called the quotient of R .*

Proof. The proof that Q is a well-defined partial order can be found in [5]. Let us show that π is surjective and monotone. Define $\pi : R \rightarrow Q$ by $\pi(s) = [s]$ for all $s \in R$, where $[s]$ is the equivalence class of s under $\sim$. π is surjective trivially, since the only equivalence classes in Q are of form $[s]$ for some $s \in R$. To see π is monotone, observe the order on Q is $[s_1] \leq_Q [s_2]$ if and only if $s_1 \leq_R s_2$. An equivalent way to state this definition is: $\pi(s_1) \leq_Q \pi(s_2)$ if and only if $s_1 \leq_R s_2$ (this is just rewriting $[s_i]$ as $\pi(s_i)$). The “if” part proves the monotonicity of π . $\square$

Lemma 9 permits us to adapt the poset decomposition theorem (theorem 5) to preorders. If R is a preorder, form the quotient $Q = R/\sim$, which will be a partial order. Apply the embedding lemma (lemma 7) to yield an embedding $\phi : Q \rightarrow \mathbb{N}^n$ of Q in $\mathbb{N}^n$. Composing ϕ with the projection π gives a monotone function $\phi \circ \pi : R \rightarrow \mathbb{N}^n$ which we call a *pseudo-embedding* of R into $\mathbb{N}^n$. By this we mean the following. Write ψ for $\phi \circ \pi$. ψ is such that $\psi(s_1) = \psi(s_2)$ exactly when $s_1 \sim s_2$. Consequently, ψ behaves like an embedding, except that it sends equivalent individuals in R to the same point in $\mathbb{N}^n$. Otherwise, it sends non-equivalent elements in R to different points in $\mathbb{N}^n$ while preserving the order relations between them. Let us record these observations as:

Theorem 10 (Preorder Decomposition Theorem). *Every finite preorder can be pseudo-embedded into $\mathbb{N}^n$ (and thus into $\mathbb{R}^n$). Moreover, every finite preorder has a minimum n , the dimension of the preorder, for which such pseudo-embedding is possible.*

Let us examine an example to help visualize the definitions.

Example 11. Consider the following game:.

<i>p</i>	<i>rock</i>	<i>stone</i>	<i>paper</i>	<i>scissors</i>
<i>rock</i>	0	0	−1	1
<i>stone</i>	0	0	−1	1
<i>paper</i>	1	1	0	−1
<i>scissors</i>	−1	−1	1	0

This game is rock-paper-scissors with a clone of rock called “stone.” In this case, $S = T = \{rock, stone, paper, scissors\}$, and $R = \{-1, 0, 1\}$ with $-1 \leq 0 \leq 1$.

By comparing the rows of this matrix, we can see that none of the strategies dominates any of the others. Every strategy does well against at least one opponent; likewise, every strategy does poorly against at least one opponent. However, $rock \sim stone$ because their rows are identical. Consequently, the induced pre-order on $\{rock, stone, paper, scissors\}$ contains only the relations $rock \leq stone$ and $stone \leq rock$. We show this preorder in figure 1.

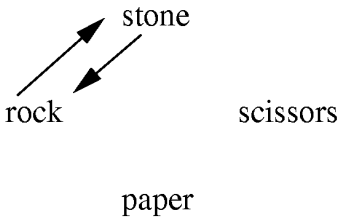


Fig. 1. The preorder rock-stone-paper-scissors displayed as a graph. An arrow between two individuals indicates a $\leq$ relationship; absence of an arrow indicates the individuals are incomparable.

When we mod out the equivalence relation $\sim$, we arrive at a partial order consisting of the three equivalence classes $\{[rock], [paper], [scissors]\}$. In this partial order, all three elements are incomparable to one another. This is, in fact, a 2-dimensional partial order, with linear realizer $\mathcal{L} = \{([rock], [paper]), ([scissors], [paper]), ([rock], [scissors])\}$ Figure 2 shows a plot of the corresponding pseudo-embedding.

2.3 Discussion

We began with a co-evolution problem which may have had intransitive cycles and other pathologies. We ended up with what amounts to an embedding of the problem into $\mathbb{N}^n$ for some unknown dimension n . $\mathbb{N}^n$ is a particularly simple partial order; in particular, it is transitive. How did we turn a pathological problem into a nice transitive one?

The first point to note is that a pseudo-embedding, while well-defined mathematically, is not easily computable. Indeed, it is has been known for some time that even learning the dimension of a poset is NP-complete [7]. Furthermore,

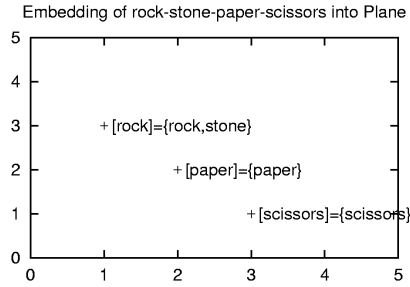


Fig. 2. The preorder rock-stone-paper-scissors pseudo-embedded into the plane $\mathbb{N}^2$

in order to compute a poset decomposition, we need to have all the elements of the poset on hand. While in many problems it is possible to *enumerate* all solutions,⁴ it is typically intractable to do so. Thus, while pseudo-embeddings exist as mathematical objects, they are at least as expensive to compute as solving the problem by brute force. This is not surprising; if we had a pseudo-embedding in hand, we could treat our problem as a greater-than game and solve it relatively easily. Pseudoembeddings thus encode a lot of information about the original problem which we have to “pay for” somehow. In essence, what we have done is reorganize the information in a problem.

A second point is that we can at least hope that the preorder decomposition of a co-evolution problem can be *approximated* by a more practical algorithm. In the next section, we will present a simple algorithm aimed at approximating the preorder decomposition of a problem and show that it works reasonably well on two instances of the numbers game.

3 Experiments

In this section we present our experimental results. We begin by recalling the numbers game, from [1]. Next, we describe the algorithms we employ and our implementation choices. Finally, we present and discuss results.

3.1 The Numbers Game

“The numbers game” [1] actually refers to a class of games. Common among them is that the set of individuals, call it S , consists of n -tuples of natural numbers. How we choose to compare individuals, and what choice we make for n , define an instance of the game. In our experiments, we will be considering two instances. In both, we will deviate somewhat from the score functions defined in [1]. Instead of returning a score, we will construct our functions to have form $p : S \times S \rightarrow \{0, 1\}$, where $S = \mathbb{N}^2$ is the set of ordered pairs of natural numbers,

⁴ Exceptions include problems with real-valued parameters.

and the function p simply says which individual is bigger (i.e., gets a bigger score or “wins” the game).

In our experiments we only present data for 2-dimensional problems, since the issues we wish to emphasize are already visible at this low dimensionality. For simplicity of presentation, we define these games for 2 dimensions only.

The Intransitive Game [IG] In this game, we first decide which dimensions of the individuals are most closely matched, and then we decide which individual is better on that dimension. The payoff function we use is:

$$p_{IG}((i_1, j_1), (i_2, j_2)) = \begin{cases} 1 & \text{if } |i_1 - i_2| > |j_1 - j_2| \text{ and } j_1 > j_2 \\ 1 & \text{if } |j_1 - j_2| > |i_1 - i_2| \text{ and } i_1 > i_2 \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

This game is intransitive; one cycle is $(1, 6)$, $(4, 5)$, $(2, 4)$ [1]. $(1, 6)$ and $(4, 5)$ are closest on the second dimension, so $(1, 6) > (4, 5)$. $(4, 5)$ and $(2, 4)$ are closest on the second dimension also, so $(4, 5) > (2, 4)$. However, $(2, 4)$ and $(1, 6)$ are closest on the first dimension, meaning $(2, 4) > (1, 6)$. Nevertheless, an individual with high values on both dimensions will tend to beat more individuals than one without, and it seems, intuitively, that the best solutions to this game are such individuals.

The Focusing Game [FG] In this game, the first and second individuals are treated asymmetrically. The second individual is scanned to see on which dimension it is highest. Then, it is compared to the first individual. The first individual is better if it is higher on the best dimension of the second individual. As a payoff function:

$$p_{FG}((i_1, j_1), (i_2, j_2)) = \begin{cases} 1 & \text{if } i_2 > j_2 \text{ and } i_1 > i_2 \\ 1 & \text{if } j_2 > i_2 \text{ and } j_1 > j_2 \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

Note that this game is transitive. However, the emphasis on one dimension at the expense of others encourages individuals to race on one of the two and neglect the second. Nevertheless, an individual which is high on both dimensions will beat more individuals than one which is focused on a single dimension.

This game is closely related to the COMPARE-ON-ONE game described in [2].⁵

3.2 Algorithms and Setup

We will compare two algorithms, a *population based co-evolutionary hillclimber* (P-CHC) and a *population based Pareto hillclimber* (P-PHC). The P-CHC has

⁵ But note that p_{FG} has range $\{0, 1\}$ and requires strict inequalities for a 1 output, whereas COMPARE-ON-ONE has range $\{-1, 1\}$ and does not require strict inequalities.

a single population. To assess the fitness of an individual, we sum how it does against all other individuals according to the game we are testing. The wrinkle is that each individual produces exactly one offspring, and the offspring can only replace its parent if it is strictly better. This algorithm uses a subjective fitness measure to assess individuals, but the constraint that an offspring can only replace its own parent is a simple form of diversity maintenance resembling deterministic crowding [8].

Our population based Pareto hillclimbing algorithm is similar in spirit to the DELPHI algorithm presented in [2]. Our P-PHC operates as follows. There are two populations, candidates and tests. The candidates are assessed by playing against all the tests. Rather than receiving a summed fitness, they receive an outcome vector, as in evolutionary multi-objective optimization [9]. The outcome vectors are then compared using Pareto dominance: candidate a is better than candidate b if a does at least as well as b does versus all tests, and does better against at least one. The tests are assessed differently, using an approximation of *informativeness* [10]. Since the outcome order is $0 < 1$, the informativeness measure presented in that paper collapses to simply counting how many pairs of candidates a test says are equal.⁶ In other words, each test has a score $f(t) = \sum_{s_1, s_2 \in S_i} \delta(p(s_1, t), p(s_2, t))$, where S_i is the current population, and $\delta(p(s_1, t), p(s_2, t))$ returns 1 if $p(s_1, t) = p(s_2, t)$, 0 otherwise. For our experiments, p will be one of p_{IG} , or p_{FG} . As in the co-evolutionary hillclimber, in the Pareto hillclimber individuals receive only one offspring, and an offspring can only replace its parent. However, the climbing is done separately in the two populations.⁷

In order to focus more closely on domain-specific problems, we do away with a bitstring genotype. In our experiments, genotype=phenotype. Individuals are simply pairs of numbers. To create a mutant, we add random noise to each coordinate with some probability. Notice there is no mutation bias in any particular direction.

We will be using a population size of 100 for all experiments. In the co-evolutionary hillclimber, the single population will be 100 individuals; in the Pareto hillclimber, there will be 50 candidates and 50 tests. The mutation rate is 100%; mutation adds +1 or -1 to each dimension. No form of crossover is used. We ran each simulation for 500 time steps.

3.3 Results

Figures 3 and 4 show performance versus evolutionary time for both P-CHC and P-PHC on the payoff functions p_{IG} and p_{FG} . Note that the co-evolutionary hillclimber out paces the Pareto hillclimber. The Pareto hillclimber must adjust not only its candidates to make an improvement, but also its tests. Updating the

⁶ Strictly speaking, this statement is not true; however, two tests which give different counts of equal candidate pairs are incomparable; thus we use it as a heuristic.

⁷ We should remark that neither of these algorithms was intended to be practical; rather, they are intended to test our ideas.

tests causes a time lag which slows down progress. The graphs are intended to be qualitative, however; what is important is that both algorithms make steady progress.

Note figure 3, the intransitive game. Unlike the algorithm used in [1], P-CHC made continuous progress on the intransitive game. Since P-CHC essentially adds only a diversity maintenance mechanism, it seems the diversity is important to the success or failure of co-evolution on this problem.

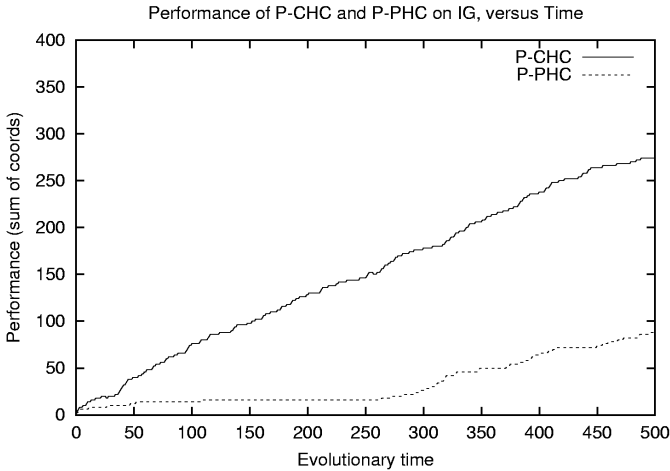


Fig. 3. Performance versus time of P-CHC and P-PHC on IG (intransitive game). Plot shows a single, typical run. Performance is measured as the sum of the coordinates; the plot shows this value for the best individual of the population at each time step.

At first glance, the Pareto co-evolution mechanism of P-PHC does not seem to be adding anything. To understand more completely what is happening, we plot in figure 5 the final candidates P-CHC found on a typical run on p_{FG} , together with the final candidates and tests which P-PHC found on a typical run. Notice how P-CHC has focused entirely on the horizontal dimension. While it made great progress there, it neglected the vertical dimension entirely. By contrast, P-PHC has maintained progress on both dimensions equally. While it did not move as far as P-CHC, it did remain balanced. Most important are the tests P-PHC found. In the plot, the tests appear to be “corralling” the candidates, keeping them in a tight group near the main diagonal. An animation of a typical run of P-PHC reveals this is indeed the case. The tests keep step behind the candidates. The same configuration of tests and candidates persists, but the group of them move slowly up and to the right, towards the better values of this game. Intuitively, we imagine the individuals found by P-CHC are brittle specialists, whereas the individuals found by P-PHC are more robust generalists.

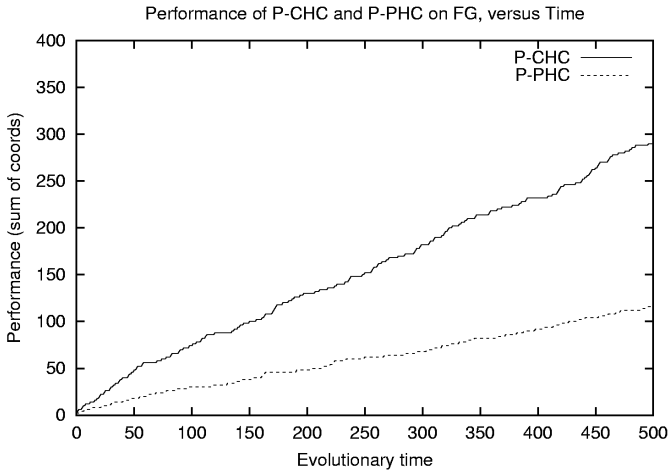


Fig. 4. Performance of P-CHC and P-PHC on FG (focusing game). Plot shows a single, typical run. Performance is measured as the sum of the coordinates of the best individual.

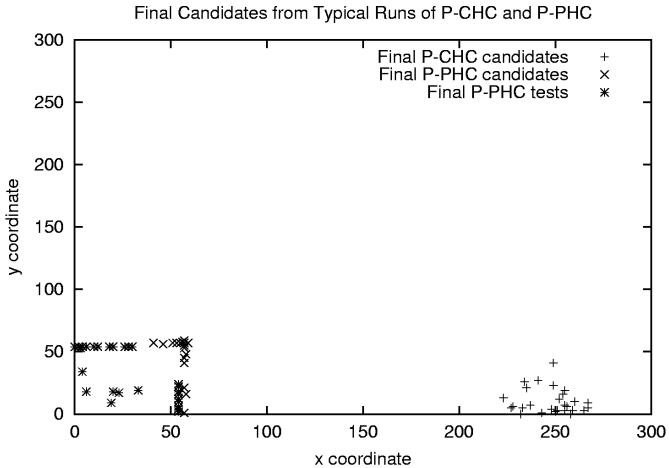


Fig. 5. Position in plane of final candidates from P-CHC run on p_{FG} (lower right), together with final candidates and tests from P-PHC (lower left). P-CHC has focused on the horizontal dimension, whereas P-PHC has improved in both dimensions. The tests which P-PHC found are arranged to “corral” the candidates along the main diagonal.

4 Conclusion

To sum up, we have shown mathematically that a wide class of co-evolution problems, if properly construed, can be looked upon as n -dimensional, transi-

tive, greater-than games. The trouble is that discovering n is an NP-complete problem in general, let alone the embedding which would permit us to convert our favorite problem domain to a nicely-behaved transitive domain. Nevertheless, we feel the mathematical result changes the face of intransitivity. We examined intransitivity experimentally, using insights gained from our mathematics, and saw that it may not be the demon it has been made out to be. Some algorithms fail because they are overspecializing on some dimensions of a game at the expense of others. While this observation is not new, the mathematical derivation of it sheds new light on the interpretation of co-evolution and suggests new algorithms which might overcome both intransitivity and overspecialization difficulties.

References

1. Watson, R., Pollack, J.: Coevolutionary Dynamics in a Minimal Substrate. In Spector, L., Goodman, E., Wu, A., Langdon, W., Voigt, H.M., Gen, M., Sen, S., Dorigo, M., Pezeshk, S., Garzon, M., Burke, E., eds.: *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO-2001*, San Francisco, CA, Morgan Kaufmann Publishers (2001)
2. de Jong, E., Pollack, J.B.: Ideal Evaluation from Coevolution. *Evolutionary Computation* (to appear)
3. Ficici, S.G., Pollack, J.B.: Pareto Optimality in Coevolutionary Learning. In: *European Conference on Artificial Life*. (2001) 316–325
4. Noble, J., Watson, R.A.: Pareto coevolution: Using performance against coevolved opponents in a game as dimensions for Pareto selection. In Spector, L., Goodman, E., Wu, A., Langdon, W., Voigt, H.M., Gen, M., Sen, S., Dorigo, M., Pezeshk, S., Garzon, M., Burke, E., eds.: *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO-2001*, San Francisco, CA, Morgan Kaufmann Publishers (2001) 493–500
5. Bucci, A., Pollack, J.B.: A Mathematical Framework for the Study of Coevolution. In: *FOGA 2002: Foundations of Genetic Algorithms VII*, San Francisco, CA, Morgan Kaufmann Publishers (2002)
6. Scheinerman, E.R.: *Mathematics: A Discrete Introduction*. 1st edn. Brooks/Cole, Pacific Grove, CA (2000)
7. Yannakakis, M.: The Complexity of the Partial Order Dimension Problem. *SIAM Journal on Algebraic and Discrete Methods* **3** (1982) 351–358
8. Mahfoud, S.W.: Crowding and Preselection Revisited. In Männer, R., Manderick, B., eds.: *Parallel Problem Solving from Nature 2*, Amsterdam, North-Holland (1992) 27–36
9. Fonseca, C.M., Fleming, P.J.: An Overview of Evolutionary Algorithms in Multi-objective Optimization. *Evolutionary Computation* **3** (1995) 1–16
10. Bucci, A., Pollack, J.B.: Order-Theoretic Analysis of Coevolution Problems: Coevolutionary Statics. In Barry, A.M., ed.: *GECCO 2002: Proceedings of the Bird of a Feather Workshops, Genetic and Evolutionary Computation Conference*, New York, AAAI (2002) 229–235

Representation Development from Pareto-Coevolution

Edwin D. de Jong

DSS Group, Utrecht University, The Netherlands
dejong@cs.uu.nl, <http://www.cs.uu.nl/~dejong/>

Abstract. Genetic algorithms generally use a fixed problem representation that maps variables of the search space to variables of the problem, and operators of variation that are fixed over time. This limits their scalability on non-separable problems. To address this issue, methods have been proposed that coevolve explicitly represented modules. An open question is how modules in such coevolutionary setups should be evaluated.

Recently, Pareto-coevolution has provided a theoretical basis for evaluation in coevolution. We define a notion of functional modularity, and objectives for module evaluation based on Pareto-Coevolution. It is shown that optimization of these objectives maximizes functional modularity. The resulting evaluation method is developed into an algorithm for variable length, open ended development of representations called *DevRep*. DevRep successfully identifies large partial solutions and greatly outperforms fixed length and variable length genetic algorithms on several test problems, including the 1024-bit Hierarchical-XOR problem.

Keywords: Development of representations, hierarchical modularity, Pareto-coevolution, Evolutionary Multi-Objective Optimization

1 Introduction

Most genetic algorithms employ a single, fixed representation that is given as part of the problem specification. To apply a genetic algorithm to a problem, a mapping has to be chosen between *genotypes* (sequences of binary or other variables) and the actual individuals they represent, *phenotypes*. We call this mapping the *representation* of the problem. For most genetic algorithms, the representation is chosen once, and does not change during the algorithm's operation. The operators of variation are typically constant too. Thus, the search space and the operators specifying the possible moves in this space cannot be adapted by the algorithm.

In combination, the use of a fixed representation and fixed operators of variation make it unlikely that different combinations of large partial solutions will be explored, see [14]. Partial solutions take the form of schemata, i.e. partial specifications of a genotype. Both forming large partial solutions and mixing existing

genotypes can be achieved in isolation. However, the standard genetic algorithm makes it unlikely for mutually exclusive partial solutions¹ to persist, since no mechanism for protecting mutually exclusive partial solutions is present. Methods that cannot represent information about mutually exclusive partial solutions are unlikely to address non-separable problems².

When niching is used, mutually exclusive partial solutions can persist. Still, unless they are explicitly represented, crossover is unlikely to respect the boundaries of such partial solutions. This limits the potential to combine large partial solutions. A solution is to represent partial solutions, *modules*, explicitly. Several methods taking this approach have been investigated so far, including GLiB [1], ADF's [8], ARL [12], ADSN [7], and SEAM [20].

The development of modules allows algorithms to start searching in terms of *combinations* of variables. Thus, such algorithms can be viewed as adapting the *representation* of the problem during search [1,12,3]. H-BOA, an apparently quite different approach to address hierarchical problems, also represents partial solutions explicitly, by using decision trees [10].

So far, there has been a lack of theory to guide the development of algorithms forming modules, in particular regarding module evaluation. Methods that simultaneously evolve modules and assemblies are instances of coevolution, as module evaluation is based on interactions with assemblies. Recently, the paradigm of *Pareto-coevolution* has provided a theoretical basis for evaluation in coevolution [6,19,2,4].

Here, we will apply Pareto-coevolution to the question of how modules may be evaluated. The first algorithm to use Pareto-coevolution for module evaluation is Watson's SEAM algorithm [20]. SEAM is designed for fixed length problems. We study the question of how combinations of large partial solutions may be explored for *variable length* problems in an open-ended setup where modules can be combined into new modules recursively and indefinitely. The problems we are interested in are large search problems that have *structure*. A search problem is large if it requires a long solution, in terms of the original variables. If information from part of the search space can be used to predict (better than random) information about other parts of the search space, we will say the problem has structure.

The structure of the paper is as follows. First, our algorithm is gradually introduced, by discussing structural and functional modularity (2), coevolution of modules and assemblies and Pareto-coevolution (Section 3), evaluation of modules based on Pareto-coevolution (Section 4), and finally the algorithm following from this principle (Section 5). The test problems are described in Section 6. Experimental results are reported in Section 7, followed by discussion and conclusions.

¹ Two partial solutions are mutually exclusive if they assign conflicting values to one or more variables.

² A problem is *separable* if each variable has a single optimal setting, independent of the other variables [16].

2 Structural versus Functional Modularity

We will distinguish between *structural modularity*, a characteristic of algorithms, and *functional modularity*, the modularity present in a problem. Watson [17] defines notions of structural and functional modularity based on the structure and behavior of a dynamical system. Here, we will say any method that represents partial solutions explicitly features *structural modularity*. While any grouping of elements leads to structural modularity, the value of a modular representation strongly depends on the particular grouping that is chosen. Ideally, the structural modules constructed by an algorithm should correspond to the functional modules present in the problem.

The primitives of a problem are the basic elements that may occur in a genotype, typically numerical values, e.g. $\{0, 1\}$, or actions or operators. A **module** is a sequence of primitives, and has a unique identifier. We will assume that for every primitive there is a module containing only that primitive. Thus, without loss of generality, individuals can be viewed as sequences of modules. Candidate solutions are sequences of modules, and are called **assemblies**.

The **compact** representation of an assembly describes the assembly in terms of the modules of which it consists, using the modules' unique ID's. The assembly's **expressed** form consists of the concatenation of the sequences of primitives represented by its modules. The sizes of the compact and expressed representations of an assembly are called its compact and expressed size. Since an assembly is expressed by concatenating the expressed forms of its modules, the position of a module, and hence of its primitives, is determined by the number and size of the modules that precede it in the assembly.

The functional modularity of a module is considered with respect to some set of assemblies, called the **context set**. We will use the operation of replacing the module at a given position within an assembly by another module. A position used in this way will be called an **insertion point**.

Using the above concepts, a definition for functional modularity in variable length search problems can now be stated. Let $\mathbb{S}$ be a context set and let $\mathbb{C}$ be a set of comparison modules.

Definition 1 (Functional Modularity). *A module A is functionally modular with respect to $\mathbb{S}$, insertion points i_S , and $\mathbb{C}$ iff:*

$$\forall S \in \mathbb{S} : \forall C \in \mathbb{C} : f(S(A, i_S)) \geq f(S(C, i_S))$$

where $S(A, i)$ specifies placing module A at the i^{th} position of S , and $f(S)$ returns the fitness of an assembly S . Functionally modular modules are also simply called functional modules.

3 Coevolution of Modules and Assemblies

We study how the development of functional modules can be achieved in a setup where modules and assemblies coevolve. A coevolutionary approach to module

formation is obtained by using two populations, one containing modules and one containing assemblies. Assemblies are candidate solutions for the problem, and hence their evaluation is given by the fitness function of the problem.

In Evolutionary Multi-Objective Optimization (EMOO [13]), individuals are evaluated on *multiple* objectives instead of a single fitness function, and the values of these objectives are treated separately; for an introduction, see e.g. [5].

The central idea in Pareto-coevolution is to view the outcomes of interactions with other evolving individuals as *objectives*. Treating the outcomes of interactions separately provides more specific information about individuals than a single value such as the average outcome, permitting better-informed methods of selection.

4 Module Evaluation: Assemblies Provide Objectives

This section shows how the Pareto-coevolution view leads to a principle for module evaluation. It will be seen that optimization of the objectives corresponds to the optimization of functional modularity. This provides a connection between Pareto-coevolution and functional modularity.

4.1 Objectives from Pareto-Coevolution

Assemblies provide situations in which modules can perform useful roles, and thereby implicitly define objectives. The maximal set of objectives that can be considered for a module therefore consists of the union of the objectives defined by some set of assemblies, e.g. a subset of the coevolving assembly population. For an individual assembly, the objective of a module in it is to contribute to the assembly's fitness by performing a useful role in it. Assemblies define a number of positions at which modules can perform a useful role. Thus, in an assembly containing n modules, each of the n positions defines an objective. For a module A , the value of the i^{th} objective of assembly S is obtained by using i as an insertion point, and considering the fitness of the assembly resulting from placing A at the insertion point: $f(S(A, i))$.

The numerical value of this objective equals the fitness of the complete assembly, and is not informative by itself. The logic behind this choice of objectives becomes clear when *comparing* the values these objectives assign to different modules. Intuitively, a module A is more valuable than another module B for a given assembly if using A instead of B has a positive effect on the overall fitness of the assembly. This is precisely what is measured when the objective values of two modules A and B are compared; A has a higher objective value than B for position i of an assembly S if $f(S(A, i)) > f(S(B, i))$. This comparison turns out positive for A if A , when replacing B at the i^{th} position of S , results in a higher overall fitness for the assembly.

Using individual assemblies as objectives for the modules they contain allows for the identification of many different specialized roles or tasks. A module can in principle be valuable even if it is only used by a small number of assemblies,

or if only some of the assemblies employing it have high fitness, while an average fitness approach would not detect the value of such a module.

Evaluation by replacing a module with other modules and comparing the overall fitness is a form of *differential fitness comparison*. This principle has been used in various forms, e.g. Cooperative Coevolution [11], COIN's Wonderful Life Utility [15], and SEAM [20].

4.2 Correspondence between Pareto-Coevolution and Functional Modularity

The previous subsection has shown how using Pareto-coevolution, assemblies can provide objectives for modules. An important question is how the resulting objectives relate to the earlier notion of functional modularity. We show that there is a direct correspondence between these.

Let A be a candidate module, chosen from a set of all possible modules $\mathbb{C}$, and let $\mathbb{S}$ be a set of assemblies. Then A is functionally modular with respect to $\mathbb{S}$, insertion points i_S , and $\mathbb{C}$ if and only if:

$$\forall S \in \mathbb{S} : \forall C \in \mathbb{C} : f(S(A, i_S)) \geq f(S(C, i_S)) \quad (1)$$

Now consider the objectives specified by the same assemblies $\mathbb{S}$ and insertion points i_S . As defined in the previous section, these are given by $f(S(A, i_S))$ for all $S \in \mathbb{S}$. A maximizes these objectives simultaneously over $\mathbb{C}$ if and only if:

$$\forall S \in \mathbb{S} : \forall C \in \mathbb{C} : f(S(A, i_S)) \geq f(S(C, i_S)) \quad (2)$$

Equation 1 and 2 are identical. Thus, a module is functionally modular for a set of assemblies and corresponding insertion points if and only if it maximizes the objectives represented by these assemblies and insertion points.

4.3 Practical Issues in Module Evaluation

For a candidate module AB , we can replace all its occurrences in assemblies by a compact representation of the candidate module $X = AB$. After doing so, each occurrence of X in an assembly defines an objective (assembly and insertion point) that is likely to be relevant in evaluating X . For efficiency reasons, we limit module evaluation to these objectives, i.e. the objectives specified by the positions where X actually occurs.

Candidate modules are identified by considering consecutive pairs of modules that occur frequently in assemblies. One of the assemblies in which such a frequent candidate module occurs is selected, and defines an objective value for the candidate module. The candidate module is evaluated by comparing its value for the objective to that of other possible candidate modules in a comparison set $\mathbb{C}$. It is only accepted as a new module if its value for the objective is equal or greater than all alternatives in $\mathbb{C}$, and strictly greater than some alternatives.

This condition ensures that no better candidate is available, and that the candidate is an improvement over alternative combinations of modules. Thus, for given $\mathbb{C}$, S , and insertion point i_S :

$$\forall C \in \mathbb{C} : f(S(A, i_S)) \geq f(S(C, i_S)) \quad \wedge \quad \exists C \in \mathbb{C} : f(S(A, i_S)) > f(S(C, i_S))$$

A possible concern is to what extent evaluation based on a single objective is sufficient. In contrast with most EMOO work, the aim here is to identify modules *maximizing* performance in one or more objectives. Thus, we are only interested in the extremes of the tradeoff front. Since modules are added incrementally rather than evolved, the objectives can at least be optimized independently. To furthermore promote modules that maximize *multiple* objectives, candidate modules are combinations that occur frequently in assemblies.

To avoid unnecessary material, a candidate module is compared to alternatives of the same or smaller expressed length. An efficiency improvement can be made by viewing the constituents A and B of a module AB as modules. We thus compare a candidate module AB to all modules A^* and $*B$, observing the length requirement, and only accept it if it obtains at least as high fitness as all of these and higher fitness than at least one of these. A final requirement is that its fitness is at least as high as when either or both constituent modules are left out; that is, the module is also compared to A , B , and $[]$.

5 The DevRep Algorithm

DevRep()

1. modules:=primitives;
2. assemblies:=generate_random_sequences(modules);
3. **while**($\neg$ stop_criterion)
4. modules := create_modules(assemblies, modules);
5. **for** i=1:interval
6. assemblies := evolve_assemblies(assemblies, modules);
7. **end**
8. **end**

Basic cycle of the DevRep algorithm.

The choices that have been made regarding module construction and evaluation lead to an algorithm that Develops a Representation for the problem as part of the search, and is therefore called *DevRep*. This method is based on earlier work presented in [3]. The population of modules is initialized to the set of primitives. The assembly population is initialized to random sequences of these modules of a given length. Next, the following loop is repeated until a stop criterion: pairs of existing modules occurring consecutively in the assemblies are

considered for consolidation into new modules, and a generation of evolving the assemblies is performed.

Create_modules does the following. Let AB be the pair of existing modules consecutively occurring most frequently in the assembly population. One assembly in which AB occurs is selected randomly. Let us write this assembly as $XABY$, where X and Y represent sequences of modules. We now consider all assemblies $XA \cdot Y$ and $X \cdot BY$ in which either A or B has been replaced by some other module, whose expressed length does not exceed that of $XABY$. Then the fitness $f(XABY)$ must be at least as high as that of all of these modified assemblies, and higher than that of at least one of the modified assemblies:

$$\begin{aligned} \forall Z : \quad & f(XABY) \geq f(XAZY) \quad \wedge \quad f(XABY) \geq f(XZBY) \\ \exists Z : \quad & f(XABY) > f(XAZY) \quad \vee \quad f(XABY) > f(XZBY) \end{aligned}$$

If this is the case, we furthermore require that the compact representation of AB contains no unnecessary elements:

$$f(XABY) > f(XAY) \quad \wedge \quad f(XABY) > f(XBY) \quad \wedge \quad f(XABY) > f(XY)$$

If these requirements are met, the new module is given a unique ID, and added to the module population. Furthermore, all occurrences of AB in the current assemblies are replaced by the new module, followed by a null module to maintain the same assembly length.³ If not, the next **max-modules-to-consider** most frequent pairs of modules are considered in order for consolidation until at most **max-modules-per-gen** new modules are found.

Evolve_assemblies is based on deterministic crowding [9]. The following cycle is repeated a number of times equal to the assembly population size. Two assemblies are selected randomly to function as parents. Offspring are produced by crossover with probability **pcross**, and by copying otherwise. The resulting assemblies are mutated at each element with probability **pmut**. Mutation replaces a module by a randomly selected element of the module population. Next, the parents are paired up with the offspring, such that the sum of the Hamming distances between the compact representations of the parent-offspring pairs is minimized. Each offspring replaces its matched parent if its fitness is equal or higher than that of its parent.

6 Test Problems

6.1 Hierarchical Test Problems

Several authors have recently studied the scalability of evolutionary algorithms [14,20,10]. As part of this, difficult hierarchical test problem were designed that yet contain structure, such as Hierarchical IF-and-only-IF (H-IFF) [18]. Preliminary experiments showed H-IFF is no longer difficult when modules can be

³ If AB occurs more than once, all occurrences are replaced.

Table 1. Target modules for the H-IFF and H-XOR problems. The target modules at each level are composed of those at the previous level, and are each other's inverse. By using XOR instead of IFF, repetition of a single type of module is no longer sufficient for solving the problem.

Level	H-IFF		H-XOR	
	A	B	A	B
4	0000000000000000	1111111111111111	0110100110010110	1001011001101001
3	00000000	11111111	01101001	10010110
2	0000	1111	0110	1001
1	00	11	01	10
0	0	1	0	1

repeated; 64-bit H-IFF was solved within a few generations, or a fraction of a second. We therefore test performance on the analogous H-XOR problem [18], which uses XOR instead of IF and only iF as its basic function, see Table 1. This problem is much more difficult for variable length methods due to its reduced potential for exploiting repetitiveness.

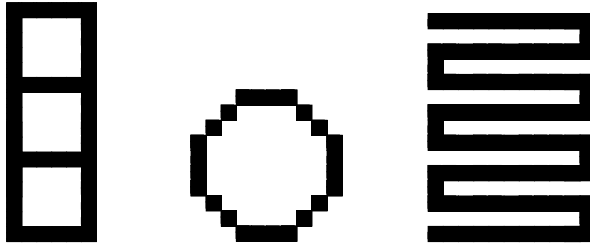


Fig. 1. Target images used in the experiments: squares, octagon, and path.

6.2 Pattern Generation

In the second test problem, the goal is to generate a picture using turtle graphics on a toroidal grid. The primitives for this problem are the following commands: TURN LEFT, TURN RIGHT, MOVE, and PUT PIXEL. The expressed form of an assembly is a sequence of these primitives. The interpretation of a sequence of primitives produces a bitmap, representing the concept specified by the assembly. The starting point for the interpretation of a sequence is the point from which the figure is drawn; thus, locating the target figure on the grid is not part of the task.

The target images are 16x16 bitmaps containing simple line drawings, see Figure 1. The two objectives are the number of black and white pixels correctly produced. Perfect solutions for these four problems require between 70 and 160 primitive operators.

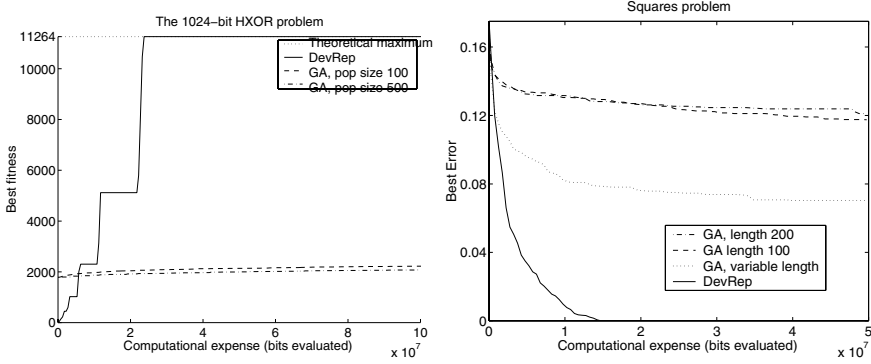


Fig. 2. Performance on the 1024-bit H-XOR problem (left) and the *squares* (right) problem.

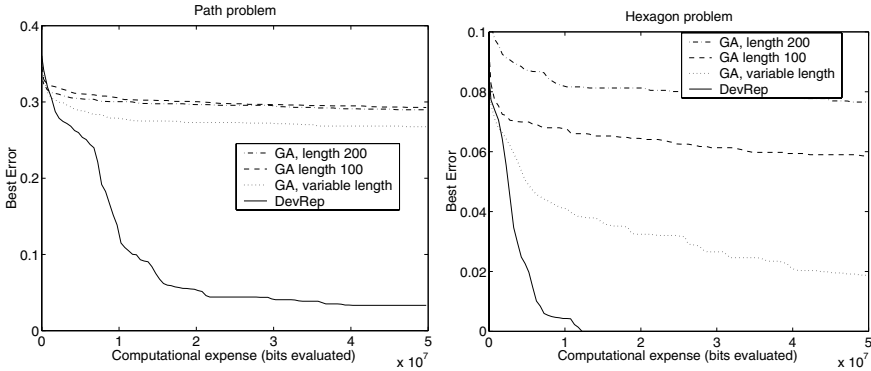


Fig. 3. Performance on the *path* (left) and *octagon* (right) problems.

7 Experimental Results

Here, we report experiments with the DevRep algorithm. Assemblies are of length 2 (H-XOR) or 10 (pattern generation). Other parameters are as follows: `max-modules-to-consider` = 5, `max-modules-per-gen` = 2, `interval` = 50, `pcross` = .9, `pmut` = .1. All curves are averaged over ten runs.

A genetic algorithm variant of DevRep is obtained simply by omitting the module formation procedure. Furthermore, while in DevRep a child replaces its parent when all of its objective values are equal or higher, for the genetic algorithm we employ the standard Pareto-dominance criterion. We employ two genetic algorithm methods using fixed length representations of 100 and 200 primitives, and a variable length method, initialized with size 10 assemblies.

The experimental results are as follows. On the 64-bit H-XOR problem (not shown), all algorithms progressed substantially, while only DevRep reached the maximum score within the given number of bit evaluations and for all runs. On the 1024-bit version of the same problem, see figure 2, the difference in scalability

between the methods becomes clear; while the genetic algorithm variants all stall at a low fitness level, DevRep is able to progress by repeatedly forming larger modules, and subsequently searching in terms of these modules. DevRep thereby again achieves maximum performance on the problem for all runs. Inspection of a run showed that the modules formed over time correspond precisely to the target modules for the problem, shown in table 1.

On the pattern generation tasks, fig. 2-3, all genetic algorithm variants performed poorly. Apparently, the biases of the genetic algorithm do not correspond well to those required for these problems, which are characterized by long range dependencies and by sequential structure. DevRep greatly improves over this performance. The average score substantially improved over the genetic algorithm methods, and while the GA methods were unable to find a correct solution for any of the problems, DevRep found perfect solutions for all problems, and in all runs except three of the 'path' problem runs.

8 Discussion

The DevRep algorithm shares several features with SEAM [20], most importantly the use of recursive module formation, leading to hierarchy, and the use of Pareto-coevolution for module evaluation; the latter distinguishes these algorithms from other methods for representation development. Compared to SEAM, the main new contribution is the application of this evaluation principle in a variable length setting. A related difference is the use of assemblies evolved on fitness, rather than random assemblies; in additional experiments, fitness based selection was found to be a necessary component.

A crucial idea in defining modularity for variable length problems was to consider modularity relative to specific subsets of all possible assemblies. This possibility is required to address non-separable problems such as H-IFF and H-XOR.

Search algorithms can be characterized by the types of patterns they are able to discover. For DevRep, these include the following:

- **Functional modularity.** By maximizing the module objectives, the algorithm searches for modules that are functionally modular, as defined in section 2.
- **Hierarchical modularity.** By looking for useful modules recursively, the algorithm searches for modules of hierarchical structure.
- **Repetitive modularity.** Modules can be used repeatedly, i.e. a combination of two consecutive primitives or operators can be used multiple times within a single individual, due to the use of position-independent coding.

9 Conclusions

Problems that require long solutions pose difficulties to standard genetic algorithms, due to the size of the associated search spaces. Still, if such problems

have *structure*, they can in principle be addressed. While several authors have investigated the simultaneous development of modules and assemblies, principled evaluation of such partial solutions has long been an open issue. Here, we have derived objectives for module evaluation in variable length problems from Pareto-coevolution.

Functional modularity is defined, and it is shown that optimization of the objectives derived from Pareto-coevolution corresponds to optimization of functional modularity. Based on this evaluation principle, the DevRep algorithm for variable length problems is developed.

DevRep was tested on Hierarchical XOR (H-XOR) up to size 1024, and on pattern generation tasks. It was found to develop large and ideal partial solutions and greatly improve performance compared to a genetic algorithm approach. DevRep is able to exploit structure in certain large search problems, in particular *functional modularity*, *hierarchical modularity*, and *repetitive modularity*. This is achieved by recursively forming modules and searching the space of combinations of such modules, thus forming modules in an open ended way.

We conclude that certain forms of structure in large search problems can be exploited by gradually consolidating learned information. Here the patterns that are detected are templates, but in principle any type of detectable pattern can be considered. According to this view, a challenge for research into large search problems is to identify the patterns present in problems of interest, and to develop corresponding algorithms exploiting those patterns.

Acknowledgements. The author wishes to thank the Agents Systems Group at the Vrije Universiteit and the members of the DEMO Lab at Brandeis, particularly Sevan Ficici, Richard Watson, and Anthony Bucci, who gave valuable feedback on this work. A TALENT-fellowship from the Netherlands Organisation for Scientific Research (NWO) is gratefully acknowledged.

References

1. Peter J. Angeline and Jordan B. Pollack. Coevolving high-level representations. In Christopher G. Langton, editor, *Artificial Life III*, volume XVII of *SFI Studies in the Sciences of Complexity*, pages 55–71, Redwood City, CA, 1994. Addison-Wesley.
2. Anthony Bucci and Jordan B. Pollack. Order-theoretic analysis of coevolution problems: Coevolutionary statics. In *Proceedings of the GECCO-2002 Workshop on Coevolution: Understanding Coevolution*, 2002.
3. Edwin D. De Jong and Tim Oates. A coevolutionary approach to representation development. In E.D. de Jong and T. Oates, editors, *Proceedings of the ICML-2002 Workshop on Development of Representations*, Sydney NSW 2052, 2002. The University of New South Wales. Online proceedings: <http://www.demon.cs.brandeis.edu/icml02ws>.
4. Edwin D. De Jong and Jordan B. Pollack. Learning the ideal evaluation function. In *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO-2003*, 2003.

5. Kalyanmoy Deb. *Multi-Objective Optimization Using Evolutionary Algorithms*. Wiley & Sons, New York, NY, 2001.
6. Sevan G. Ficici and Jordan B. Pollack. Pareto optimality in coevolutionary learning. In Jozef Kelemen, editor, *Sixth European Conference on Artificial Life*, Berlin, 2001. Springer.
7. Frederic Gruau. *Neural Network Synthesis Using Cellular Encoding and the Genetic Algorithm*. PhD thesis, PhD Thesis, Ecole Normale Supérieure de Lyon, 1994.
8. John R. Koza. *Genetic Programming II: Automatic Discovery of Reusable Programs*. The MIT Press, Cambridge, MA, May 1994.
9. Samir W. Mahfoud. *Niching Methods for Genetic Algorithms*. PhD thesis, University of Illinois at Urbana-Champaign, Urbana, IL, May 1995. IlliGAL Report 95001.
10. Martin Pelikan and David E. Goldberg. Escaping hierarchical traps with competent genetic algorithms. In L. Spector, E.D. Goodman, A. Wu, W. B. Langdon, H.-M. Voigt, M. Gen, S. Sen, M. Dorigo, S. Pezeshk, M. H. Garzon, and E. Burke, editors, *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO-2001*, pages 511–518, San Francisco, CA, 2001. Morgan Kaufmann.
11. Mitchell A. Potter and Kenneth A. De Jong. Cooperative coevolution: An architecture for evolving coadapted subcomponents. *Evolutionary Computation*, 8(1):1–29, 2000.
12. Justinian P. Rosca and Dana H. Ballard. Discovery of subroutines in genetic programming. In P.J. Angeline and K. E. Kinnear, Jr., editors, *Advances in Genetic Programming 2*, chapter 9, pages 177–202. The MIT Press, Cambridge, MA, 1996.
13. J. David Schaffer. Multiple objective optimization with vector evaluated genetic algorithms. In John J. Grefenstette, editor, *Proceedings of the First International Conference on Genetic Algorithms and their Applications*, pages 93–100, Hillsdale, NJ, 1985. Lawrence Erlbaum Associates.
14. Dirk Thierens. Scalability problems of simple genetic algorithms. *Evolutionary Computation*, 7(4):331–352, 1999.
15. Kagan Tumer and David Wolpert. Collective intelligence and Braess' paradox. In *Proceedings of the 7th Conference on Artificial Intelligence (AAAI-00) and of the 12th Conference on Innovative Applications of Artificial Intelligence (IAAI-00)*, pages 104–109, Menlo Park, CA, 2000. AAAI Press.
16. Richard A. Watson. *Compositional Evolution: Interdisciplinary Investigations in Evolvability, Modularity, and Symbiosis*. PhD thesis, Brandeis University, 2002.
17. Richard A. Watson. Modular interdependency in complex dynamical systems. In Bilotta et al., editor, *Workshop Proceedings of the 8th International Conference on the Simulation and Synthesis of Living Systems*. UNSW Australia, 2003.
18. Richard A. Watson, Gregory S. Hornby, and Jordan B. Pollack. Modeling building-block interdependency. In A.E. Eiben, Th. Bäck, M. Schoenauer, and H.-P. Schwefel, editors, *Parallel Problem Solving from Nature, PPSN-V*, volume 1498 of *LNCS*, pages 97–106, Berlin, 1998. Springer.
19. Richard A. Watson and Jordan B. Pollack. Symbiotic combination as an alternative to sexual recombination in genetic algorithms. In M. Schoenauer, K. Deb, G. Rudolph, X. Yao, E. Lutton, J. Julian Merelo, and H.-P. Schwefel, editors, *Parallel Problem Solving from Nature, PPSN-VI*, volume 1917 of *LNCS*, Berlin, 2000. Springer.
20. Richard A. Watson and Jordan B. Pollack. A computational model of symbiotic composition in evolutionary transitions. *Biosystems*, 69(2-3):187–209, May 2003. Special Issue on Evolvability, ed. Nehaniv.

Learning the Ideal Evaluation Function

Edwin D. de Jong^{*} and Jordan B. Pollack

DEMO Lab, Volen National Center for Complex Systems, Brandeis University
MS018, 415 South street, Waltham MA 02454-9110, USA
{edwin|pollack}@cs.brandeis.edu, <http://demo.cs.brandeis.edu>

Abstract. Designing an adequate fitness function requires substantial knowledge of a problem and of features that indicate progress towards a solution. Coevolution takes the human out of the loop by dynamically constructing the evaluation function based on interactions between evolving individuals. A question is to what extent such automatic evaluation can be adequate. We define the notion of an ideal evaluation function. It is shown that coevolution can in principle achieve ideal evaluation. Moreover, progress towards ideal evaluation can be measured. This observation leads to an algorithm for coevolution. The algorithm makes stable progress on several challenging abstract test problems.

Keywords: Coevolution, Pareto-Coevolution, Complete Evaluation Set, ideal evaluation, underlying objectives, Pareto-hillclimber, over-specialization

Designing an adequate fitness function requires substantial domain knowledge and can be a critical factor in evolution, see e.g. [9]. Often though, tests revealing information about the qualities of individuals can readily be performed. In chess for example, absolute evaluation of strategies is extremely difficult, while comparing individuals only requires knowledge of the rules of the game. If individuals can be evaluated based on tests, coevolution can be used to circumvent the problem of defining a fitness function.

Coevolution has already produced a number of promising results [10,19,12,17]. However, there are various ways in which evaluation in coevolution can become inaccurate [21,2,16]. As a step towards accurate evaluation, Juillé defines a domain-specific *ideal trainer* [11]. Rosin provides an *automatic* mechanism for accurate evaluation, but the approach is based on a single-objective perspective, and likely to stall for problems with multiple underlying objectives. Pareto-coevolution [6,20] uses the outcomes of a learner against coevolving evaluators (tests) as objectives in the sense of Evolutionary Multi-Objective Optimization.

By combining Rosin's complete set of tests with Ficici's important notion of *distinctions* [7], we arrive at the concept of a *Complete Evaluation Set*. The complete evaluation set was first described in [3], and detects all differences between learners relevant to selection.

^{*} Current address: DSS Group, Utrecht University. dejong@cs.uu.nl

We prove that given a complete evaluation set as evaluators, Pareto-coevolution leads to *ideal evaluation*, i.e. evaluation according to all underlying objectives of a problem. Using order theory, Bucci has defined a set of *maximally informative evaluators* [1]. While this set also makes all distinctions necessary for learner selection, it is different, as the complete evaluation set is a *maximally informative set* of evaluators. By virtue of this, the complete evaluation set has the property that its required size is bounded and small.

The complete evaluation set provides a practical way for coevolution methods to approximate ideal evaluation. An algorithm based on this principle is described, and found to achieve stable progress on a number of test problems that could not be addressed by standard coevolution methods used for comparison. This paper summarizes the results described in our technical report [3]. A more extensive account of this work is to appear in [4].

1 Evaluation in Coevolution

We consider problems where multiple objectives may underly performance. This includes as a special case single fitness value problems. The theoretical *ideal evaluation function* specifies which individuals would be preferred over which other individuals if the underlying objectives would be available. We demonstrate that using the outcomes of interactions between coevolving individuals as objectives, it is possible to construct an evaluation function that is precisely equivalent to the ideal evaluation function.

1.1 An Ideal Evaluation Function

The problem of evaluating individuals according to multiple objectives is studied in Evolutionary Multi-Objective Optimization (EMOO), see e.g. [8,5]. We follow EMOO in using the Pareto-dominance relation to compare individuals:

Definition 1 (Pareto-dominance). *An individual a dominates another individual b with respect to a set of objectives O if:*

$$\text{dom}_O(a, b) \iff \forall i : O(a, i) \geq O(b, i) \wedge \exists i : O(a, i) > O(b, i) \quad (1)$$

where $O(x, i)$ returns the value of the i^{th} objective of x , $1 \leq i \leq n$, and n is the number of objectives contained in O .

To obtain an evaluation function F_{ideal} that determines for any pair of individuals a and b whether a is to be preferred over b , we can directly employ the Pareto-dominance relation based on the (unknown) underlying objectives U :

$$F_{\text{ideal}}(a, b) = \text{dom}_U(a, b) \quad (2)$$

$$U(x, i) = x_i \quad (3)$$

In general, the solution to a multi-objective problem is a tradeoff front of individuals that achieve the different objectives to different degrees. If a single optimum exists, as in problems with scalar fitness functions, this individual is also the solution of the corresponding EMOO problem.

1.2 Coevolution: Interactions as a Basis for Evaluation

The difficulty of evaluation in coevolution is that selection does not have access to the ideal evaluation function. Instead, selection decisions must be based on the outcomes of interactions between individuals. We will demonstrate that these interactions can provide sufficient information for ideal evaluation.

We distinguish between *learners*, and *evaluators*. Learners are to address the problem at hand. The aim of the evaluators is to distinguish between learners. The set of all possible learners is denoted as $\mathbb{L}$, and the set of all possible evaluators as $\mathbb{E}$. Particular sets of learners and evaluators are denoted as L and E .

All interactions are assumed to be pairwise. An interaction is a function $G : \mathbb{L} \times \mathbb{E} \rightarrow \mathbb{O}$ that accepts a learner and an evaluator. It returns an outcome for the learner from some ordered set of values $\mathbb{O}$, e.g. real numbers or game outcomes. An interaction $G(a, e)$ may be thought of as a two-player game between a and e , or as a test or test-case that e poses to a . The interaction between a and e reveals *some* information about a 's underlying objectives, while it is unknown what this information is, or what the underlying objectives are.

Clearly, in order for the interaction function G to be useful in evaluating individuals, it must bear some relation to the underlying objectives that determine the quality of individuals. Specifically, we require that any increase in an underlying objective of an individual a must be reflected in an increased outcome of its interaction with some player b . Conversely, the information contained in G should not provide misleading information by indicating an improvement when there is none.

Formally, the *interaction requirement* specifies that for any pair of learners $a, b \in \mathbb{L}$:

$$\exists i : a_i > b_i \iff \exists e \in \mathbb{E} : G(a, e) > G(b, e) \quad (4)$$

Each learner is evaluated based on its outcomes against the current set of evaluators. Following Pareto-coevolution [6,20], these outcomes are treated as objectives. This results in the following evaluation function F_{coev} for learners:

$$F_{\text{coev}} = \text{dom}_{O_G^E}(a, b) \quad (5)$$

where $a, b \in L$ are learners, and the k^{th} objective of a learner $L^i \in L$ is the outcome of its interaction G with the k^{th} evaluator $E^k \in E$:

$$O_G^E(L^i, k) = G(L^i, E^k) \quad (6)$$

2 Principled Evaluation in Coevolution

An evaluator $e \in \mathbb{E}$ *distinguishes* between two learners $a, b \in \mathbb{L}$ if a 's outcome against e is higher than b 's outcome:

$$\text{dist}(e, a, b) \iff G(a, e) > G(b, e) \quad (7)$$

We define a *Complete Evaluation Set* to be a set of evaluators E that make all distinctions that can be made between the learners in L :

Definition 2 (Complete Evaluation Set). *An evaluation set $E \subseteq \mathbb{E}$ is complete for an interaction function G and a set of learners L if and only if:*

$$\forall a, b \in L : [\exists e \in \mathbb{E} : G(a, e) > G(b, e) \implies \exists e' \in E : G(a, e') > G(b, e')] \quad (8)$$

We will write E_L^* to denote an evaluation set that satisfies this property for a set of learners L . The theoretical result of this paper is that the use of a complete evaluation set E_L^* as objectives for a set of learners L renders the coevolutionary evaluation function equivalent to the ideal evaluation function:

Theorem 1 (Equivalence with the ideal evaluation function). *Let $F_{\text{coev}}(a, b) = \underset{O_G^{E_L^*}}{\text{dom}}(a, b)$ be a coevolutionary evaluation function for L based on a complete evaluation set E_L^* . Let $F_{\text{ideal}}(a, b) = \underset{U}{\text{dom}}(a, b)$ be the ideal evaluation function for L , based on the underlying objectives U . Furthermore, let G satisfy the interaction requirement for U . Then for any pair of learners $a, b \in L : F_{\text{coev}}(a, b) = F_{\text{ideal}}(a, b)$.*

A proof is given in appendix A. The finding implies that by treating the outcomes of learners against evaluators as objectives, ideal evaluation can in principle be achieved. Thus, it may be seen as a motivation for Pareto-Coevolution.

2.1 Approximating the Complete Evaluation Set

We now consider how algorithms may approximate the complete evaluation set. This is surprisingly tractable, since the number of potential distinctions is the square of the number of learners. Thus, we can treat all potential distinctions between learners as objectives, resulting in a setup where evaluators strive to find all possible distinctions between learners:

$$O(E^k, n_l \cdot i + j) = \begin{cases} 1 & \text{if } G(L^i, E^k) > G(L^j, E^k) \\ 0 & \text{otherwise} \end{cases} \quad (9)$$

where $O(E^k, n)$ is the n^{th} objective of an evaluator $E^k \in E$, L^i is a learner, $n_l = |L|$ is the number of learners and $G(l, e)$ is the interaction function accepting a learner and an evaluator. A convenient representation of the objectives of evaluators is as the entries in a square matrix, where the columns and rows represent the learners, and each entry represents a distinction between two learners, see figure 1 and eq. 7.

3 An Algorithm for Pareto-Coevolution

The above idea can be translated into an outline for algorithms by combining a current population of learners and a set of offspring into a single set of learners. To obtain an evaluation set for this set of learners, we invoke a secondary evolutionary process. This leads to an outline for algorithms, see figure 2.

Interaction outcomes				Resulting distinctions			
$G(L_i, E_k)$	E1	E2	E3	$\text{dist}(L_i, L_j)$	L1	L2	L3
L1	0	1	0	L1	0	1	0
L2	0	0	1	L2	1	0	1
L3	1	1	0	L3	1	1	0

$G(L2, E3) > G(L3, E3)$

Fig. 1. Matrix representation of the possible distinctions that can be made between a set of learners (example). A distinction between learners L^i and L^j can be made (1) if an evaluator E^k exists such that the outcome of L_i against E^k exceeds that of L_j .

Convergence to ideal evaluation can be guaranteed in the limit by generating every possible evaluator with non-zero probability, and collecting any evaluator making a new distinction; for n learners, this leads to a set of at most n^2 evaluators. In practice, we iterate the inner loop for a single step only, so as to balance the computational effort spent on evolving learners and evaluators.

Concerning learner selection, preliminary experiments led to the finding that non-dominance is not strict enough as a selection criterion for learners and can result in regress. Therefore, a learner may replace an existing individual only if it dominates that individual. This simple technique is sufficient when a global optimum exists. For an algorithm also striving towards a balanced distribution of individuals over the tradeoff front, see [14].

1. $L_{\text{pop}} := \text{random\_population}()$
2. $E_{\text{pop}} := \text{random\_population}()$
3. **while** $\neg$ performance-criterion
4. $L_{\text{tot}} := L_{\text{pop}} \cup \text{generate}(L_{\text{pop}})$
5. **while** $\neg$ distinctions-criterion
6. $E_{\text{tot}} := E_{\text{pop}} \cup \text{generate}(E_{\text{pop}})$
7. $\forall i, k : G[i, k] := G(L^i, E^k)$
8. $\forall k, i, j : d[k, i, j] := (G[i, k] > G[j, k])$
9. evaluate(E_{tot}, d)
10. $E_{\text{pop}} := \text{select}(E_{\text{tot}})$
11. **end**
12. evaluate(L_{tot}, G)
13. $L_{\text{pop}} := \text{select}(L_{\text{tot}})$
14. **end**

Fig. 2. Outline for coevolution algorithms that approximate the ideal evaluation function.

The strict selection consideration also applies to evaluator selection. In addition, diverse evaluators must be maintained, representing all underlying objectives. Therefore, an evaluator will be replaced by its offspring only, and only if this offspring dominates it. This is similar to the *deterministic crowding* method for diversity maintenance, see [15]. We call such individuals *Pareto-hillclimbers*; the PAES algorithm [13] is another example of a Pareto-hillclimber.

We have arrived at a setup where, given a population of learners L and a population of evaluators E , new learners are evaluated based on the evaluators in E and can replace any learner they dominate, while evaluators are Pareto-hillclimbers that use the distinctions between the learners in L as their objectives. This method will be called DELPHI, which stands for Dominance-based Evaluation of Learners on Pareto-Hillclimbing Individuals.

4 Test Problems and Experimental Setup

We will now investigate the algorithm derived from the ideal evaluation principle in experiments. The test problems employed are variants of the Numbers Game [21]. Individuals are vectors of real valued variables. The underlying objectives for the problems correspond precisely to these variables. Hence, the aim should be to maximize each of the individual's variables. However, as we aim to study coevolution, the selection mechanism may not use knowledge of the underlying objectives, but is based on the outcomes of interactions between individuals. The difficulty of the task is determined among other factors by the information the interaction function G provides about the underlying objectives of an individual.

The purpose of the test problems is to test to what extent coevolution algorithms are able to provide accurate evaluation, i.e. evaluation according to all underlying objectives. To this end, the problems should make accurate evaluation difficult. This is achieved by making it likely for evaluators to represent only a subset of the dimensions or objectives in the problem. When this occurs, learners can only progress on a subset of the underlying objectives, a phenomenon called *over-specialization* or *focusing* [21]. In this case the minimum value of learners will not increase further. By using the minimum value of individuals as a performance measure, we can detect whether progress is being made on all underlying objectives.

The first test-problem is called COMPARE-ON-ALL. In this problem, the learner and the evaluator are compared based on *all* of the evaluator's dimensions. The outcome of the interaction function for this problem is positive (1) if and only if the learner's values are all at least as high as those of its evaluator:

$$G_{\text{all}}(a, e) = \begin{cases} 1 & \text{if } \forall i : a_i \geq e_i \\ -1 & \text{otherwise} \end{cases} \quad (10)$$

where a is a learner, e is an evaluator, and x_i denotes the value of individual x in dimension i . In the COMPARE-ON-ONE problem, the learner and the evaluator are compared based on only *one* of the evaluator's dimensions, namely the

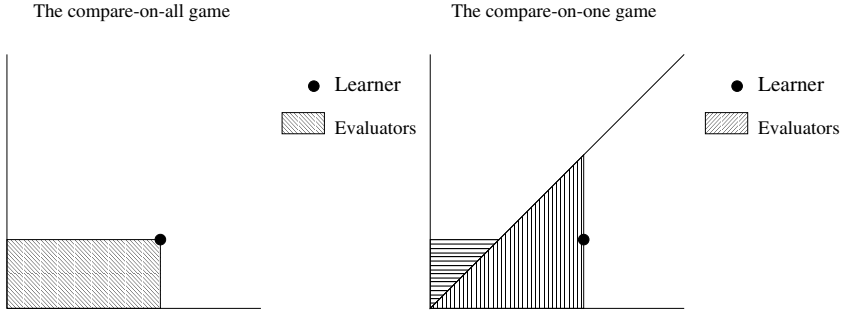


Fig. 3. The grey areas show all evaluators that are solved by the learner in the figure. Left: the COMPARE-ON-ALL game. A learner receives a positive outcome if it is equal or greater than the evaluator in every dimension. Right: the COMPARE-ON-ONE game. A learner receives a positive outcome if it is equal or greater in the evaluator’s highest dimension.

evaluator’s dimension with the highest value. The games are illustrated in figure 3.

$$m = \arg \max_i e_i \quad (11)$$

$$G_{one}(a, e) = \begin{cases} 1 & \text{if } a_m \geq e_m \\ -1 & \text{otherwise} \end{cases} \quad (12)$$

While evaluators in the compare-on-all game can compare learners based on all of their dimensions, this is not possible in the compare-on-one game. Therefore, evaluators in different regions of the space must be maintained. This results in a strong risk of maintaining evaluators for only some of the underlying objectives, as desired.

5 Experimental Results

The setup is as follows. Initial values in each dimension are chosen uniformly from $[0, 0.05]$. A new generation of individuals is created using mutation. Mutation adds a value chosen uniformly from $[-d - b, d - b]$ to a dimension i , where $d = 0.1$ is the *mutation distance* and $b = 0.05$ (where used) is the *mutation bias*. Mutation is applied to two randomly chosen dimensions. Thus, an increase in one dimension will often be accompanied by a decrease in another, and an improved interaction outcome does not imply improvement on all objectives. The size of learner and evaluator populations and of new generations is 50, resulting in learner and evaluator sets of size 100. All experiments (except the trajectory graph) are averaged over 100 runs.

We performed experiments with the COMPARE-ON-ALL and COMPARE-ON-ONE game in 2-dimensional and 5-dimensional form, with and without mutation bias. Due to space limits, we present results for the easiest and most difficult variants in the problem set: 2-dimensional COMPARE-ON-ALL without mutation

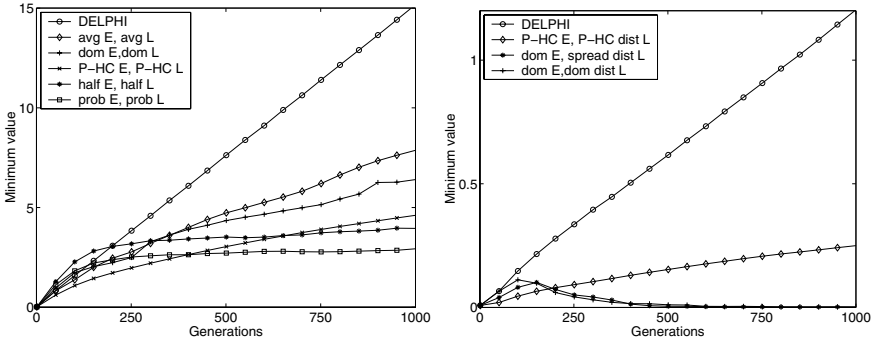


Fig. 4. Left: Performance of DELPHI and a number of competitive methods on the 2-dimensional COMPARE-ON-ALL problem. All methods achieve some progress on this problem. **Right:** DELPHI and comparison methods on 5-dimensional COMPARE-ON-ONE with mutation bias. Only the methods employing Pareto-Hillclimbing still achieve sustained progress; the other methods overspecialize, and neglect one or more objectives.

bias and 5-dimensional COMPARE-ON-ONE with mutation bias. For the latter problem, 86% of the mutations that produce an increase in some dimension cause a (typically larger) decrease in some other dimension.

We first compare DELPHI to several competitive coevolution methods. In AVG E, AVG L, the fitness of learners is the average score against evaluators, vice versa. Individuals are selected into the next population with a fitness-proportional probability. PROB E, PROB L views the outcomes as objectives, and employs a standard EMOO method [8] sorting individuals based on the number of individuals they are dominated by and using the normalized rank as the probability of selection. A stricter variant HALF E, HALF L selects the best half of the population. Still more strict is a method replacing an existing individual by any new individual that dominates it (DOM E, DOM L). Finally, we require that the replacer must be the offspring of the replacee (P-HC E, P-HC L), so that both learners and evaluators are Pareto-hillclimbers.

Figure 4 shows the average minimum value for the two-dimensional COMPARE-ON-ALL problem. All competitive methods are able to achieve some progress. DELPHI outperforms all of these, and makes remarkably constant progress.

To test whether choices made in developing DELPHI are necessary, we perform several control experiments. This time, the much more difficult COMPARE-ON-ONE problem is used with five dimensions and with mutation bias. All methods use the outcomes of interactions with evaluators as the objectives for learners, and use the distinctions between learners as objectives for evaluators. DOM E, SPREAD DIST L attempts to make evaluators spread over the possible distinctions. The fitness contribution for making a distinction is shared with other evaluators making the distinction. This *competitive fitness sharing* [18] method was the most successful of several methods used in [7] when applied to distinc-

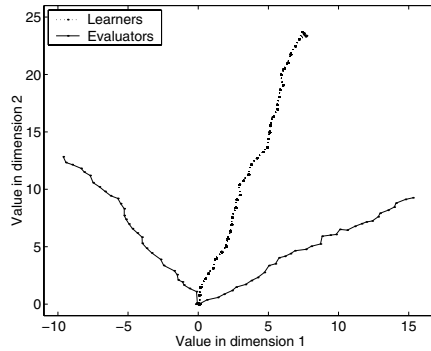


Fig. 5. Trajectories in version of the COMPARE-ON-ONE problem where the underlying objectives have been rotated 30 degrees anti-clockwise. The evaluators still identify the underlying dimensions when these do not correspond to the variables of the problem.

tions, as it is here. P-HC E, P-HC L tests whether learners may also benefit from the parent criterion; both learners and evaluators are Pareto-hillclimbers. To test if the parent criterion is necessary in evaluator selection, DOM E, DOM DIST L uses dominance for both learner and evaluator selection.

For this difficult test problem, only methods employing Pareto-Hillclimbing for the evaluation of evaluators achieve sustained progress on all objectives, see fig. 4. The comparison methods are unable to do so, and even deteriorate due to overspecialization, i.e. values are not maintained or improved for all objectives simultaneously. In summary, only DELPHI displays consistent and considerable progress across all test problems.

Finally, we investigate whether evaluators identify the underlying objectives when these have no direct correspondence to the variables of the problem. To test this, individuals in COMPARE-ON-ONE are projected onto a rotated coordinate system. The variables and operators of variation remain unchanged. As the trajectories in figure 5 show, the evaluators approximately identify the new underlying objectives of the problem, while learners progress evenly in both of the extracted underlying dimensions. Thus, the identification of the underlying objectives was not merely due to a correspondence between the variables and objectives of the problem.

6 Conclusions

Coevolution in principle offers a potential for learning in problems where no adequate evaluation function is known. We began by considering what the ideal evaluation function would be if one would have access to the underlying objectives of a problem. Since these underlying objectives are not available, actual evaluation in coevolution must be based on interactions between individuals. The theoretical result of the article is that in the limit of finding all possible distinctions, this evaluation becomes equal to the ideal evaluation function.

The result immediately suggests a practical operational criterion for *approximating* the ideal evaluation function in the form of Ficici's distinctions [7]. We have developed an algorithm based on this principle called DELPHI. The algorithm evaluates learners by using coevolving *evaluators* as objectives, while these evaluators are evaluated by using their ability to make distinctions between learners as objectives. Strict criteria for learner and evaluator selection are found to be instrumental in DELPHI's ability to achieve sustained progress.

DELPHI was found to substantially outperform comparison methods on several abstract test problems of varying difficulty. Experimental evidence was presented indicating that the evaluators identify the underlying objectives of the problem. While the current article has explored one particular algorithm, the idea of approximating the ideal evaluation function can be taken up in many different ways, and provides a principled approach to evaluation in coevolution. We therefore hope that this work may stimulate the development of new, reliable algorithms for coevolution.

Acknowledgements. The authors would like to thank Kenneth Stanley and the anonymous reviewers for helpful comments and suggestions, and the members of the DEMO-lab, in particular Anthony Bucci, Sevan Ficici and Richard Watson, for this and for discussions that were vital in the development of this work. EdJ gratefully acknowledges a TALENT fellowship from the Netherlands Organization for Scientific Research (NWO).

References

1. Anthony Bucci and Jordan B. Pollack. Order-theoretic analysis of coevolution problems: Coevolutionary statics. In *Proceedings of the GECCO-2002 Workshop on Coevolution: Understanding Coevolution*, 2002.
2. D. Cliff and G. F. Miller. Tracking the Red Queen: Measurements of adaptive progress in co-evolutionary simulations. In F. Morán, A. Moreno, J. J. Merelo, and P. Chacón, editors, *Proceedings of the Third European Conference on Artificial Life: Advances in Artificial Life*, volume 929 of *LNAI*, pages 200–218, Berlin, 1995. Springer.
3. Edwin D. De Jong and Jordan B. Pollack. Principled Evaluation in Coevolution. Technical Report CS-02-225, Brandeis University, May 31, 2002.
4. Edwin D. De Jong and Jordan B. Pollack. Ideal evaluation from coevolution. *Evolutionary Computation*, accepted for publication.
5. Kalyanmoy Deb. *Multi-Objective Optimization Using Evolutionary Algorithms*. Wiley & Sons, New York, NY, 2001.
6. Sevan G. Ficici and Jordan B. Pollack. A game-theoretic approach to the simple coevolutionary algorithm. In M. Schoenauer, K. Deb, G. Rudolph, X. Yao, E. Lut-ton, J. Julian Merelo, and H.-P. Schwefel, editors, *Parallel Problem Solving from Nature, PPSN-VI*, volume 1917 of *LNCS*, Berlin, 2000. Springer.
7. Sevan G. Ficici and Jordan B. Pollack. Pareto optimality in coevolutionary learning. In Jozef Kelemen, editor, *Sixth European Conference on Artificial Life*, Berlin, 2001. Springer.

8. Carlos M. Fonseca and Peter J. Fleming. Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization. In Stephanie Forrest, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms, ICGA-93*, pages 416–423, San Francisco, CA, 1993. Morgan Kaufmann.
9. Pablo Funes. *Evolution of Complexity in Real-World Domains*. PhD thesis, Brandeis University, Waltham, MA, 2001.
10. D. W. Hillis. Co-evolving parasites improve simulated evolution in an optimization procedure. *Physica D*, 42:228–234, 1990.
11. Hugues Juille. *Methods for Statistical Inference: Extending the Evolutionary Computation Paradigm*. PhD thesis, Brandeis University, 1999.
12. Hugues Juille and Jordan B. Pollack. Co-evolving intertwined spirals. In L.J. Fogel, P.J. Angeline, and T. Baeck, editors, *Evolutionary Programming V: Proceedings of the Fifth Annual Conference on Evolutionary Programming*, pages 461–467, Cambridge, MA, 1996. The MIT Press.
13. Joshua D. Knowles, Richard A. Watson, and David W. Corne. Reducing Local Optima in Single-Objective Problems by Multi-objectivization. In E. Zitzler, K. Deb, L. Thiele, C.A. Coello Coello, and D. Corne, editors, *First International Conference on Evolutionary Multi-Criterion Optimization*, volume 1993 of *LNCS*, pages 268–282. Springer-Verlag, 2001.
14. Marco Laumanns, Lothar Thiele, Kalyanmoy Deb, and Eckart Zitzler. Combining convergence and diversity in evolutionary multi-objective optimization. *Evolutionary Computation*, 10(3):263–282, 2002.
15. Samir W. Mahfoud. *Niching Methods for Genetic Algorithms*. PhD thesis, University of Illinois at Urbana-Champaign, Urbana, IL, May 1995. IlliGAL Report 95001.
16. Ludo Pagie and Paulien Hogeweg. Information integration and Red Queen dynamics in coevolutionary optimization. In *Proceedings of the 2000 Congress on Evolutionary Computation, CEC-00*, pages 1260–1267, Piscataway, NJ, 2000. IEEE Press.
17. Jordan B. Pollack and Alan D. Blair. Co-evolution in the successful learning of backgammon strategy. *Machine Learning*, 32(1):225–240, 1998.
18. Christopher D. Rosin. *Coevolutionary Search among Adversaries*. PhD thesis, University of California, San Diego, CA, 1997.
19. Karl Sims. Evolving 3D morphology and behavior by competition. In R. Brooks and P. Maes, editors, *Artificial Life IV*, pages 28–39, Cambridge, MA, 1994. The MIT Press.
20. Richard A. Watson and Jordan B. Pollack. Symbiotic combination as an alternative to sexual recombination in genetic algorithms. In M. Schoenauer, K. Deb, G. Rudolph, X. Yao, E. Lutton, J. Julian Merelo, and H.-P. Schwefel, editors, *Parallel Problem Solving from Nature, PPSN-VI*, volume 1917 of *LNCS*, Berlin, 2000. Springer.
21. Richard A. Watson and Jordan B. Pollack. Coevolutionary dynamics in a minimal substrate. In L. Spector, E. Goodman, A. Wu, W.B. Langdon, H.-M. Voigt, M. Gen, S. Sen, M. Dorigo, S. Pezeshk, M. Garzon, and E. Burke, editors, *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO-2001*, pages 702–709, San Francisco, CA, 2001. Morgan Kaufmann.

Appendix A: Proof of the Equivalence

Proof (Equivalence with the ideal evaluation function). To prove the equivalence theorem, we show that given the interaction requirement for G , the coevolutionary evaluation function F_{coev} equals the ideal evaluation function F_{ideal} :

$$F_{\text{coev}}(a, b) \iff F_{\text{ideal}}(a, b) \quad (13)$$

$$\text{dom}(a, b) \iff \text{dom}(a, b) \text{ (by (5) and (2))} \quad (14)$$

$$O_G^{E_L^*} [\forall e \in E_L^* : G(a, e) \geq G(b, e) \wedge \exists e \in E_L^* : G(a, e) > G(b, e)] \quad (15)$$

$$\iff [\forall i : a_i \geq b_i \wedge \exists i : a_i > b_i] \text{ (by (6) and (3))} \quad (16)$$

$$\text{Assume: } \forall e \in E_L^* : G(a, e) \geq G(b, e) \wedge \exists e \in E_L^* : G(a, e) > G(b, e) \quad (17)$$

$$\text{Assume: } \exists i : b_i > a_i \quad (18)$$

$$\Rightarrow \exists e \in \mathbb{E} : G(b, e) > G(a, e) \text{ (by (4))} \quad (19)$$

$$\Rightarrow \exists e \in E_L^* : G(b, e) > G(a, e) \text{ (by (8))} \quad (20)$$

$$\text{This contradicts (17). Therefore (18) cannot hold, so:} \quad (21)$$

$$\nexists i : b_i > a_i \quad (22)$$

$$\Rightarrow \forall i : a_i \geq b_i \quad (23)$$

$$\text{Furthermore: } \exists i : a_i > b_i \text{ (by (17, right) and (4))} \quad (24)$$

Combining (23) and (24) proves the implication. To show the reverse implication:

$$\text{Assume: } \forall i : a_i \geq b_i \wedge \exists i : a_i > b_i \quad (25)$$

$$\text{Assume: } \exists e \in \mathbb{E} : G(b, e) > G(a, e) \quad (26)$$

$$\exists i : b_i > a_i \text{ (by (4))} \quad (27)$$

$$\text{This contradicts (25). Therefore (26) cannot hold, so:} \quad (28)$$

$$\nexists e \in \mathbb{E} : G(b, e) > G(a, e) \quad (29)$$

$$\Rightarrow \forall e \in \mathbb{E} : G(a, e) \geq G(b, e) \quad (30)$$

$$\text{And since } E_L^* \text{ is a subset of } \mathbb{E}: \quad (31)$$

$$\Rightarrow \forall e \in E_L^* : G(a, e) \geq G(b, e) \quad (32)$$

$$\exists e \in \mathbb{E} : G(a, e) > G(b, e) \text{ (by (25, right) and (4))} \quad (33)$$

$$\exists e \in E_L^* : G(a, e) > G(b, e) \text{ (by (33) and (8))} \quad (34)$$

Combining (32) and (34) proves the reverse implication, and completes the proof. ■

A Game-Theoretic Memory Mechanism for Coevolution

Sevan G. Ficici and Jordan B. Pollack

Department of Computer Science
Brandeis University
Waltham Massachusetts 02454 USA
www.demo.cs.brandeis.edu

Abstract. One problem associated with coevolutionary algorithms is that of *forgetting*, where one or more previously acquired traits are lost only to be needed later. We introduce a new coevolutionary memory mechanism to help prevent forgetting that is built upon game-theoretic principles, specifically Nash equilibrium. This “Nash memory” mechanism has the following properties: 1) It accumulates a collection of salient traits discovered by search, and represents this collection as a *mixed strategy*. 2) This mixed strategy monotonically approaches the quality of a Nash equilibrium strategy as search progresses, thus acting as a “ratchet” mechanism. 3) The memory naturally embodies the result (solution) obtained by the coevolutionary process. 4) The memory appropriately handles intransitive cycles (subject to resource limitations). We demonstrate our Nash memory using Watson and Pollack’s intransitive numbers game, and compare its performance to the conventional “Hall of Fame” memory and the more recently proposed Dominance Tournament.

1 Introduction

Among the problems one often confronts when using a coevolutionary algorithm is that of *forgetting*, where one or more previously acquired traits (i.e., components of behavior) are lost only to be needed later, and so must be re-learned. Since selection pressure determines which traits have the opportunity to reproduce, the disappearance of a trait has but a few explanations. First, a trait is selected *against* when individuals with that trait are less fit, on average, than individuals without it. Second, a trait is subject to *drift* when individuals with that trait are equally fit, on average, as individuals without it. This drift may occur in either of two ways (or both): Due to sampling error in the population dynamics (where fewer of the individuals with the trait actually reproduce, thus reducing their numbers and increasing the risk of extinction for that trait), and due to variational biases that cause the trait to be lost in the generation of offspring. (Even if a trait is selected for, the variational process may be strongly biased against it, thus causing offspring to lack the trait; such variational bias is the impetus behind the technique of *elitism* that is often used in evolutionary algorithms [6].) The disappearance of a trait becomes an instance of forgetting

if the discarded trait is subsequently able to contribute *positively* to the fitness of an individual lacking that trait.

Such variability in a trait's contribution to fitness is possible in coevolution because the observed quality of an individual (and the traits it carries) may be highly contingent upon the context of its evaluation—the population of coevolving individuals. Thus, at one moment in evolutionary time, a trait may be highly undesirable (or of neutral worth but difficult to maintain) and purged, only to become of value at some later point in time. Simple zero-sum games with intransitivities illustrate how contingency can cause a coevolutionary system to learn, forget, and re-learn traits in a cyclic fashion; the familiar Rock-Paper-Scissors game is the canonical example of an intransitive cycle (Rock beats Scissors, which beats Paper, which in turn beats Rock), where one strategy exploits the next, leading to a perpetual alternation of selective pressure for and then against each strategy in turn.

The problem of designing a heuristic “memory mechanism” to prevent forgetting intrinsically entails the problem of deciding what it is that we are trying to “remember”—what is our *solution concept*? That is, what collection of traits constitutes the desired or “correct” set, and what properties does this collection have? A principled answer to this question is imperative when a domain forces mutual exclusivity between certain traits, or when an evolutionary representation (genome) cannot simultaneously encode all desired traits. Once we have our solution concept, what organizing principle do we use in the memory mechanism to obtain it? Thus, we view a memory mechanism as an accumulator of traits; a trait enters and remains in the memory only if it is “worth” remembering, according to our organizing principle.

In this paper, we obtain our solution concept—Nash equilibrium—from game theory, and use game theory to construct an organizing principle for a new memory mechanism for coevolution. Using Watson and Pollack's *intransitive numbers game* [13], which is rife with intransitive cycles, we demonstrate the ability of our “Nash memory” mechanism to approximate a monotonic and asymptotic convergence to the Nash equilibrium of the game. Further, we show that the commonly used “best of generation” memory mechanisms (e.g., Rosin and Belew's *Hall of Fame* [9]), as well as Stanley and Mäikkiläinen's more recently proposed *Dominance Tournament* [11] mechanism, do not generally accumulate a collection of traits that corresponds to Nash equilibrium.

The paper is organized as follows. Section 2 reviews key concepts from game theory. Section 3 continues with additional concepts such as *security* and *domination*. Section 4 then reviews the literature on methods to detect and prevent forgetting. Section 5 details the construction and operation of our Nash memory mechanism. Section 6 defines and discusses the properties of the intransitive numbers game. Section 7 presents our experimental setup and results. Section 8 offers concluding remarks.

2 Game Theory Fundamentals

Here we present some fundamental points on game theory [5]. Due to limited space, we must restrict our discussion to symmetric zero-sum games (in strategic form) for two players; nevertheless, our Nash memory mechanism is easily generalized to asymmetric constant-sum games for two players.

Being symmetric, the game $\mathbf{G}$ defines a single set of *pure strategies*, $\mathcal{S}$, that is made available to both players. A player may adopt a pure strategy $s \in \mathcal{S}$ or may instead play a *mixed strategy*, which is specified by a probability distribution over $\mathcal{S}$. Pure strategies played with non-zero probability by a mixed strategy m are *in support* of (also known as “carriers” of) m . The function $C(m)$ returns the set of pure strategies that support the mixture m ; hence, $C(m) = \{s \in \mathcal{S} : \Pr(s|m) > 0\}$. A pure strategy can be understood as a degenerate mixture.

For any pair of strategy choices made by the two players, the game $\mathbf{G}$ specifies the expected payoff earned by each player; by convention, $E(\alpha, \beta)$ denotes the expected payoff earned by strategy α when played against β . Since the game $\mathbf{G}$ is zero-sum, $E(\alpha, \beta) + E(\beta, \alpha) = 0$. This implies that $E(\alpha, \alpha) = 0$.

Any pure or mixed strategy s^* that is its own “best response” is a *Nash equilibrium* strategy. More precisely, s^* is a Nash iff $\forall$ mixtures $m : E(s^*, s^*) \geq E(m, s^*)$. That is, if one player plays s^* , then the highest payoff obtainable by the other player is received by also playing s^* . Thus, a Nash strategy guarantees that $\forall$ mixtures $m : E(s^*, m) \geq 0$. Due to this property, Nash strategies provide the maximal *security level* that can be obtained in a zero-sum game—no other solution concept can guarantee a higher expected payoff without regard to the opponent’s strategy choice. This security level is also known as the *value* of the game. Finally, all games with finite $\mathcal{S}$ have at least one Nash equilibrium.

3 Additional Concepts

The *security set* of a mixture m is the set of pure strategies against which m earns an expected payoff greater than or equal to zero; that is, $S(m) = \{s \in \mathcal{S} : E(m, s) \geq 0\}$. If the security set of a mixture contains the mixture’s support set, then the mixture is *support-secure*; mixture m is support-secure iff $S(m) \supseteq C(m)$. All pure strategies are trivially support-secure. For any pure or mixed Nash strategy s^* , $S(s^*) \supseteq C(s^*)$ and $S(s^*) = \mathcal{S}$. The *vulnerability set* of a mixture m is the complement (with respect to $\mathcal{S}$) of the security set: $V(m) = \overline{S(m)}$.

One strategy α *dominates* another strategy β iff $\forall s \in \mathcal{S} : E(\alpha, s) \geq E(\beta, s)$ and $\exists s \in \mathcal{S} : E(\alpha, s) > E(\beta, s)$. By definition, for any strategy s and any Nash strategy s^* , $E(s^*, s) \geq 0$; therefore, any strategy that dominates a Nash must be Nash, as well. At the same time, Nash strategies need not dominate all (or any) non-Nash strategies. We examine this point further, below.

4 Memory Methods and Solution Concepts

Research to prevent forgetting includes a number of *memory mechanisms* that maintain a collection of “good” individuals (according to some organizing prin-

ciple) discovered over evolutionary time; the memory thus encapsulates a wider range of phenotypes than is typically found in the coevolving population at any one time. The additional phenotypic variety afforded by the memory is used to augment the evaluation process, broaden selection pressure, and thereby reduce the likelihood of forgetting. (Note that memory mechanisms are often identical to methods used to detect forgetting [2, 7, 11]. Some domains, however, permit the use of an external objective metric of behavior [4], or allow easy comparison to a known optimal solution [8], as alternative ways to detect forgetting.)

Almost all memory mechanisms in the literature are instances of a general “best of generation” (BOG) model where 1) the most fit individual in each of the m most recent generations is retained by the memory mechanism and 2) l of the m ($l \leq m$) retained individuals are sampled without replacement for use in testing individuals in the current generation. Examples of this approach are found in Sims [10] ($m = 1, l = 1$), Rosin and Belew [9] (e.g., $m = \infty, l = 20$), and Nolfi and Floreano [7] ($m = 10, l = 10$).

In a contrasting approach, Stanley and Miikkulainen [11] propose that their *dominance tournament* (DT)—a method intended to monitor coevolutionary progress—can be adapted for use as a memory mechanism. The principle is to retain the most fit individual of the current generation only if it beats all the individuals previously retained by the memory; that is, the “generation champion” must dominate the contents of the memory to be included. (Note, however, that domination is determined only with respect to the memory’s contents.) Thus, no intransitive cycles can exist amongst the strategies in the memory.

As we state above, deciding how to organize a memory entails deciding what kind of solution we wish to obtain. If we view the memory as a repository of the most salient traits discovered so far, then the logical extension of this view implies that we should probe the *memory* for the result (solution) obtained through coevolution, not the coevolving population. In the standard coevolutionary algorithm, the coevolving population is expected both to perform search *and* represent the result of the coevolution. But, there is no reason to believe that these two tasks are orthogonal, let alone mutually supportive; indeed, there is evidence that these tasks may interfere with each other [3].

A well-designed memory mechanism relieves the coevolving population from the burden (or, at least, the requirement) of representing the solution, allowing the population to concentrate on search (to improve the solution represented by the memory). The memory mechanism we introduce in this paper uses game theory, particularly Nash equilibrium, as an organizing principle. A important feature of the Nash concept is that it allows a solution to be a *collective* of strategies. The appeal of this property is highlighted by results of Nolfi and Floreano [7]. In the presence of intransitivity, they emphasize that the real solution they obtain through coevolution is not a single, objectively best champion strategy—none exists to find—but, rather a set of locations in genotype space that are optimally poised for easy transformation into alternative strategies that have been effective in evolutionary history. That is, when a population cannot simultaneously and stably represent all the strategies in certain intransitive cycles, the

best it can do is optimally traverse the cycle. This lack of a best pure strategy, coupled with a dynamic that focuses on a particular set of pure strategies, is highly suggestive of the *mixtures* that the Nash solution concept supports.

5 Construction and Operation of Nash Memory

5.1 General Framework and Instantiation

The Nash memory mechanism we examine in this paper is a particular instantiation of a general framework, which consists of two mutually exclusive sets of pure strategies, $\mathcal{N}$ and $\mathcal{M}$. In addition to the memory mechanism, we assume the existence of some external search heuristic $\mathcal{H}$. The set $\mathcal{N}$ is unbounded in size and defined to be the support set for a mixture that is secure at least against the elements of $\mathcal{N}$ and $\mathcal{M}$; that is, $S(\mathcal{N}) \supseteq \mathcal{N} \cup \mathcal{M}$. (We use $\mathcal{N}$ to denote either the mixture's support set or the mixed strategy itself, as convenient.) The objective of $\mathcal{N}$ is to represent a mixed strategy that is optimal (secure) with respect to what the search heuristic has discovered thus far; we wish the protection afforded by $\mathcal{N}$ (ideally) to increase monotonically as search progresses, thereby forming a better and better approximation of a Nash strategy for the game $\mathbf{G}$. The purpose of set $\mathcal{M}$ is to act as an accumulator or memory; it contains pure strategies that are not currently useful for $\mathcal{N}$ but were in the past and may be again in the future. The capacity of $\mathcal{M}$, however, is finite. In the present realization of the abstract memory model, we define $\mathcal{M}$ to be simply an unordered set of pure strategies with a cardinality no greater than some specified value c , which gives the *capacity* of the memory. We can imagine that c can vary over evolutionary time, in some adaptive fashion, but we instead assume a fixed value.

5.2 Initialization and First Update

We initialize $\mathcal{N}$ and $\mathcal{M}$ to be the empty set. Let $\mathcal{Q}$ be a set of strategies delivered by the search heuristic to the memory mechanism; we assume $|\mathcal{Q}| < c$. The first set $\mathcal{Q}$ to arrive updates $\mathcal{N}$ such that $C(\mathcal{N}) \subseteq \mathcal{Q}$ and $S(\mathcal{N}) \supseteq \mathcal{Q}$; the set $\mathcal{M}$ is updated to contain those elements of $\mathcal{Q}$ not in support of $\mathcal{N}$. For subsequent values of $\mathcal{Q}$, we must test the elements of $\mathcal{Q}$ against $\mathcal{N}$ to see if $\mathcal{N}$ and $\mathcal{M}$ require updating.

5.3 Testing $\mathcal{N}$

To verify that a mixed strategy is Nash, one need only check that the mixed strategy is secure against all pure strategies; testing against all possible mixtures (of which there is an uncountable infinity) is not required [12]. Consequently, if $E(q, \mathcal{N}) > 0$ for any $q \in \mathcal{Q}$, then $\mathcal{N}$ is demonstrably not a Nash strategy and we attempt to improve our approximation; otherwise, we leave $\mathcal{N}$ and $\mathcal{M}$ undisturbed and wait for the search heuristic to deliver a new set of strategies. We compute the value $E(q, \mathcal{N})$ by playing q against each strategy in support of $\mathcal{N}$ and taking a weighted average of outcomes, the weights being the probability distribution for the mixture: $E(q, \mathcal{N}) = \sum_{n \in C(\mathcal{N})} \Pr(n|\mathcal{N})E(q, n)$.

5.4 Updating $\mathcal{N}$ and $\mathcal{M}$

We define the set $\mathcal{W} = \{q \in \mathcal{Q} : E(q, \mathcal{N}) > 0\}$, that is, the “winners” from $\mathcal{Q}$. Given the pre-update values of $\mathcal{N}$ and $\mathcal{M}$, we define the post-update value $\mathcal{N}'$ such that $C(\mathcal{N}') \subseteq (\mathcal{W} \cup \mathcal{N} \cup \mathcal{M})$ and $S(\mathcal{N}') \supseteq (\mathcal{W} \cup \mathcal{N} \cup \mathcal{M})$; we obtain the value of $\mathcal{N}'$ with *linear programming*, which is the standard method for solving zero-sum games and for which polynomial-time algorithms exist [12]. Note that $S(\mathcal{N}')$ is not necessarily a superset of $S(\mathcal{N})$.

As Figure 1 illustrates, the post-update value $\mathcal{M}'$ contains zero or more items from each of three sources; some strategies in $\mathcal{W}$ may not be required, some strategies in $\mathcal{N}$ may be released, and some strategies may be retained from $\mathcal{M}$ (while others may be recalled from $\mathcal{M}$ to $\mathcal{N}'$). If the resultant $\mathcal{M}'$ has a cardinality $|\mathcal{M}'| > c$, then we discard items from $\mathcal{M}'$ until the capacity constraint is met. We can imagine a number of policies for removing items from $\mathcal{M}'$; we currently remove items at random first from those retained from $\mathcal{M}$, then those released from $\mathcal{N}$ (there is never the need to remove those from $\mathcal{W}$).

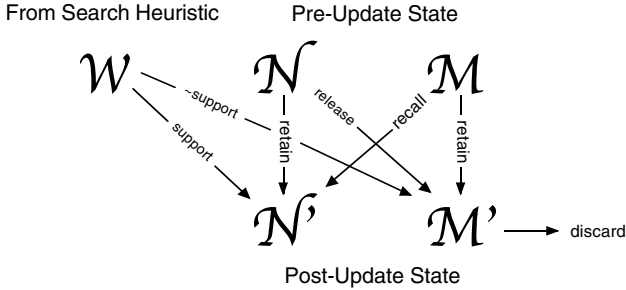


Fig. 1. Schematic of update step.

6 Intransitive Numbers Game

Watson and Pollack’s *intransitive numbers game* [13] is a coevolutionary domain that is permeated by intransitive cycles. The game’s geometric nature allows easy visualization of coevolutionary dynamics; we will use it to illustrate the operation of our Nash memory mechanism.

6.1 Definition

Each pure strategy of the intransitive numbers game is an n -dimensional vector, or point in n -dimensional space. For a pair of strategies α and β , the winning strategy is the one with higher magnitude in the dimension of least difference between the two strategies. More precisely, for two strategies α and β , the expected outcome $E(\alpha, \beta)$ is:

$$E(\alpha, \beta) = \begin{cases} 0 & \text{if } \min(h) = \infty \\ \text{sign}(\sum_{i=1}^n g_i) & \text{otherwise} \end{cases} \quad (1)$$

$$g_i = \begin{cases} \alpha_i - \beta_i & \text{if } h_i = \min(h) \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

$$h_i = \begin{cases} |\alpha_i - \beta_i| & \text{if } |\alpha_i - \beta_i| \geq \epsilon \\ \infty & \text{otherwise} \end{cases} \quad (3)$$

where ϵ is the smallest magnitude difference we wish to consider significant.

Equation 1 eliminates two asymmetries that exist in the original definition found in [13]: First, the game is now formally symmetric; second, no single dimension is the arbiter when the deltas in multiple dimensions are minimal. Note also that, while we use the term “expected outcome,” the game is deterministic.

While Equation 1 defines the outcome between two specified strategies, it does not define the universe of strategies we are to consider, leaving the game’s definition incomplete. Clearly, the strategy space may be finite or infinite, countable or uncountable; indeed, the intransitive numbers game can provide an impoverished form of open-endedness, if desired. For our purposes, we define the strategy space to be n -dimensional vectors of natural numbers, where each dimension spans the interval $[0, k]$; this yields $(k + 1)^n$ distinct pure strategies.

6.2 Game Properties

The game defined above has a single Nash equilibrium strategy for all $\epsilon < k$ ($\epsilon = k$ implies that all strategies tie each other, making them all Nash strategies—a particularly uninteresting game). The Nash strategy is the pure strategy with

value k in all n dimensions, i.e., $\langle \overbrace{k, \dots, k}^n \rangle$. Of course, other definitions of the strategy space may yield multiple Nash as well as mixed Nash strategies.

As with all Nash equilibria in symmetric zero-sum games, the Nash strategy of our game does not lose to any other strategy. With $\epsilon = 1$, the Nash ties only itself and beats all others; thus, the Nash also dominates all other strategies. Here, the solution concepts of Nash equilibrium and domination agree.

With $\epsilon = 0$, however, the solution concepts point to very different outcomes. In this case, the Nash additionally ties all strategies that have the value k in any dimension. (The strategies that tie the Nash are those on the surface of the n -dimensional space. Thus, as n grows, the percentage of tying strategies asymptotically approaches 100%, even though the number of losing (i.e., interior) strategies grows exponentially. Nevertheless, for $k = 100$ and $n = 10$, the percentage of tying strategies is still $< 10\%$. Thus, concern over the asymptotic growth of tying strategies must keep in mind the values of k and n .)

Thus, setting $\epsilon = 0$ introduces additional ways in which a tie may occur between strategies. In particular, the Nash now ties strategies that others can beat; for example, in two dimensions, $\langle k, k \rangle$ (the Nash) ties both $\langle k, k - 1 \rangle$ and $\langle k - 2, k \rangle$, yet $\langle k - 2, k \rangle$ beats $\langle k, k - 1 \rangle$. *Therefore, for a non-Nash strategy*

to transform itself into the Nash, it may be required to discard certain skills—the ability to beat certain strategies; the goal of security is not always served by mere accumulation of prowess against opponents. Due to these additional ties, virtually no strategy is dominated; those that are dominated are done so by the Nash. With $n = 2$, only four strategies are dominated, regardless of the value of k : $\langle 0, 0 \rangle$, $\langle k - 1, k \rangle$, $\langle k, k - 1 \rangle$, and $\langle k - 1, k - 1 \rangle$. With $n > 2$, the number of dominated strategies is only one (strategy $\langle 0, 0, \dots, 0 \rangle$). Finally, with either value of epsilon, the Nash strategy is the pure strategy with the most wins (though, this is not a general feature of Nash equilibrium strategies).

7 Experiments

Rather than simply compare the performance of a coevolutionary algorithm with and without the aid of our memory mechanism, we wish instead to probe the ability of the memory mechanism to discover mixed strategies that provide greater and greater security. To accomplish this, we use $\mathcal{N}$ as a quasi-static evaluation function against which we repeatedly evolve a population of strategies.

7.1 Methods

The following procedure essentially transforms a *co*-evolutionary domain into an evolutionary domain. We use the game defined in Section 6.1 with $\epsilon = 0$, $k = 100$, and $n = 2$. We represent an individual pure strategy for this game with a bit-string b of length 200, such that the expressed strategy is $\langle \sum_{i=0}^{99} b_i, \sum_{i=100}^{199} b_i \rangle$. The only variation operator we apply is bit-wise mutation with a per-bit mutation probability of 0.01; recombinational operators are not used. We construct a random strategy by setting each of the 200 bits to one with a probability of 0.5; this yields an expected strategy of $\langle 50, 50 \rangle$. The memory’s capacity is $c = 100$.

The evolution proceeds in *epochs*, as follows: 0) Initialize memory and update with a random strategy; 1) Initialize population (of size 100) with random strategies; 2) Evolve population against $\mathcal{N}$ for 30 generations (one epoch); 3) If highest-scoring individual $\hat{s}$ in population beats $\mathcal{N}$, then update memory with $\hat{s}$; 4) Goto Step 1. Each generation in Step 2 proceeds as follows: a) Evaluate each individual (pure strategy) s in the population against $\mathcal{N}$ (the score w_i obtained by each individual i is in the range $[-1, 1]$); b) The fitness of individual i is $f_i = w_i - \min(w) + 0.1$ (all fitness values are > 0); c) Copy the 10 most-fit individuals to next generation (elitism); d) Fill the remaining 90 positions with offspring using “fitness-proportionate” selection and asexual reproduction.

7.2 Results: Nash Memory

Figure 2 (Left) shows the mean and median scores obtained by the most fit individual $\hat{s}$ at the end of each epoch over 52 trials of our experiment. At the beginning of the experiment, the evolution step (Step 2, above) easily finds pure strategies that obtain the maximal score of 1.0 when played against the Nash

memory's mixture $\mathcal{N}$. Over the next 50 epochs, however, the ability of evolution to discover pure strategies that score well against $\mathcal{N}$ is gradually neutralized as the memory mechanism integrates knowledge gained from previous epochs. Indeed, over subsequent epochs, the median score obtained by evolution asymptotically approaches zero, which is the *value* of the intransitive numbers game (and zero-sum games, in general—see Section 2, above). The mean score, however, actually becomes negative; thus, the distribution is not normal. This indicates that $\mathcal{N}$ tends to be somewhat superior to the strategies that evolution is able to discover. Over the course of an experiment, the number of pure strategies in support of $\mathcal{N}$ tends to grow. The mean size of $C(\mathcal{N})$ at Epoch 500 is about 45, though the distribution is not normal (median is 50).

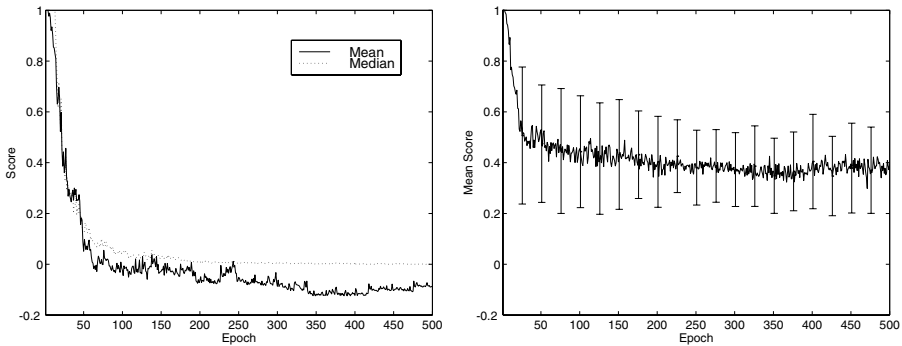


Fig. 2. Performance of evolution over time against (Left) Nash memory, over 52 trials, and (Right) BOG memory ($m = \infty$, $l = 100$), over 54 trials.

Figure 3 shows the behavior of a typical trial. (For convenience, $\mathcal{N}^i$ will denote the value of $\mathcal{N}$ at the beginning of the i -th epoch, similarly for $\mathcal{M}^i$.) The memory's strategy $\mathcal{N}^1$ is initialized to the pure strategy $\langle 53, 41 \rangle$ (indicated by the square in Figure 3, Right). The evolution step (Step 2, above) easily finds a pure strategy $\langle 55, 47 \rangle$ that beats $\mathcal{N}^1$. Accordingly, we update the memory such that $\mathcal{N}^2$ is $\langle 55, 47 \rangle$, and move $\langle 53, 41 \rangle$ to the memory set $\mathcal{M}^2$.

In Epoch 2, the evolution step discovers the pure strategy $\langle 48, 52 \rangle$, which beats $\mathcal{N}^2$. Since $\mathcal{M}^2$ contains something, we must evaluate the performance of $\langle 48, 52 \rangle$ against the contents of $\mathcal{M}^2$ before we can determine $\mathcal{N}^3$. Though $\langle 48, 52 \rangle$ beats $\langle 55, 47 \rangle$, we find that it loses to $\langle 53, 41 \rangle$ (the strategy in $\mathcal{M}^2$). Thus, we find an intransitive cycle; indeed, this intransitivity is symmetric (as in the Rock-Scissors-Paper game), so no one pure strategy can be argued to be any better than another (in any sense). This is not to say that the three strategies are interchangeable, however, for each one has unique strengths and weaknesses with respect to the other two. Since none of these pure strategies has an empty vulnerability set (see Section 3, above), $\mathcal{N}^3$ must be a mixture of some kind. The solution (found with linear programming) happens to be a mixed strategy

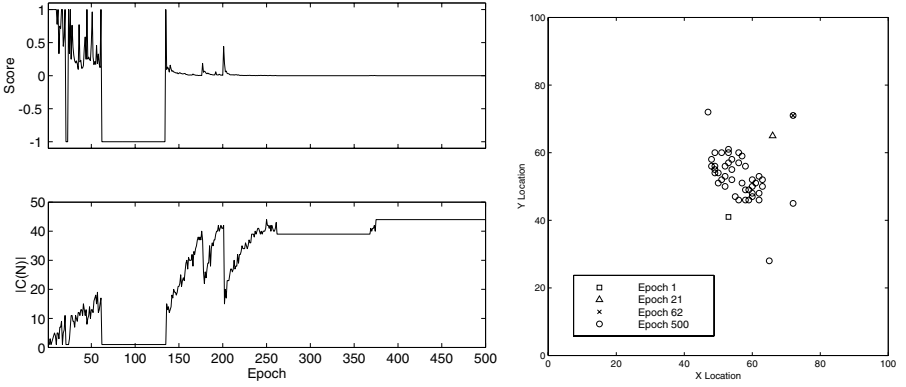


Fig. 3. Example run of evolution against Nash memory over time. Left: Upper graph gives maximal score achieved by evolution against $\mathcal{N}$ at the end of each epoch; lower graph shows size of support set $C(\mathcal{N})$ at the beginning of each epoch. Right: Pure strategies in support of $\mathcal{N}$ at the beginning of Epochs 1, 21, 62, and 500.

where each of the pure strategies is played with equal probability; $\mathcal{N}^3$ has all three strategies in support and $\mathcal{M}^3$ is empty.

By the time we arrive at $\mathcal{N}^{20}$, eleven strategies are in support. In this epoch, evolution discovers pure strategy $\langle 66, 65 \rangle$ (triangle in Figure 3, Right), which beats each of the eleven support strategies. Further, because $\mathcal{N}^{21} = \{\langle 66, 65 \rangle\}$, we can infer that $\langle 66, 65 \rangle$ is also secure against the contents of $\mathcal{M}^{20}$. The strategy $\mathcal{N}^{21}$ is sufficiently good that the evolution step obtains the worst possible score (-1) over the next few epochs. Later, we encounter a similar, but more pronounced, situation where $\mathcal{N}^{62}$ is the pure strategy $\langle 72, 71 \rangle$ ('x' in Figure 3, Right). This strategy remains until Epoch 136, where it is replaced by a mixture. The mixture $\mathcal{N}^{500}$ (circles in Figure 3, Right) retains $\langle 72, 71 \rangle$ in support.

7.3 Results: Best of Generation and Dominance Tournament

We also run our experiment using the best-of-generation memory mechanism with $m = \infty$, $l = 100$ (see Section 4, above). During evaluation, an evolving individual's score against the memory is the average score obtained against the strategies sampled from the memory; the highest-scoring individual at the end of an epoch is added to the memory if its score is > 0 . Figure 2 (Right) shows the mean scores (with standard deviation) obtained by the most fit individual $\hat{s}$ at the end of each epoch over 54 trials of this experiment; these data are normally distributed. The mean score obtained by $\hat{s}$ between epochs 250 and 500 is ≈ 0.375 ($\sigma \approx 0.160$). In other experiments where $m = 100$, $l = 100$ (not shown), the expected score of $\hat{s}$ at steady-state is ≈ 0.758 ($\sigma \approx 0.148$). Thus, the organizing principle behind BOG provides considerably less effective learning.

We next use the Dominance Tournament as our memory mechanism. The score given to an evolving individual is the number of consecutive pure strate-

gies, from the most to least recently placed in the memory, that the individual beats. At the end of an epoch, if any individuals exist that beat the entire contents of the memory, we randomly select one and place it in the memory. The DT method quickly converges onto a pure strategy that cannot be dominated by any other *with respect to the memory's contents*. Because we only consider those strategies in the memory to determine dominance, and not the entire universe of strategies, false positives may occur; indeed, all but four strategies in our game are actually *non-dominated*. Thus, the strategy onto which we converge is highly path-dependent; particularly, we may converge onto the dominated strategy $\langle 100, 99 \rangle$ (or $\langle 99, 100 \rangle$ —see Section 6), after which the memory will reject the Nash $\langle 100, 100 \rangle$, which dominates $\langle 100, 99 \rangle$ but does not beat it. Further, of all the pure strategies onto which we may converge, only the Nash equilibrium strategy has an empty vulnerability set; all the others are beatable.

7.4 Results: Bootstrapping with the Nash Memory

The results we report in Section 7.2 only show that the Nash memory is able to learn a mixed strategy $\mathcal{N}$ that is secure against what the evolution step is likely to discover, given the limitations we place on the evolution, such as the random initial population. In particular, the mixed strategies $\mathcal{N}$ obtained above are *not* the Nash equilibrium for the numbers game.

Therefore, we create a new outer loop around the method described in Section 7.1 to explore the ability of the memory to constructively contribute genetic material to the search process. When the evolution step is unable to score higher than 0.04 against $\mathcal{N}$ for 50 consecutive epochs, we change the procedure used to initialize the population: A snapshot of $\mathcal{N}$ is taken and used to initialize populations in all subsequent epochs (until our outer-loop criterion is met once again). Specifically, the initial population contains the strategies in support of $\mathcal{N}$ in proportion to their probabilities in the mixture distribution.

This new initialization process allows the evolution step to further challenge the Nash memory. While we do not converge onto the precise Nash strategy for our game (which is $\langle 100, 100 \rangle$), we do obtain a mixture $\mathcal{N}$ (with 123 pure strategies in support) that virtually eliminates vulnerability to all strategies ($\mathcal{N}$'s worst score ≈ -0.05) except the true Nash ($\mathcal{N}$'s score ≈ -0.323); the memory mechanism is poised to accept the true Nash strategy, if search can find it. Thus, the solution $\mathcal{N}$ is an excellent approximation to the true Nash strategy *in behavior*, though superficially they appear nothing alike.

8 Conclusion

We examine the performance of three distinct memory mechanisms using Watson and Pollack's intransitive numbers game: Nash memory, Best-of-Generation, and Dominance Tournament. We do not intend to argue that the BOG and DT mechanisms cannot improve the performance of a coevolutionary algorithm; indeed, the BOG approach is known to help, e.g. [9, 7]. We are more interested in the role of coevolutionary memory and its organization.

We show that the Nash memory improves its collection of traits, expressed as a mixed strategy, as search exposes the memory to new traits. Over time, the mixed strategy asymptotically approaches the performance of the Nash equilibrium strategy for the numbers game, thus providing 1) an excellent approximation of the (game-theoretic) solution, and 2) a gradually increasing challenge for evolving strategies. In contrast, the BOG method exhibits only limited ability, when faced with pervasive intransitivity, to increasingly challenge evolution; lacking a strong organizing principle, BOG does not converge to the Nash equilibrium. The DT method implements the principle of domination, but only with respect to local knowledge; without the global knowledge required to properly determine domination, the numbers game easily leads the DT method astray.

Acknowledgements

The authors wish to thank Anthony Bucci, Edwin de Jong, Shivakumar Viswanathan, and Richard Watson for many useful discussions.

References

1. R. Brooks and P. Maes, editors. *Proc. 4th Conf. on Artif. Life*. MIT Press, 1994.
2. D. Cliff and G. F. Miller. Tracking the red queen: Measurements of adaptive progress in co-evolutionary simulations. In F. Moran et al., editors, *3rd Euro. Conf. on Artificial Life*, pages 200–218. Springer Verlag, 1995.
3. S. G. Ficici, O. Melnik, and J. B. Pollack. A game-theoretic investigation of selection methods used in evolutionary algorithms. In A. Zalzala et al., editors, *Proc. of 2000 Congress on Evolutionary Computation*, pages 880–887. IEEE Press, 2000.
4. S. G. Ficici and J. B. Pollack. Challenges in coevolutionary learning: Arms-race dynamics, open-endedness, and mediocre stable states. In C. Adami et al., editors, *Proc. of the Sixth Conf. on Artificial Life*, pages 238–247. MIT Press, 1998.
5. D. Fudenberg and J. Tirole. *Game Theory*. MIT Press, 1998.
6. D. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison Wesley, 1989.
7. S. Nolfi and D. Floreano. Co-evolving predator and prey robots: Do ‘arm races’ arise in artificial evolution? *Artificial Life*, 4(4):311–335, 1998.
8. C. W. Reynolds. Competition, coevolution and the game of tag. In Brooks and Maes [1], pages 59–69.
9. C. Rosin and R. Belew. New methods for competitive co-evolution. *Evolutionary Computation*, 5(1):1–29, 1997.
10. K. Sims. Evolving 3d morphology and behavior by competition. In Brooks and Maes [1], pages 28–39.
11. K. O. Stanley and R. Miikkulainen. The dominance tournament method of monitoring progress in coevolution. In A. Barry, editor, *2002 Genetic and Evolutionary Computation Conference Workshop Program*, pages 242–248, 2002.
12. P. R. Thie. *An Introduction to Linear Programming and Game Theory*. John Wiley and Sons, 1988.
13. R. A. Watson and J. B. Pollack. Coevolutionary dynamics in a minimal substrate. In L. Spector et al., editors, *Proc. 2001 Genetic and Evolutionary Computation Conf.*, pages 702–709. Morgan Kaufmann, 2001.

The Paradox of the Plankton: Oscillations and Chaos in Multispecies Evolution

Jeffrey Horn and James Cattron

Department of Mathematics and Computer Science
Northern Michigan University
1401 Presque Isle Avenue
Marquette, Michigan, 49855, USA
jhorn@nmu.edu, jcattron@alumni.nmu.edu
<http://cs.nmu.edu/~jeffhorn,~jcattron>

Abstract. Two theoretical ecologists have recently discovered that even under the simplest models of competition, three species are sufficient to generate permanent oscillations, and five species can generate chaos (Huisman & Weissing, 2001). We can show that these results carry over into genetic algorithm (GA) resource sharing after making one minor change in the “usual” sharing methods. We also bring together previous, scattered results showing oscillatory and chaotic behavior in the “usual” GA sharing methods themselves. Thus one could argue that oscillations and chaos are fairly easy to generate once individuals are allowed to influence each other, even if such interactions are extremely simple, natural, and indirect, as they are under resource sharing. We suggest that great care be taken before assuming that any particular implementation of resource sharing leads to a unique and stable equilibrium.

1 Introduction and Background

Population biologists have long known about oscillations and chaotic behavior in multispecies competition models. But in the much simpler and more abstract models of evolution employed in genetic algorithms, we usually assume smooth convergence to stable equilibria (especially under selection alone). In particular, the simple and natural method of niching via the sharing of common resources has been shown to induce stable, long-term equilibria with multiple species co-existing (e.g., Horn, 1997). Yet even under the widely-used technique of simply dividing up finite resources among competing individuals, we can find evidence of oscillatory behavior and non-monotonic approaches to equilibrium.

1.1 Resource Sharing

A natural niching effect is implicitly induced by competition for limited resources (i.e., finite rewards). The sharing procedure we discuss here is common to the models of theoretical ecologists as well as to those of GA practitioners. The fundamental steps of resource sharing are intuitive:

1. For each of the finite resources r_i , divide it up among all individuals contending for it, in proportion to the strengths of their claims. (Thus two equally deserving individuals should be allocated equal amounts of the resource.)
2. For each individual, add all rewards/credits earned in the first step, and use this amount (perhaps scaled) as the fitness for GA selection.
3. After a new generation is produced, replenish/renew the resources and start over at the first step above.

The notions of competition and niche overlap are easy to visualize in the case of resource sharing. In Figure 1, the circle represents the resources *covered* by the corresponding species. In a learning classifier system, for example, the circles would represent the subset of examples that are correctly classified by individuals of that species¹. The resources in the overlapped niches are covered by multiple species, and must be shared among the individuals of *all* such species. (The resources covered by only one species must also be shared, but only among members of that one species.)

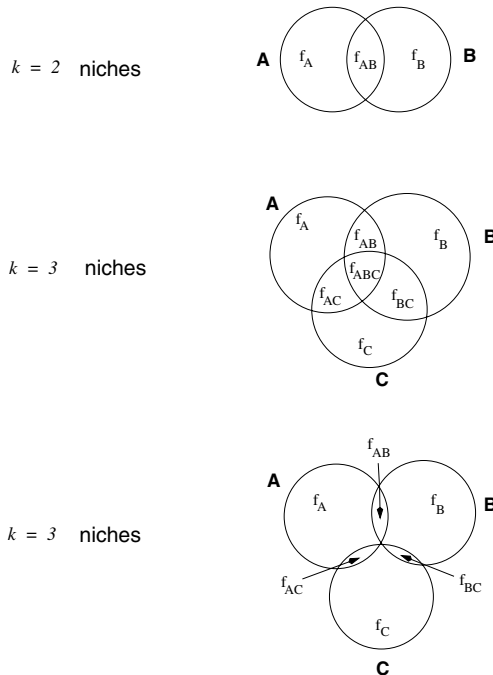


Fig. 1. Different situations of overlapping resource coverage.

¹ For the purposes of this paper, we will use the term “niche” to refer to such a subset of resources covered by a species.

To be explicit about the actual sharing mechanism, we calculate the shared fitness for the members of a species **A** in the situation shown in Figure 1, bottom (i.e., only pairwise niche overlaps, with $f_{ABC} = 0$). Let f_A , f_B , and f_C be the *objective* (i.e., unshared) fitnesses for rules **A**, **B**, and **C** respectively². Let f_{AB} be the amount of resources in the overlapping coverage of species **A** and **B**. That is, f_{AB} is the amount of resources shared by **A** and **B**. Let n_A, n_B, n_C be the number of members of each of the three species, in our population of size N (thus $N = n_A + n_B + n_C$). We calculate the shared fitness of **A**:

$$f_{sh,A} = \frac{f_A - f_{AB} - f_{AC}}{n_A} + \frac{f_{AB}}{n_A + n_B} + \frac{f_{AC}}{n_A + n_C}. \quad (1)$$

Similarly for $f_{sh,B}$ and $f_{sh,C}$.

To simulate an actual experiment (i.e., a run of a GA), we use the well-known method of *expected proportion equations*. We assume a generational GA with proportionate selection and no crossover or mutation:

$$P_{A,t+1} = \frac{n_{A,t} f_{sh,A,t}}{\sum_{\forall \text{species } X} (n_{X,t} f_{sh,X,t})} = \frac{P_{A,t} f_{sh,A,t}}{\sum_{\forall \text{species } X} (P_{X,t} f_{sh,X,t})}, \quad (2)$$

where $P_{A,t}$ means the proportion of the population taken up by copies of species **A** at time (generation) t (i.e., $n_{A,t+1}/N$), and $f_{sh,A,t}$ is the shared fitness (e.g., Equation 1) of **A** at time t .

Resource sharing is often incorporated in adaptive, or simulated, systems, including: learning classifier systems (LCS) (Booker, 1982; Wilson, 1994; Horn, Goldberg, & Deb, 1994), immune system models (Smith, Forrest, & Perelson, 1993), evolving cellular automata (Werfel, Mitchell, & Crutchfield, 2000; Juillé & Pollack, 1998), and ecological simulations (Huberman, 1988). It is known by other names, such as *example sharing* (McCallum & Spackman, 1990) and *shared sampling* (Rosin & Belew, 1997).

2 Oscillations in Traditional Models of Resource Sharing

2.1 Two-Species Oscillations

Oei, Goldberg, and Chang (1991) showed how the “naïve” combination of fitness sharing and tournament selection leads to oscillations and chaos. In their model, the population consisted solely of two species with no niche overlap³. If tournament selection is applied to the shared fitness values calculated at generation t , then the species with the lower shared fitness will lose every competition in which it is paired with the other species. This will result in an overcompensation by selection, in which the more crowded (and therefore less fit) species will

² In this paper we use the term *objective fitness* to mean the credit or reward earned by an individual for covering each resource, summed over all covered resources, **before** any sharing method is applied.

³ Since resource sharing and fitness sharing are equivalent methods when there is no niche overlap (Horn, 1997), the following result applies to resource sharing as well.

suddenly become the under-represented and more highly fit species in the next generation.

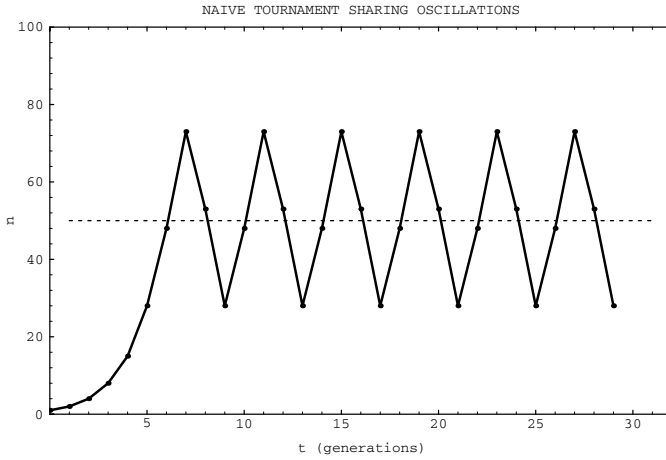


Fig. 2. Lotka-Volterra predator-prey oscillations under *naïve tournament sharing*.

These predicted swings in population are borne out in the plot of expected proportions shown in Figure 2. We trace the expected proportion of **A**s in a size $N = 100$ population under naïve tournament sharing with two species of equal objective fitness: $f_A = f_B$. We start with one copy of **A** at $t = 0$, the initial generation. We observe a rapid convergence toward equilibrium, followed by an overshooting, then an undershooting, and so on; an apparently periodic oscillation. This seems reminiscent of predator-prey oscillations generated by Lotka-Volterra growth equations.

Although the oscillations in Figure 2 seem periodic, we note that this is an artifact of the model. By looking only at expected values, we are averaging out the randomness of an individual run. An actual run would involve the stochastic nature of tournament selection (in the random selection of tournament competitors). Oei, et. al. (1991) predict chaotic behavior, calculating a Lyapunov exponent of approximately 0.21, even with only two niches. In the case of $k > 2$ niches, we can imagine that the oscillations would be so coupled as to always result in chaotic behavior.

2.2 Non-monotonic Convergence with Three Species

Horn (1997) analyzed the behavior and stability of resource sharing under proportionate selection. He looked at the existence and stability of equilibrium for all situations of overlap, but most of this analysis was limited to the case of only

two species. Horn did take a brief look at three overlapping niches, and found the following interesting result.

If all three pairwise niche overlaps are present (as in Figure 1), then it is possible to have non-monotonic convergence to equilibrium. That is, one or more species can “overshoot” its equilibrium proportion, as in Figure 3. This overshoot is *expected*, and is not due to stochastic effects of selection during a single run. We speculate that this “error” in expected convergence is related to the increased complexity of the niching equilibrium equations. For three mutually overlapped niches, the equilibrium condition yields a system of cubic equations to solve. Furthermore, the complexity of such equations for k mutually overlapping niches can be shown to be bounded from below: the equations must be polynomials of degree $2k - 3$ or greater (Horn, 1997).

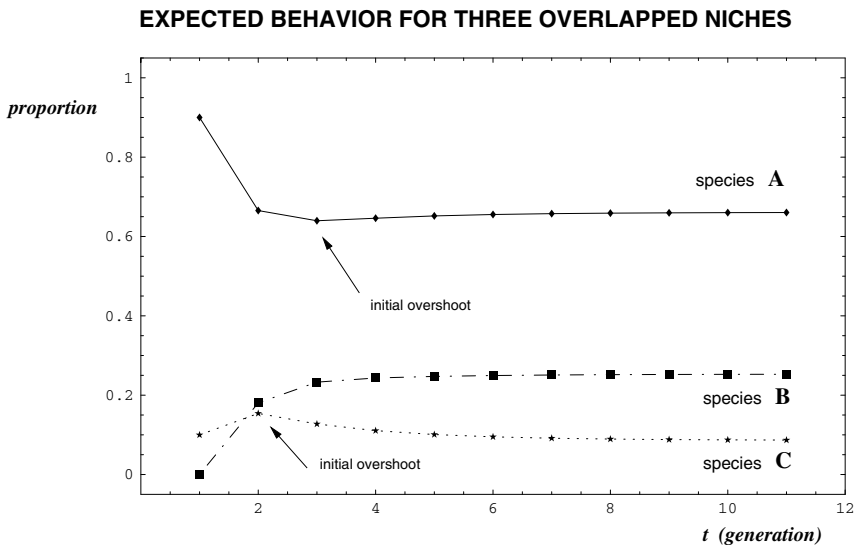


Fig. 3. Small, initial oscillations even under traditional “summed fitness”.

3 Phytoplankton Models of Resource Sharing

Recent work by two theoretical ecologists (Huisman & Weissing, 1999; 2001), has shown that competition for resources by as few as three species can result in long-term oscillations, even in the traditionally convergent models of plankton species growth. For as few as five species, apparently chaotic behavior can emerge. Huisman and Weissing propose these phenomena as one possible new explanation of the *paradox of the plankton*, in which the number of co-existing plankton species far exceeds the number of limiting resources, in direct contradiction of theoretical predictions. Continuously fluctuating species levels can

support more species than a steady, stable equilibrium distribution. Their results show that external factors are not necessary to maintain non-equilibrium conditions; the inherent complexity of the “simple” model itself can be sufficient.

Here we attempt to extract the essential aspects of their models and duplicate some of their results in our models of resource sharing in GAs. We note that there are major differences between our model of resource sharing in a GA and their “well-known resource competition model that has been tested and verified extensively using competition experiments with phytoplankton species” (Huisman & Weissing, 1999). For example, where we assume a fixed population size, their population size varies and is constrained only by the finite resources themselves. Still, there are many similarities, such as the sharing of resources.

3.1 Differential Competition

First we try to induce oscillations among multiple species by noting that Huisman and Weissing’s models allow *differential competition* for overlapped resources. That is, one species **I** might be better than another species **J** when competing for the resources in their overlap f_{IJ} . Thus species **I** would obtain a greater share of f_{IJ} than would **J**. In contrast, our models described above all assume equal competitiveness for overlapped resources, and so we have always divided the contested resources evenly among species.

Now we try to add this differential competition to our model. In the phytoplankton model, c_{ij} denotes the *content* of resource i in species j . In our model we will let $c_{I,IJ}$ denote the competitive advantage of species **I** over species **J** in obtaining the resource f_{IJ} . Thus $c_{A,AB} = 2.0$ means that **A** is twice as good as **B** at obtaining resources from the overlap f_{AB} , and so **A** will receive twice the share that **B** gets from this overlap:

$$f_{sh,A} = \frac{f_A - f_{AB}}{n_A} + \frac{c_{A,AB} * f_{AB}}{c_{A,AB} * n_A + n_B} \quad f_{sh,B} = \frac{f_B - f_{AB}}{n_B} + \frac{f_{AB}}{c_{A,AB} * n_A + n_B}. \quad (3)$$

This generalization⁴ seems natural. What can it add to the complexity of multispecies competition? We looked at the expected evolution of five species, with pairwise niche overlaps and different competitive resource ratios. After some experimentation, the most complex behavior we were able to generate is a “double overshoot” of equilibrium by a species, similar to Figure 3. This is a further step away from the usual monotonic approach to equilibrium, but does not seem a promising way to show long-term oscillations and non-equilibrium dynamics.

3.2 The Law of the Minimum

Differential competition does not seem to be enough to induce long-term oscillations in our GA model of resource sharing. We note another major difference

⁴ Note that we get back our original shared fitness formulae by setting all competitive factors $c_{I,IJ}$ to one.

between our model and the Plankton model. Huisman and Weissing (2000) “assume that the specific growth rates follow the Monod equation, and are determined by the resource that is the most limiting according to Liebig’s ‘law of the minimum’:

$$\mu_i(R_1, \dots, R_k) = \min \left(\frac{r_i R_1}{K_{1i} + R_1}, \dots, \frac{r_i R_k}{K_{ki} + R_k} \right), \quad (4)$$

where R_i are the k resources being shared.

Since a *min* function can sometimes introduce “switching” behavior, we attempt to incorporate it in our model of resource sharing. Whereas we simply summed the different components of the shared fitness expression (Equation 1), we might instead take the minimum of the components:

$$f_{sh,A} = \min \left(\frac{f_A - f_{AB} - f_{AC}}{n_A}, \frac{c_{A,AB} * f_{AB}}{c_{A,AB} * n_A + n_B}, \frac{c_{A,AC} * f_{AC}}{c_{A,AC} * n_A + n_C} \right). \quad (5)$$

Note that we have added the competitive factors introduced in Equation 3 above. We want to use differential competition to induce a *rock-paper-scissors* relationship among the three overlapping species, as in (Huisman & Weissing, 1999). To do so, we set our competitive factors as follows: $c_{A,AB} = 2$, $c_{B,BC} = 2$, and $c_{C,AC} = 2$, with all other $c_{I,IJ} = 1$. Thus **A** “beats” **B**, **B** beats **C**, and **C** beats **A**. These settings are meant to induce a cyclical behavior, in which an increase in the proportion of species **A** causes a decline in species **B** which causes an increase in **C** which causes a decline in **A**, and so on.

Plugging the shared fitness of Equation 5 into the expected proportions of Equation 2, we plot the time evolution of expected proportions in Figure 4, assuming starting proportions of $P_{A,0} = 0.2$, $P_{B,0} = 0.5$, $P_{C,0} = 0.3$. Finally, we see the “non-transient” oscillations that Huisman and Weissing were able to find. These follow the rock-paper-scissors behavior of sequential ascendancy of each species in the cycle.

3.3 Five Species and Chaos

Huisman and Weissing were able to induce apparently chaotic behavior with as few as five species (in contrast to the seemingly periodic oscillations for three species). Here we attempt to duplicate this effect in our modified model of GA resource sharing.

In (Huisman & Weissing, 2001), the authors set up two rock-paper-scissors “trios” of species, with one species common to both trios. This combination produced chaotic oscillations. We attempt to follow their lead by adding two new species **D** and **E** in a rock-scissors-paper relationship with **A**. In Figure 5 we can see apparently chaotic oscillations that eventually lead to the demise of one species, **C**. The loss of a species seems to break the chaotic cycling, and it appears that immediately a stable equilibrium distribution of the four remaining species is reached.

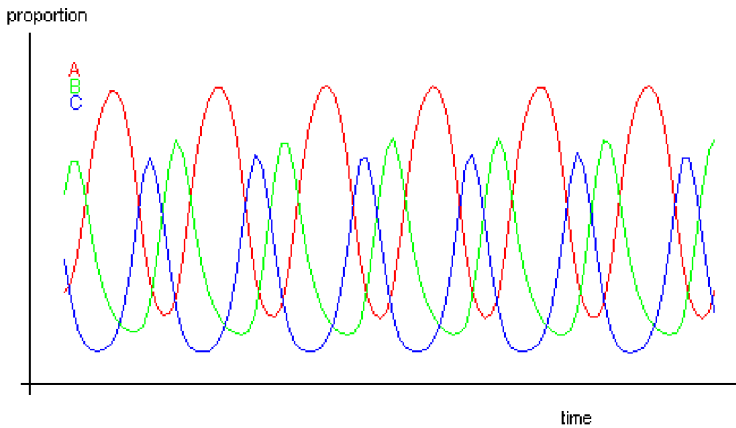


Fig. 4. Permanent oscillations.

We consider the extinction of a member species to signify the end of a trio. We can then ask which trio will win, given a particular initial population distribution. Huisman and Weissing found in their model that the survival of each species, and hence the success of the trios, was highly dependent on the initial conditions, such as the initial species counts. They proceeded to generate fractal-like images in graphs in which the independent variables are the initial species counts and the dependent variable, dictating the color at that coordinate, is the identity of the winning (surviving) trio.

Here we investigate whether our model can generate a fractal-like image based on the apparently chaotic behavior exhibited in Figure 5. We choose to vary the initial proportions of species **B** (x-axis), and **D** (y-axis). Since we assume a fixed population size (unlike Huisman and Weissing), we must decrease other species' proportions as we increase another's. We choose to set $P_{C,0} = 0.4 - P_{B,0}$ and $P_{E,0} = 0.4 - P_{D,0}$, leaving $P_{A,0} = 0.2$. Thus we are simply varying the ratio of two members of each trio, on each axis. Only the initial proportions vary. All other parameters, such as the competitive factors and all of the fitnesses, are constant.

Since our use of proportions implies an infinite population, we arbitrarily choose a threshold of 0.000001 to indicate the extinction of a species, thus simulating a population size of one million. If $P_{X,t}$ falls below $\frac{1}{N} = 0.000001$, then species **X** is considered to have gone extinct, and its corresponding trio(s) is considered to have lost. In Figure 6 we plot the entire range of feasible values of $P_{B,0}$ and $P_{C,0}$. The resolution of our grid is 400 by 400 "pixels". We color each of the 160,000 pixels by iterating the expected proportions equations (as in Equation 5) until a species is eliminated or until a maximum of 300 generations is reached. We then color the pixel as shown in the legend of Figure 6: red for

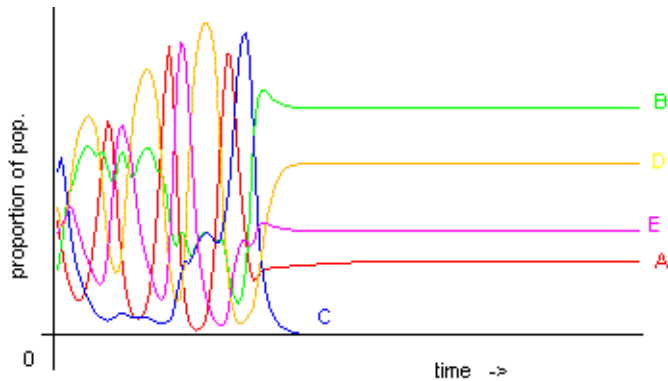


Fig. 5. Chaotic, transient oscillations leading to extinction.

a win by trio ABC, blue for an ADE win, and yellow if neither trio has been eliminated by the maximum number of generations⁵.

Figure 6 exhibits fractal characteristics, although further analysis is needed before we can call it a fractal. But we can gain additional confidence by plotting a much narrower range of initial proportion values and finding similar complexity. In Figure 7 we look at a region from Figure 6 that is one one hundredth the range along both axes, thus making the area one ten thousandth the size of the plot in Figure 6. We still plot 400 by 400 pixels, and at such resolution we see no less complexity.

3.4 Discussion

How relevant are these results? The most significant change we made to GA resource sharing was the substitution of the *min* function for the usual Σ (sum) function in combining the components of shared fitness. How realistic is this change?

For theoretical ecologists, Liebig's *law of the minimum* is widely accepted as modeling the needs of organisms to reproduce under competition for a few limited resources. In the case of phytoplankton, resources such as nitrogen, iron, phosphorus, silicon, and sunlight are all critical for growth, so that the least available becomes the primary limiting factor of the moment. We could imagine a similar situation for simulations of life, and for artificial life models. Instances from other fields of applied EC seem plausible. For example, one could imagine the evolution of robots (or robot strategies) whose ultimate goal is to assemble “widgets” by obtaining various widget parts from a complex environment (e.g.,

⁵ We also use green to signify that species **A**, a member of both trios, was the first to go. But that situation did not arise in our plots.

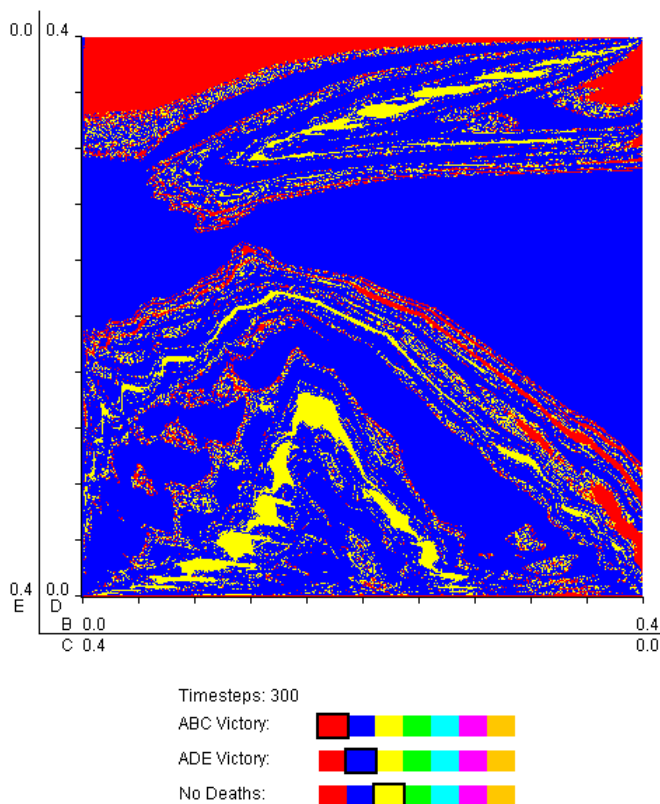


Fig. 6. An apparently fractal pattern.

a junkyard). The number of widgets that a robot can assemble is limited by the part which is hardest for the robot to obtain. If the stockpile of parts are “shared” among the competing robots, then indeed the law of the minimum applies.

4 Conclusions and Future Work

There seem to be many ways to implement resource sharing with oscillatory and even chaotic behavior. Yet resource (and fitness) sharing are generally associated with unique, stable, steady-state populations of multiple species. Indeed, the oscillations and chaos we have seen under sharing are better known and studied in the field of *evolutionary game theory* (EGT), in which species compete pairwise according to a payoff matrix, and selection is performed based on each individual’s total payoff.

For example, Ficici, et. al. (2000) found oscillatory and chaotic behavior similar to that induced by *naïve tournament sharing*, but for other selection

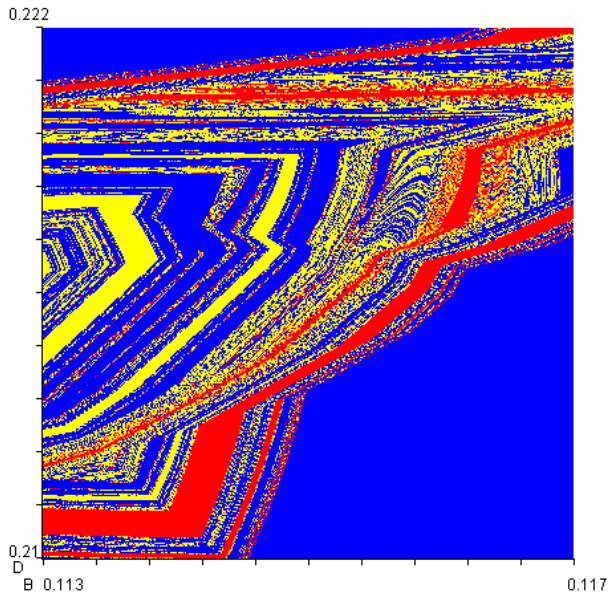


Fig. 7. Zooming in on $\frac{1}{10,000}^{th}$ of the previous plot.

schemes (e.g., truncation, linear-rank, Boltzmann), when the selection pressure was high. Although they did not analyze fitness or resource sharing specifically, their domain, the *Hawk-Dove* game, induces a similar coupling (Lotka-Volterra) between two species.

Another example of a tie-in with EGT is the comparison of our rock-paper-scissors, five-species results with the work of Watson and Pollack (2001). They investigate similar dynamics arising from “intransitive superiority”, in which a species **A** beats species **B** which beats species **C** which beats **A**, according to the payoff matrix.

Clearly there is a relationship between the interspecies dynamics introduced by resource sharing and those induced by pairwise games. There are also clear differences, however. While resource sharing adheres to the principal of conservation of resources, EGT in general involves non-zero-sum games. Still, it seems that a very promising extension of our findings here would be mapping resource sharing to EGT payoff matrices.

It appears then that some of the unstable dynamics recently analyzed in theoretical ecology and in EGT can find their way into our GA runs via resource sharing, once considered a rather weak, passive, and predictable form of species interaction. In future, we as practitioners must be careful not to assume the existence of a unique, stable equilibrium under every regime of resource sharing.

References

- Booker, L. B. (1989). Triggered rule discovery in classifier systems. In J. D. Schaffer, (Ed.), *Proceedings of the Third International Conference on Genetic Algorithms (ICGA 3)*. San Mateo, CA: Morgan Kaufmann. 265–274.
- Ficici, S. G., Melnik, O., & Pollack, J. B. (2000). A game-theoretic investigation of selection methods used in evolutionary algorithms. In A. Zalzala, et al (Ed.s), *Proceedings of the 2000 Congress on Evolutionary Computation*. IEEE Press.
- Horn, J. (1997). *The Nature of Niching: Genetic Algorithms and the Evolution of Optimal, Cooperative Populations*. Ph.D. thesis, University of Illinois at Urbana-Champaign, (UMI Dissertation Services, No. 9812622).
- Horn, J., Goldberg, D. E., & Deb, K. (1994). Implicit niching in a learning classifier system: nature's way. *Evolutionary Computation*, 2(1). 37–66.
- Huberman, B. A. (1988). The ecology of computation. In B. A. Huberman (Ed.), *The Ecology of Computation*. Amsterdam, Holland: Elsevier Science Publishers B. V. 1–4.
- Huisman, J., & Weissing, F. J. (1999). Biodiversity of plankton by species oscillations and chaos. *Nature*, 402. November 25, 1999, 407–410.
- Huisman, J., & Weissing, F. J. (2001). Biological conditions for oscillations and chaos generated by multispecies competition. *Ecology*, 82(10). 2001, 2682–2695.
- Juillé, H., & Pollack, J. B. (1998). Coevolving the “ideal” trainer: application to the discovery of cellular automata rules. In J. R. Koza, et. al., (Ed.s), *Genetic Programming 1998*. San Francisco, CA: Morgan Kaufmann. 519–527.
- McCallum, R. A., & Spackman, K. A. (1990). Using genetic algorithms to learn disjunctive rules from examples. In B. W. Porter & R. J. Mooney, (Ed.s), *Machine Learning: Proceedings of the Seventh International Conference*. Palo Alto, CA: Morgan Kaufmann. 149–152.
- Oei, C. K., Goldberg, D. E., & Chang, S. (1991) Tournament selection, niching, and the preservation of diversity. *IlligAL Report No. 91011*. Illinois Genetic Algorithms Laboratory, University of Illinois at Urbana-Champaign, Urbana, IL. December, 1991.
- Rosin, C. D., & Belew, R. K. (1997). New methods for competitive coevolution. *Evolutionary Computation*, 5(1). Spring, 1997, 1–29.
- Smith, R. E., Forrest, S., & Perelson, A. S. (1993). Searching for diverse, cooperative populations with genetic algorithms. *Evolutionary Computation*, 1(2). 127–150.
- Watson, R.A., & Pollack, J.B. (2001). Coevolutionary dynamics in a minimal substrate. In L. Spector, et. al. (Ed.s), *Proceedings of the 2001 Genetic and Evolutionary Computation Conference*, Morgan Kaufmann.
- Werfel, J., Mitchell, M., & Crutchfield, J. P. (1999). Resource sharing and coevolution in evolving cellular automata. *IEEE Transactions on Evolutionary Computation*, 4(4). November, 2000, 388–393.
- Wilson, S. W. (1994). ZCS: A zeroth level classifier system. *Evolutionary Computation*, 2(1). 1–18.

Exploring the Explorative Advantage of the Cooperative Coevolutionary (1+1) EA

Thomas Jansen^{1*} and R. Paul Wiegand²

¹ FB 4, LS2, Univ. Dortmund, 44221 Dortmund, Germany

Thomas.Jansen@udo.edu

² Krasnow Institute, George Mason University, Fairfax, VA 22030

paul@tesseract.org

Abstract. Using a well-known cooperative coevolutionary function optimization framework, a very simple cooperative coevolutionary (1+1) EA is defined. This algorithm is investigated in the context of expected optimization time. The focus is on the impact the cooperative coevolutionary approach has and on the possible advantage it may have over more traditional evolutionary approaches. Therefore, a systematic comparison between the expected optimization times of this coevolutionary algorithm and the ordinary (1+1) EA is presented. The main result is that separability of the objective function alone is not sufficient to make the cooperative coevolutionary approach beneficial. By presenting a clear structured example function and analyzing the algorithms' performance, it is shown that the cooperative coevolutionary approach comes with new explorative possibilities. This can lead to an immense speed-up of the optimization.

1 Introduction

Coevolutionary algorithms are known to have even more complex dynamics than ordinary evolutionary algorithms. This makes theoretical investigations even more challenging. One possible application common to both evolutionary and coevolutionary algorithms is optimization. In such applications, the question of the optimization efficiency is of obvious high interest. This is true from a theoretical, as well as from a practical point of view. While for evolutionary algorithms such run time analyses are known, we present results of this type for a coevolutionary algorithm for the first time. Coevolutionary algorithms may be designed for function optimization applications in a wide variety of ways. The well-known cooperative coevolutionary optimization framework provided by Potter and De Jong (7) is quite general and has proven to be advantageous in different applications (e.g., Iorio and Li (4)). An attractive advantage of this framework is that any evolutionary algorithm (EA) can be used as a component of the framework.

* The research was partly conducted during a visit to George Mason University. This was supported by a fellowship within the post-doctoral program of the German Academic Exchange Service (DAAD).

However, since these cooperative coevolutionary algorithms involve several EAs working almost independently on separate *pieces* of a problem, one of the key issues with the framework is the question of how a problem representation can be decomposed in productive ways. Since we concentrate our attention on the maximization of pseudo-Boolean functions $f: \{0,1\}^n \rightarrow \mathbb{R}$, there are very natural and obvious ways we can make such representation choices. A bit string $x \in \{0,1\}^n$ of length n is divided into k separate components $x^{(1)}, \dots, x^{(k)}$. Given such a decomposition, there are then k EAs, each operating on one of these components. When a function value has to be computed, a bit string of length n is reconstructed from the individual components by picking representative individuals from the other EAs. Obviously, the choice of the EA that serves as underlying search heuristic has great impact on the performance of this cooperative coevolutionary algorithm (CCEA). We use the well-known (1+1) EA for this purpose because we feel that it is perhaps the simplest EA that still shares many important properties with more complex EAs, which makes it an attractive candidate for analysis.

Whether this mechanism of dividing the optimization problem f into k sub-problems and treating them almost independently of one another is an advantage strongly depends on properties of the function f . In applications, *a priori* knowledge about f is required in order to define an appropriate division. We neglect this problem here and investigate only problems where the division in sub-problems matches the objective function f . The investigation of the impact of the separation of inseparable parts is beyond the scope of this paper. Intuitively, separability of f seems to be necessary for the CCEA to have advantages over that EA this is used as underlying search heuristic. After all, we could solve linearly separable blocks with completely independent algorithms and then concatenate the solutions, if we like. Moreover, one expects that such an advantage should grow with the degree of separability of the objective function f . Indeed, in the extreme we could imagine a lot of algorithms simultaneously solving lots of little problems, then aggregating the solutions. Linear functions like the well-known OneMax problem have a maximal degree of separability. This makes them natural candidates for our investigations. Regardless of our intuition, however, it will turn out that separability alone is not sufficient to make the CCEA superior to the “stand-alone EA.”

Another aspect that comes with the CCEA are increased explorative possibilities. Important EA parameters, like the mutation probability, are often defined depending on the string length, i.e., the dimension of the search space. For binary mutations, $1/n$ is most often recommended for strings of length n . Since the components have shorter length, an increased mutation probability is the consequence. This differs from increased mutation probabilities in a “stand-alone” EA in two ways. First, one can have different mutation probabilities for different components of the string with a CCEA in a natural way. Second, since mutation is done in the components separately, the CCEA can search in these components more efficiently, while the partitioning mechanism may afford the algorithm some added protection from the increased disruption. The components that are not

“active” are guaranteed not to be changed in that step. We present a class of example functions where this becomes very clear.

In the next section we give precise formal definitions of the (1+1) EA, the CC (1+1) EA, the notion of separability, and the notion of expected optimization time. In Section 3 we analyze the expected optimization time of the CC (1+1) EA on the class of linear functions and compare it with the expected optimization time of the (1+1) EA. Surprisingly, we will see that in spite of the total separability of linear functions the CC (1+1) EA has no advantage over the (1+1) EA. This leads us to concentrate on the effects of the increased mutation probability. In Section 4, we define a class of example functions, CLOB, and analyze the performance of the (1+1) EA and the CC (1+1) EA. We will see that the cooperative coevolutionary function optimization approach can reduce the expected optimization time from super-polynomial to polynomial or from polynomial to a polynomial of much smaller degree. In Section 5, we conclude with a short summary and a brief discussion of possible directions of future research.

2 Definitions and Framework

The (1+1) EA is an extremely simple evolutionary algorithm with population size 1, no crossover, standard bit-wise mutations, and plus-selection known from evolution strategies. Due to its simplicity it is an ideal subject for theoretical research. In fact, there is a wealth of known results regarding its expected optimization time on many different problems (Mühlenbein (6), Rudolph (9), Garnier, Kallel, Schoenauer (3), Droste, Jansen, and Wegener (2)). Since we are interested in a comparison of the performance of the EA alone as opposed to its use in the CCEA, known results and, even more importantly, known analytical tools and methods (Droste et. al. 1) are important aspects that make the (1+1) EA the ideal choice for us.

Algorithm 1. ((1+1) Evolutionary Algorithm ((1+1) EA))

1. **Initialization**
Choose $x \in \{0,1\}^n$ uniformly at random.
2. **Mutation**
Create y by copying x and, independently for each bit flip this bit with probability $1/n$.
3. **Selection**
If $f(y) \geq f(x)$, set $x := y$.
4. **Continue at line 2.**

We do not care about finding an appropriate stopping criterion and let the algorithm run forever. In our analysis we are interested in the first point of time when $f(x)$ is maximal, i.e., a global maximum is found. As a measure of time we count the number of function evaluations.

For the CC (1+1) EA, we have to divide x into k components. For the sake of simplicity, we assume that x can be divided into k components of equal length

l , i.e., $l = n/k \in \mathbb{N}$. The generalization of our results to the case $n/k \notin \mathbb{N}$ with $k - 1$ components of equal length $\lfloor n/k \rfloor$ and one longer component of length $n - (k - 1) \cdot \lfloor n/k \rfloor$ is trivial. The k components are denoted as $x^{(1)}, \dots, x^{(k)}$ and we have $x^{(i)} = x_{(i-1) \cdot l + 1} \dots x_{i \cdot l}$ for each $i \in \{1, \dots, k\}$. For the functions considered here, this is an appropriate way of distributing the bits to the k components.

Algorithm 2. (Cooperative Coevolutionary (1+1) EA (CC (1+1) EA))

1. **Initialization**
Independently for each $i \in \{1, \dots, k\}$,
choose $x^{(i)} \in \{0, 1\}^l$ uniformly at random.
2. $a := 1$
3. **Mutation**
Create $y^{(a)}$ by copying $x^{(a)}$ and, independently for each bit,
flip this bit with probability $\min\{1/l, 1/2\}$.
4. **Selection**
If $f(x^{(1)} \dots y^{(a)} \dots x^{(k)}) \geq f(x^{(1)} \dots x^{(a)} \dots x^{(k)})$, set $x^{(a)} := y^{(a)}$.
5. $a := a + 1$
6. If $a > k$, then continue at line 2, else continue at line 3.

We use $\min\{1/l, 1/2\}$ as mutation probability instead of $1/l$ in order to deal with the case $k = n$, i.e., $l = 1$. We consider $1/2$ to be an appropriate upper bound on the mutation probability. The idea of mutation is to create small random changes. A mutation probability of $1/2$ is already equivalent to pure random search. Indeed, larger mutation probabilities are against this basic “small random changes” idea of mutation. This can be better for some functions and is in fact superior for the functions considered here. Since this introduces annoying special cases that have hardly any practical relevance, we exclude this extreme case.

The CC (1+1) EA works with k independent (1+1) EAs. The i -th (1+1) EA operates on $x^{(i)}$ and creates the offspring $y^{(i)}$. For the purpose of selection the k strings $x^{(i)}$ are concatenated and the function value of this string is compared to the function value of the string that is obtained by replacing $x^{(a)}$ by $y^{(a)}$. The (1+1) EA with number a is called *active*. Again, we do not care about a stopping criterion and analyze the first point of time until the function value of a global maximum is evaluated. Here we also use the number of function evaluations as time measure.

Consistent with existing terminology in the literature (Potter and De Jong 8), we call one iteration of the CC (1+1) EA where one mutation and one selection step take place a *generation*. Note, that it takes k generations until each (1+1) EA was active once. Since this is an event of interest, we call k consecutive generations a *round*.

Definition 1. Let the random variable T denote the number of function evaluations until for some $x \in \{0, 1\}^n$ with $f(x) = \max\{f(x') \mid x' \in \{0, 1\}^n\}$ the

function value $f(x)$ is computed by the considered evolutionary algorithm. The expectation $E(T)$ is called expected optimization time.

When analyzing the expected run time of randomized algorithms, one finds bounds of this expected run time depending on the input size (Motwani and Raghavan 5). Most often, asymptotic bounds for growing input lengths are given. We adopt this perspective and use the dimension of the search space n as measure for the “input size.” We use the well-known O , Ω , and Θ notions to express upper, lower, and matching upper and lower bounds for the expected optimization time.

Definition 2. Let $f, g: \mathbb{N}_0 \rightarrow \mathbb{R}$ be two functions. We say $f = O(g)$, if

$$\exists n_0 \in \mathbb{N}, c \in \mathbb{R}^+: \forall n \geq n_0: f(n) \leq c \cdot g(n)$$

holds. We say $f = \Omega(g)$, if $g = O(f)$ holds. We say $f = \Theta(g)$, if $f = O(g)$ and $f = \Omega(g)$ both hold.

As discussed in Section 1, an important property of pseudo-Boolean functions is separability. For the sake of clarity, we give a precise definition.

Definition 3. Let $f: \{0, 1\}^n \rightarrow \mathbb{R}$ be any pseudo-Boolean function. We say that f is s -separable if there exists a partition of $\{1, \dots, n\}$ into disjoint sets $I_1, \dots, I_r$, where $1 \leq r \leq n$, and if there exists a matching number of pseudo-Boolean functions $g_1, \dots, g_r$ with $g_j: \{0, 1\}^{|I_j|} \rightarrow \mathbb{R}$ such that

$$\forall x = x_1 \cdots x_n \in \{0, 1\}^n: f(x) = \sum_{j=1}^r g_j \left(x_{i_{j,1}} \cdots x_{i_{j,|I_j|}} \right)$$

holds, with $I_j = \{i_{j,1}, \dots, i_{j,|I_j|}\}$ and $|I_j| \leq s$ for all $j \in \{1, \dots, r\}$.

We say that f is exactly s -separable, if f is s -separable but not $(s-1)$ -separable.

If a function f is known to be s -separable, it is possible to use the sets I_j for a division of x for the CC (1+1) EA. Then each (1+1) EA operates on a function g_j and the function value f is the sum of the g_j -values. If the decomposition into sub-problems is expected to be beneficial, it should be so if s is small and the decomposition matches the sets I_j . Obviously, the extreme case $s = 1$ corresponds to linear functions, where the function value is the weighted sum of the bits, i. e., $f(x) = w_0 + w_1 \cdot x_1 + \cdots + w_n \cdot x_n$ with $w_0, \dots, w_n \in \mathbb{R}$. Therefore, we investigate the performance of the CC (1+1) EA on linear functions first.

3 Linear Functions

Linear functions, or 1-separable functions, are very simple functions. They can be optimized bit-wise without any interaction between different bits. It is easy to see that this can be done in $O(n)$ steps. An especially simple linear function

is OneMax, where the function value equals the number of ones in the bit-string. It is long known that the (1+1) EA has expected optimization time $\Theta(n \log n)$ on OneMax (Mühlenbein 6). The same bound holds for any linear function without zero weights, and the upper bound $O(n \log n)$ holds for any linear function (Droste, Jansen, and Wegener 2). We want to compare this with the expected optimization time of the CC (1+1) EA.

Theorem 1. *The expected optimization time of the CC (1+1) EA for a linear function $f: \{0, 1\}^n \rightarrow \mathbb{R}$ with all non-zero weights is $\Omega(n \log n)$ regardless of the number of components k .*

Proof. According to our discussion we have $k \in \{1, \dots, n\}$ with $n/k \in \mathbb{N}$. We denote the length of each component by $l := n/k$. First, we assume $k < n$. We consider $(n - k) \ln n$ generations of the CC (1+1) EA and look at the first (1+1) EA operating on the component $x^{(1)}$. This EA is active in each k -th generation. Thus, it is active in $((n - k) \ln n)/k = (l - 1) \ln n$ of those generations. With probability $1/2$, at least half of the bits need to flip at least once after random initialization. This is true since we assume that all weights are different from 0. Therefore, each bit has an unique optimal value, 1 for positive weights and 0 for negative weights. The probability that among $l/2$ bits there is at least one that has not flipped at all is bounded below by

$$\begin{aligned} 1 - \left(1 - \left(1 - \frac{1}{l}\right)^{(l-1) \ln n}\right)^{l/2} &\geq 1 - (1 - e^{-\ln n})^{l/2} = 1 - \left(1 - \frac{1}{n}\right)^{l/2} \\ &\geq 1 - e^{-1/(2k)} \geq 1 - \frac{1}{1 + 1/(2k)} = \frac{1}{2k + 1} \geq \frac{1}{3k}. \end{aligned}$$

Since the k (1+1) EA are independent, the probability that there is one that has not reached the optimum is bounded below by $1 - (1 - 1/(3k))^k \geq 1 - e^{-1/3}$. Thus, the expected optimization time of the CC (1+1) EA with $k < n$ on a linear function without zero weights is $\Omega(n \log n)$.

For $k = n$ we have n (1+1) EA with mutation probability $1/2$ operating on one bit each. Each bit has an unique optimal value. We are waiting for the first point of time when each bit has had this optimal value at least once. This is equivalent to throwing n coins independently and repeating this until each coin came up head at least once. On average, the number of coins that never came up head is halved in each round. It is easy to see that on average this requires $\Omega(\log n)$ rounds with all together $\Omega(n \log n)$ coin tosses. $\square$

We see that the CC (1+1) EA has no advantage over the (1+1) EA at all on linear functions in spite of their total separability. This holds regardless of the number of components k . We conjecture that the expected optimization time is $\Theta(n \log n)$, i. e., asymptotically equal to the (1+1) EA. Since this leads away from our line of argumentation we do not investigate this conjecture here.

4 A Function Class with Tunable Advantage for the CC (1+1) EA

Recall that there were two aspects of the CC (1+1) EA framework that could lead to potential advantage over a (1+1) EA: partitioning of the problem and increased focus of the variation operators on the smaller components created by the partitioning. However, as we have just discussed, we now know that separability alone is not sufficient to make the cooperative coevolutionary optimization framework advantageous. Now we turn our attention to the second piece of the puzzle: increased explorative attention on the smaller components. More specifically, dividing the problem to be solved by separate (1+1) EAs results in an increased mutation probability in our case.

Let us consider one round of the CC (1+1) EA and compare this with k generations of the (1+1) EA. Remember that we use the number of function evaluations as measure for the optimization time. Note that both algorithms make the same number of function evaluations in the considered time period. We concentrate on $l = n/k$ bits that form one component in the CC (1+1) EA, e. g., the first l bits. In the CC (1+1) EA the (1+1) EA operating on these bits is active once in this round. The expected number of b bit mutations, i. e., mutations where exactly b bits in the bits $x_1, \dots, x_l$ flip, equals $\binom{l}{b} \left(\frac{1}{l}\right)^b \left(1 - \frac{1}{l}\right)^{l-b}$. For the (1+1) EA in one generation the expected number of b bit mutations in the bits $x_1, \dots, x_l$ equals $\binom{l}{b} \left(\frac{1}{n}\right)^b \left(1 - \frac{1}{n}\right)^{l-b}$. Thus, in one round, or k generations, the expected number of such b bit mutations equals $k \cdot \binom{l}{b} \left(\frac{1}{n}\right)^b \left(1 - \frac{1}{n}\right)^{l-b}$. For $b = 1$ we have $\left(1 - \frac{1}{l}\right)^{l-1}$ for the CC (1+1) EA and $\left(1 - \frac{1}{n}\right)^{l-1}$ for the (1+1) EA which are similar values. For $b = 2$ we have $((l-1)/(2l)) \left(1 - \frac{1}{l}\right)^{l-2}$ for the CC (1+1) EA and $((l-1)/(2n)) \left(1 - \frac{1}{n}\right)^{l-2}$ for the (1+1) EA, which is approximately a factor $1/k$ smaller. For small b , i. e., for the most relevant cases, the expected number of b bit mutations is approximately a factor of k^{b-1} larger for the CC (1+1) EA than for the (1+1) EA. This may result in an huge advantage for the CC (1+1) EA.

In order to investigate this, we define an objective function, which is separable and requires b bit mutations in order to be optimized. Since we want results for general values of b , we define a class of functions with parameter b . We use the well-known LeadingOnes problem as inspiration (Rudolph 9).

Definition 4. For $n \in \mathbb{N}$ and $b \in \{1, \dots, n\}$ with $n/b \in \mathbb{N}$ we define the function $LOB_b: \{0, 1\}^n \rightarrow \mathbb{R}$ (short for *LeadingOnesBlocks*) by

$$LOB_b(x) := \sum_{i=1}^{n/b} \prod_{j=1}^{b \cdot i} x_j$$

for all $x \in \{0, 1\}^n$.

LOB_b is identical to the so-called Royal Staircase function (van Nimwegen and Crutchfield 10) which was defined and used in a different context. Obviously,

the function value $\text{LOB}_b(x)$ equals the number of consecutive blocks of length b with all bits set to one (scanning x from left to right). Consider the (1+1) EA operating on LOB_b . After random initialization the bits have random values and all bits right of the left most bit with value 0 remain random (see Droste, Jansen, and Wegener 2 for a thorough discussion). Therefore, it is not at all clear that b bit mutations are needed. Moreover, LOB_b is not separable, i. e., it is exactly n -separable. We resolve both issues by embedding LOB_b in another function definition. The difficulty with respect to the random bits is resolved by taking a leading ones block of a higher value and subtracting OneMax in order to force the bits right of the left most zero bit to become zero bits. We achieve separability by concatenating k independent copies of such functions, which is a well-known technique to generate functions with a controllable degree of separability.

Definition 5. For $n \in \mathbb{N}$, $k \in \{1, \dots, n\}$ with $n/k \in \mathbb{N}$, and $b \in \{1, \dots, n/k\}$ with $n/(bk) \in \mathbb{N}$, we define the function $\text{CLOB}_{b,k}: \{0, 1\}^n \rightarrow \mathbb{R}$ (short for Concatenated LOB) by

$$\text{CLOB}_{b,k}(x) := \left(\sum_{h=1}^k n \cdot \text{LOB}_b(x_{(h-1) \cdot l + 1} \cdots x_{h \cdot l}) \right) - \text{OneMax}(x)$$

for all $x = x_1 \cdots x_n \in \{0, 1\}^n$, with $l := n/k$.

We have k independent functions, the i -th function operates on the bits $x_{(i-1) \cdot l + 1} \cdots x_{i \cdot l}$. For each of these functions the function value equals n times the number of consecutive leading ones blocks (where b is the size of each block) minus the number of one bits in all its bit positions. The function value $\text{CLOB}_{b,k}$ is simply the sum of all these function values.

Since we are interested in finding out whether the increased mutation probability of the CC (1+1) EA proves to be beneficial we concentrate on $\text{CLOB}_{b,k}$ with $b > 1$. We always consider the case where the CC (1+1) EA makes complete use of the separability of $\text{CLOB}_{b,k}$. Therefore, the number of components or sub-populations equals the function parameter k . In order to avoid technical difficulties we restrict ourselves to values of k with $k \leq n/4$. This excludes the case $k = n/2$ only, since $k = n$ is only possible with $b = 1$. We start our investigations with an upper bound on the expected optimization time of the CC (1+1) EA.

Theorem 2. The expected optimization time of the CC (1+1) EA on the function $\text{CLOB}_{b,k}: \{0, 1\}^n \rightarrow \mathbb{R}$ is $O(kl^b (\frac{l}{b} + \ln k))$ with $l := n/k$, where the number of components of the CC (1+1) EA is k , and $2 \leq b \leq n/k$, $1 \leq k \leq n/4$, and $n/(bk) \in \mathbb{N}$ hold.

Proof. Since we have $n/(bk) \in \mathbb{N}$ we have k components $x^{(1)}, \dots, x^{(k)}$ of length $l := n/k$ each. In each component the size of the blocks rewarded by $\text{CLOB}_{b,k}$ equals b and there are exactly $l/b \in \mathbb{N}$ such blocks in each component.

We consider the first (1+1) EA operating on $x^{(1)}$. As long as $x^{(1)}$ differs from 1^l , there is always a mutation of at most b specific bits that increases the function

value by at least $n - b$. After at most l/b such mutations $x^{(1)} = 1^l$ holds. The probability of such a mutation is bounded below by $(1/l)^b(1 - 1/l)^{l-b} \geq 1/(el^b)$. We consider $k \cdot 10e \cdot l^b((l/b) + \ln k)$ generations. The first (1+1) EA is active in $10e \cdot l^b((l/b) + \ln k)$ generations. The expected number of such mutations is bounded below by $10((l/b) + \ln k)$. Chernoff bounds yield that the probability not to have at least $(l/b) + \ln k$ such mutations is bounded above by $e^{-4((l/b) + \ln k)} \leq \min\{e^{-4}, k^{-4}\}$. In the case $k = 1$, this immediately implies the claimed bound on the expected optimization time. Otherwise, the probability that there is a component different from 1^l is bounded above by $k \cdot (1/k^4) = 1/k^3$. This again implies the claimed upper bound and completes the proof. $\square$

The expected optimization time $O(kl^b((l/b) + \ln k))$ grows exponentially with b as could be expected. Note, however, that the basis is l , the length of each component. This supports our intuition that the exploitation of the separability together with the increased mutation probability help the CC (1+1) EA to be more efficient on $\text{CLOB}_{b,k}$. We now prove this belief to be correct by presenting a lower bound for the expected optimization time of the (1+1) EA.

Theorem 3. *The expected optimization time of the (1+1) EA on the function $\text{CLOB}_{b,k}: \{0, 1\}^n \rightarrow \mathbb{R}$ is $\Omega(n^b(n/(bk) + \ln k))$, if $2 \leq b \leq n/k$, $1 \leq k \leq n/4$, and $n/(bk) \in \mathbb{N}$ holds.*

Proof. The proof consists of two main steps. First, we prove that with probability at least $1/8$ the (1+1) EA needs to make at least $\lceil k/8 \rceil \cdot l/b$ mutations of b specific bits to find the optimum of $\text{CLOB}_{b,k}$. Second, we estimate the expected waiting time for this number of mutations.

Consider some bit string $x \in \{0, 1\}^n$. It is divided into k pieces of length $l = n/k$ each. Each piece contains l/b blocks of length b . Since each leading block that contains 1-bits only contributes $n - b$ to the function value, these 1-blocks are most important.

Consider one mutation generating an offspring y . Of course, y is divided into pieces and blocks in the same way as x . But the bit values may be different. We distinguish three different types of mutation steps that create y from x . Note that our classification is complete, i. e., no other mutations are possible.

First, the number of leading 1-blocks may be smaller in y than in x . We can ignore such mutations since we have $\text{CLOB}_{b,k}(y) < \text{CLOB}_{b,k}(x)$ in this case. Then y will not replace its parent x .

Second, the number of leading 1-blocks may be the same in x and y . Again, mutations with $\text{CLOB}_{b,k}(y) < \text{CLOB}_{b,k}(x)$ can be ignored. Thus, we are only concerned with the case $\text{CLOB}_{b,k}(y) \geq \text{CLOB}_{b,k}(x)$. Since the number of leading 1-blocks is the same in x and y , the number of 0-bits cannot be smaller in y compared to x . This is due to the OneMax part in $\text{CLOB}_{b,k}$.

Third, the number of 1-blocks may be larger in y than in x . For blocks with at least two 0-bits in x the probability to become a 1-block in y is bounded above by $1/n^2$. We know that the OneMax part of $\text{CLOB}_{b,k}$ leads the (1+1) EA to all zero blocks in $O(n \log n)$ steps. Thus, with probability $O((\log n)/n)$ such steps do not occur before we have a string of the form

$$1^{j_1 \cdot b} 0^{((l/b) - j_1) \cdot b} 1^{j_2 \cdot b} 0^{((l/b) - j_2) \cdot b} \dots 1^{j_k \cdot b} 0^{((l/b) - j_k) \cdot b}$$

as current string of the (1+1) EA.

The probability that we have at least two 0-bits in the first block of a specific piece after random initialization is bounded below by $1/4$. It is easy to see that with probability at least $1/4$ we have at least $\lceil k/8 \rceil$ such pieces after random initialization. This implies that with probability at least $1/8$ we have at least $\lceil k/8 \rceil$ pieces which are of the form 0^l after $O(n \log n)$ generations. This completes the first part of the proof.

Each 0-block can only become a 1-block by a specific mutation of b bits all flipping in one step. Furthermore, only the leftmost 0-block in each piece is available for such a mutation leading to an offspring y that replaces its parent x . Let i be the number of 0-blocks in x . For $i \leq k$, there are up to i blocks available for such mutations. Thus, the probability for such a mutation is bounded above by i/n^b in this case. For $i > k$, there cannot be more than k 0-blocks available for such mutations, since we have at most one leftmost 0-block in each of the k pieces. Thus, for $i > k$, the probability for such a mutation is bounded above by k/n^b . This yields

$$\frac{1}{8} \cdot \left(\sum_{i=1}^k \frac{n^b}{i} + \sum_{i=k+1}^{\lceil k/8 \rceil l/b} \frac{n^b}{k} \right) \geq \frac{n^b}{8} \cdot \left(\ln k + \frac{kl}{8bk} \right) = \Omega \left(n^b \cdot \left(\frac{n}{bk} + \log n \right) \right)$$

as lower bound on the expected optimization. $\square$

We want to see the benefits the increased mutation probability due to the cooperative coevolutionary approach can cause. Thus, our interest is not specifically concentrated on the concrete expected optimization times of the (1+1) EA and the CC (1+1) EA on $\text{CLOB}_{b,k}$. We are much more interested in a comparison. When comparing (expected) run times of two algorithms solving the same problem it is most often sensible to consider the ratio of the two (expected) run times. Therefore, we consider the expected optimization time of the (1+1) EA divided by the expected optimization time of the CC (1+1) EA. We see that

$$\frac{\Omega \left(n^b \cdot \left(\frac{n}{bk} + \log n \right) \right)}{O \left(l^b ((l/b) + \log k) \right)} = \Omega \left(k^{b-1} \right)$$

holds. We can say that the CC (1+1) EA has an advantage of order at least k^{b-1} . The parameter b is a parameter of the problem. In our special setting, this holds for k , too, since we divide the problem as much as possible. Using c components, where $c \leq k$, would reveal that this parameter c influences the advantage of the CC (1+1) EA in a way k does in the expression above. Obviously, c is a parameter of the algorithm. Choosing c as large as the objective function $\text{CLOB}_{b,k}$ allows yields the best result. This confirms our intuition that the separability of the problem should be exploited as much as possible. We see that for some values of k and b this can decrease the expected optimization time from super-polynomial for the (1+1) EA to polynomial for the CC (1+1) EA. This is, for example, the case for $k = n(\log \log n)/(2 \log n)$ and $b = (\log n)/\log \log n$.

It should be clear that simply increasing the mutation probability in the (1+1) EA will not resolve the difference. Increased mutation probabilities lead to a larger number of steps where the offspring y does not replace its parents x , since the number of leading ones blocks is decreased due to mutations. As a result, the CC (1+1) EA gains clear advantage over the (1+1) EA on this $\text{CLOB}_{b,k}$ class of functions. Moreover, this advantage is drawn from more than a simple partitioning of the problem. The advantage stems from the coevolutionary algorithm's ability to increase the focus of attention of the mutation operator, while using the partitioning mechanism to protect the remaining components from the increased disruption.

5 Conclusion

We investigated a quite general cooperative coevolutionary function optimization framework that was introduced by Potter and De Jong (7). One feature of this framework is that it can be instantiated using any evolutionary algorithm as underlying search heuristic. We used the well-known (1+1) EA and presented the CC (1+1) EA, an extremely simple cooperative coevolutionary algorithm. The main advantage of the (1+1) EA is the multitude of known results and powerful analytical tools. This enabled us to present the run time or optimization time analysis for a coevolutionary algorithm. To our knowledge, this is the first such analysis of coevolution published. The focus of our investigation was on separability. Indeed, when applying the Potter and De Jong 7 cooperative coevolutionary approach, practitioners make implicit assumptions about the separability of the function in order to come up with appropriate divisions of the problem space. Given such a static partition of a string into components, the CCEA is expected to exploit the separability of the problem and to gain an advantage over the employed EA when used alone. We were able to prove that separability alone is not sufficient to give the CCEA any advantage. We compared the expected optimization time of the (1+1) EA with that of the CC (1+1) EA on linear functions that are of maximal separability. We found that the CC (1+1) EA is not faster. Motivated by this finding we discussed the expected frequency of mutations for both algorithms. The main point is that b bit mutations occur noticeably more often for the CC (1+1) EA for $b > 1$ only. The expected frequency of mutations changing only one single bit is asymptotically the same for both algorithms. This leads to the definition of $\text{CLOB}_{b,k}$, a family of separable functions where b bit mutations are needed for successful optimization. For this family of functions we were able to prove that the cooperative coevolutionary approach leads to an immense speed-up. The advantage of the CC (1+1) EA over the (1+1) EA can be of super-polynomial order. Moreover, this advantage stems not only from the ability of the CC (1+1) EA to partition the problem, but because coevolution can use this partitioning to concentrate increased variation on smaller parts of the problem.

Our results are a first and important step towards a clearer understanding of coevolutionary algorithms. But there are a lot of open problems. An upper bound for the expected optimization time of the CC (1+1) EA on linear functions

needs to be proven. Using standard arguments the bound $O(n \log^2 n)$ is easy to show; however, we conjecture that the actual expected optimization time is $O(n \log n)$ for any linear function and $\Theta(n \log n)$ for linear functions without zero weights. For $\text{CLOB}_{b,k}$ we provided neither a lower bound proof of the expected optimization time of the CC (1+1) EA nor an upper bound proof of the expected optimization time of the (1+1) EA. A lower bound for the CC (1+1) EA that is asymptotically tight is not difficult to prove. A good upper bound for the (1+1) EA is slightly more difficult.

Furthermore, it is obviously desirable to have more comparisons for more general parameter settings and other objective functions. The systematic investigation of the effects of running the CC (1+1) EA with partitions into components that do not match the separability of the objective function is also the subject of future research. A main point of interest is the analysis of other cooperative coevolutionary algorithms where more complex EAs that use a population and crossover are employed as underlying search heuristics. The investigation of such CCEAs that are more realistic leads to new, interesting, and much more challenging problems for future research.

References

- S. Droste, T. Jansen, G. Rudolph, H.-P. Schwefel, K. Tinnefeld, and I. Wegener (2003). Theory of evolutionary algorithms and genetic programming. In H.-P. Schwefel, I. Wegener, and K. Weinert (Eds.), *Advances in Computational Intelligence*, Berlin, Germany, 107–144. Springer.
- S. Droste, T. Jansen, and I. Wegener (2002). On the analysis of the (1+1) evolutionary algorithm. *Theoretical Computer Science* 276, 51–81.
- J. Garnier, L. Kallel, and M. Schoenauer (1999). Rigorous hitting times for binary mutations. *Evolutionary Computation* 7(2), 173–203.
- A. Iorio and X. Li (2002). Parameter control within a co-operative co-evolutionary genetic algorithm. In J. J. Merelo Guervós, P. Adamidis, H.-G. Beyer, J.-L. Fernández-Villacañás, and H.-P. Schwefel (Eds.), *Proceedings of the Seventh Conference on Parallel Problem Solving From Nature (PPSN VII)*, Berlin, Germany, 247–256. Springer.
- R. Motwani and P. Raghavan (1995). *Randomized Algorithms*. Cambridge: Cambridge University Press.
- H. Mühlenbein (1992). How genetic algorithms really work. Mutation and hillclimbing. In R. Männer and R. Manderick (Eds.), *Proceedings of the Second Conference on Parallel Problem Solving from Nature (PPSN II)*, Amsterdam, The Netherlands, 15–25. North-Holland.
- M. A. Potter and K. A. De Jong (1994). A cooperative coevolutionary approach to function optimization. In Y. Davidor, H.-P. Schwefel, and R. Männer (Eds.), *Proceedings of the Third Conference on Parallel Problem Solving From Nature (PPSN III)*, Berlin, Germany, 249–257. Springer.
- M. A. Potter and K. A. De Jong (2002). Cooperative coevolution: An architecture for evolving coadapted subcomponents. *Evolutionary Computation* 8(1), 1–29.
- G. Rudolph (1997). *Convergence Properties of Evolutionary Algorithms*. Hamburg, Germany: Dr. Kovač.
- E. van Nimwegen and J. P. Crutchfield (2001). Optimizing epochal evolutionary search: Population-size dependent theory. *Machine Learning* 45(1), 77–114.

PalmPrints: A Novel Co-evolutionary Algorithm for Clustering Finger Images

Nawwaf Kharma, Ching Y. Suen, and Pei F. Guo

Departments of Electrical & Computer Engineering and Computer Science,
Concordia University, 1455 de Maisonneuve Blvd. West, Montreal, QC, H3G 1M8, Canada
kharma@ece.concordia.ca

Abstract. The purpose of this study is to explore an alternative means of hand image classification, one that requires minimal human intervention. The main tool for accomplishing this is a Genetic Algorithm (GA). This study is more than just another GA application; it introduces (a) a novel cooperative co-evolutionary clustering algorithm with dynamic clustering and feature selection; (b) an extended fitness function, which is particularly suited to an integrated dynamic clustering space. Despite its complexity, the results of this study are clear: the GA evolved an average clustering of 4 clusters, with minimal overlap between them.

1 Introduction

Biometric approaches to identity verification offer a mostly *convenient* and potentially *effective* means of personal identification. All such techniques, whether palm-based or not, rely on the individual's most-unique and stable, physical or behavioural characteristics.

The use of multiple sets of features requires feature selection as a prerequisite for the subsequent application of classification or clustering [5, 8]. In [5], a hybrid genetic algorithm (GA) for feature selection resulted in (a) better convergence properties; (b) significant improvement in terms of final performance; and (c) the acquisition of subset-size feature control. Again, in [8], a GA, in combination with a k-nearest neighbour classifier, was successfully employed in feature dimensionality reduction.

Clustering is the grouping of similar objects (e.g. hand images) together in one set. It is an important unsupervised classification technique. The simplest and most well known clustering algorithm is the k-means algorithm. However, this algorithm requires that the user specifies, before hand, the desired number of clusters. An evolutionary strategy implementing variable length clustering in the x-y plane was developed to address the problem of dynamic clustering [3]. Additionally, a genetic clustering algorithm was used to determine the best number of clusters, while simultaneously clustering objects [9].

Genetic algorithms are randomized search and optimization techniques guided by the principles of evolution and natural genetics, and offering a large amount of implicit parallelism. GAs perform search in complex, large and multi-modal landscapes. They have been used to provide (near-)optimal solutions to many optimization problems [4].

Cooperative co-evolution refers to the simultaneous evolution of two or more species with coupled fitness. Such evolution allows the discovery of complex solutions wherever complex solutions are needed. The fitness of an individual depends on its ability to collaborate with individuals from other species. In this way, the evolutionary pressure stemming from the difficulty of the problem favours the development of cooperative individual strategies [7].

In this paper, we propose a cooperative co-evolutionary clustering algorithm, which integrates dynamic clustering, with (hand-based) feature selection. The co-evolutionary part is defined as the problem of partitioning a set of hand objects into a number of clusters without *a priori* knowledge of the feature space. The paper is organized as follows. In section 2, hand feature extraction is described. In section 3, cooperative co-evolutionary clustering and feature selection are presented, along with implementation results. Finally, the conclusions are presented in section 4.

2 Feature Extraction

Hand geometry refers to the geometric structure of the hand. Shape analysis requires the extraction of object features, often normalized, and invariant to various geometric transformations such as translation, rotation and (to a lesser degree) scaling. The features used may be divided into two sets: geometric features and statistical features.

2.1 Geometric Features

The geometrical features measured can be divided into six categories:

- Finger Width(s): the distance between the minima of the two phalanges at either side of a finger. The line connecting those two phalanges is termed the finger base-line.
- Finger Height(s): the length of the line starting at the fingertip and intersecting (at right angles) with the finger base-line.
- Finger Circumference(s): The length of the finger contour.
- Finger Angle(s): The two acute angles made between the finger base-line and the two lines connecting the phalange minima with the finger tip.
- Finger Base Length(s): The length of the finger base-lines.
- Palm Aspect Ratio: the ratio of the 'palm width' to the 'palm height'. Palm width is (double) the distance between the phalange joint of the middle finger, and the midpoint of the line connecting the outer points of the base lines of the thumb and pinkie (call it *mp*). Palm length is (double) the shortest distance between *mp* and the right edge of the palm image.

2.2 Statistical Features

Before any statistical features are measured, the fingers are re-oriented (see Fig. 1), such that they are standing upright by using the Rotation and Shifting of the Coordinate Systems. Then, each 2D finger contour is mapped onto a 1D contour (see Fig. 2), taking the finger midpoint centre as its reference point. The shape analysis for four fingers (excluding the thumb) is measured using: (1) Central moments; (2) Fourier descriptors; (3) Zernike moments.

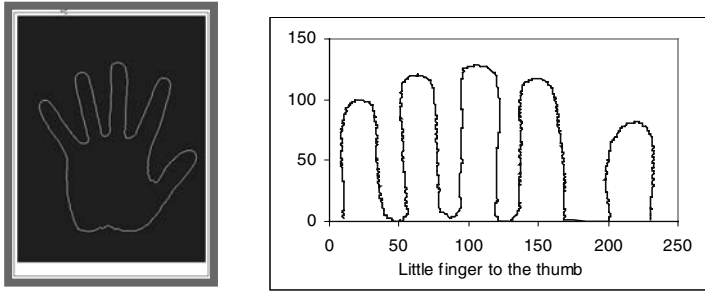


Fig. 1. Hand Fingers (vertically re-oriented) using the Rotation and Shifting of the Coordinate Systems

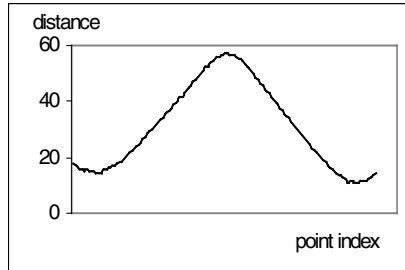


Fig. 2. 1D Contour of a Finger. The y-axis represents the Euclidean distance between the contour point and the finger midpoint centre (called the reference point)

Central Moments. For a digital image, the p_{th} order regular moment with respect to a one-dimensional function $F[n]$ is defined as:

$$R_p = \sum_{n=0}^N n^p \cdot F[n]$$

The normalized one-dimensional p_{th} order central moments are defined as:

$$M_p = \sum_{n=0}^N (n - \bar{n})^p \cdot F[n] \quad \bar{n} = R_1 / R_0$$

$F[n]$: with $n \in [0, N]$; the Euclidean distance between point n and the finger reference point.

N : the total number of pixels.

Fourier Descriptors. We define a normalized cumulative function Φ^* as an expanding Fourier series to obtain descriptive coefficients (Fourier Descriptors or FD's). Given a periodic 1D digital function $F[n]$ in $[0, N]$ points (periodic), the expanding Fourier series is:

$$\Phi^*(t) = \frac{a_0}{2} + \sum_{k=1}^{\infty} \left(a_k \cos \frac{2\pi k}{N} \cdot t + b_k \sin \frac{2\pi k}{N} \cdot t \right)$$

$$a_k = \frac{2}{N} \sum_{n=1}^N F[n] \cdot \cos \frac{2\pi k}{N} \cdot n, \quad b_k = \frac{2}{N} \sum_{n=1}^N F[n] \cdot \sin \frac{2\pi k}{N} \cdot n$$

The k_{th} harmonic amplitudes of the Fourier Descriptors are:

$$A_k = \sqrt{a_k^2 + b_k^2} \quad k = 1, 2, \dots$$

Zernike Moments. For a digital image with a polar form function $f(\rho, \varphi)$, the normalized $(n+m)_{th}$ order Zernike moments is approximated by:

$$Z_{nm} \approx \frac{n+1}{N} \sum_j f(\rho_j, \varphi_j) \cdot V_{nm}^*(\rho_j, \varphi_j)$$

$$V_{nm}(\rho, \varphi) = R_{nm}(\rho) \cdot e^{jm\varphi}, \quad x_j^2 + y_j^2 \leq 1$$

$$R_{nm}(\rho) = \sum_{s=0}^{(n-|m|/2)} \frac{(-1)^s (n-s)! \rho^{n-2s}}{s! ((n+|m|)/2-s)! ((n-|m|)/2-s)!}$$

n : a positive integer.

m : a positive or negative integer subject to the constraints that $n-|m|$ is even, $|m| \leq n$.

$f(\rho_j, \varphi_j)$: the length of vector between point j and the finger reference point.

3 Co-evolution in Dynamic Clustering and Feature Selection

Our clustering application involves the optimization of three quantities, which together form a complete solution, (1) the set of features (dimensions) used for clustering; (2) the actual cluster centres; and (3) the total number of clusters. Since this is the case, and since the relationship between the three quantities is complementary (as opposed to adversarial), it makes sense to use cooperative (as

opposed to competitive) co-evolution as the model for the overall genetic optimization process. Indeed, it is our hypothesis that whenever a (complete) potential solution (i) is comprised of a number of complementary components; (ii) has a medium-high degree of dimensionality; and (iii) features a relatively low level of coupling between the various components; then attempting a cooperative co-evolutionary approach is justified.

In similarity-based clustering techniques, a number of cluster centres are proposed. An input pattern (point) is assigned to the cluster whose centre is closest to the point. After all the points are assigned to clusters, the cluster centres are re-computed. Then, the points are re-assigned to the (new) clusters based (again) on their distance from the new cluster centres. This process is iterative, and hence it continues until the locations of the cluster centres stabilize.

During co-evolutionary clustering, the above occurs, but in addition, less discriminatory features are eliminated, leaving a more efficient subset for use. As a result, the overall output of the genetic optimization process is a number of traditionally good (i.e. tight and well-separated) clusters, which also exist in the smallest possible feature space.

The co-evolutionary genetic algorithm used entails that we have two populations (one of cluster centres and another of dimension selections: more on this below), each going through a typical GA process. This process is iterative and follows these steps: (a) fitness evaluation; (b) selection; (c) the application of crossover and mutation (to generate the next population); (d) convergence testing (to decide whether to exit or not); (e) back to (a).

This continues until the convergence test is satisfied and the process is stopped. The GA process is applied to the first population and in parallel (but totally independently) to the second population. The only difference between a GA applied to one (evolving population) and a GA applied to two cooperatively co-evolving populations is that fitness evaluation of an individual in one population is done after that individual is joined to another individual in the other population. Hence, the fitness of individuals in one population is actually coupled with (and is evaluated with the help of) individuals in the other population.

Below, is a description of the most important aspects of the genetic algorithm applied to the co-evolving populations that make-up PalmPrints. First, the way individuals are represented (as chromosomes) is described. This is followed by an explanation of step (a) to step (e), listed above. Finally, a discussion of the results is presented.

3.1 Chromosomal Representation

In any co-evolutionary genetic algorithm, two (or more) populations co-evolve. In our case, there are only two populations, (a) a population of cluster centres (*Cpop*), each represented by a variable-length vector of real numbers; and (b) a population of 'dimension-selections', or simply dimensions (*Dpop*), each represented by a vector of bits. Each individual in *Cpop* represents a (whole) number of cluster centre coordinates. The total number of coordinates equals the number of clusters. On the other hand, each individual ('dimension-selection') in *Dpop* indicates, via its '1' bits,

which dimensions will be used and which, via its '0' bits, will not be used. Splicing an individual (or chromosome) from *Cpop* with an individual (or chromosome) from *Dpop* will give us an overall chromosome that has the following form:

$$\{(A1, B1, \dots, Z1), (A2, B2, \dots, Z2), \dots (An, Bn, \dots, Zn), 10110\dots0\}$$

Taken as a single representational unit, this chromosome determines:

- (1) The *number of clusters*, via the number of cluster centres in the left-hand side of the chromosome;
- (2) The actual *cluster centres*, via the coordinates of cluster centres, also presented in the left-hand side of the chromosome; and
- (3) The *number of dimensions* (or features) used to represent the cluster centres, via the bit vector on the right-hand side of the chromosome.

As an example, the chromosome presented above has n clusters in three dimensions: the first, third and fourth dimensions. (This is so because the bit vector has 1 in its first bit location, 1 in its third bit location and 1 in its fourth bit location.) The maximum number of feature dimensions (allowed in this example) is equal to the number of letters in the English alphabet: 26, while the minimum is 1. And, the maximum number of clusters (which is not shown) is $m > n$.

3.2 Crossover and Mutation, Generally

In our approach, the crossover operators need to (a) deal with varying-length chromosomes; (b) allow for a varying number of feature dimensions; (c) allow for a varying number of clusters; and (d) to be able to adjust the values of the coordinates of the various cluster centres. This is not a trivial task, and is achieved via a host of crossover operators, each tuned for its own task. This is explained below.

Crossover and Mutation for Cpop. *Cpop* needs crossover and mutation operators suited for variable-length clusters as well as real-valued parameters. When crossing over two parent chromosomes to produce two new child chromosomes, the algorithm follows a three-step procedure:

- (a) The length of a child chromosome is randomly selected from the range: $[2, MaxLength]$, where *MaxLength* is equal to the total number of clusters in both parent chromosomes;
- (b) Each child chromosome picks up copies of cluster centre coordinates, from each of the two parents, in proportion to the relative fitness of the parents (to each other); and finally,
- (c) The actual values of the cluster coordinates are modified using the following (mutation) formula for i th feature with α randomly selected from the range $[0,1]$:

$$f_i = \min(F_i) + \alpha [\max(F_i) - \min(F_i)] . \quad (1)$$

F_i : the i th feature dimension, $i = 0, 1, 2, \dots$

α : a random value ranged $[0,1]$.

$\min(f_i) / \max(f_i)$: minimum / maximum value that feature i can take.

With α changed within $[0,1]$, the function of equation (1) varies the i th feature dimension in its own distinguished feature range $[\min(F_i), \max(F_i)]$ as for the variation of actual values of the cluster coordinates (see Fig. 3).

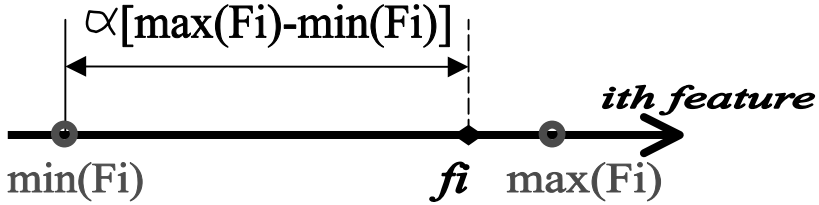


Fig. 3. Variation of the i th feature dimension within $[\min(F_i), \max(F_i)]$ with a random value α ranged $[0,1]$

In addition to crossover, mutation is applied, with a probability μ_c to one set of cluster centre coordinates. The value of μ_c used is 0.2 (or 20%).

Crossover and Mutation for Dpop. *Dpop* needs one crossover operator suited for fixed length binary-valued parameters. For a binary representation of *Dpop* chromosomes, single-point crossover is applied. Following that, mutation is applied with a mutation rate of μ_d . The value of μ_d used is 0.02.

3.3 Selection and Generation of Future Generations

For both populations, elitism is applied first, and causes copies of the fittest chromosomes to be carried over (without change) from the current generation to the next generation. Elitism is set at 12% of *Cpop* and 10% of *Dpop*. Another 12% of *Cpop* and 10% of *Dpop* are generated via the crossing over of pairs of elite individuals, to generate an equal number of children. The rest (76% of *Cpop* and 80% of *Dpop*) of the next generation is generated through the application of crossover and mutation (in that order) to randomly selected individuals from the non-elite part of the current generation. Crossover is applied with a probability of 1 (i.e. all selected individuals are crossed over), while mutation is applied with a probability of 20% for *Cpop* and 2% for *Dpop*.

3.4 Fitness Function

Since the Mean Square Error (MSE) can always be decreased by adding a data point as a cluster centre, fitness was a monotonically decreasing function of cluster numbers. The fitness function (MSE) was poorly suited for comparing clustering situations that had a different numbers of clusters. A heuristic MSE was chosen with dynamic cluster n , based on the one given by [3].

In our own approach of dynamic clustering with feature selection in a co-evolutionary GA, there are two dynamic variables interchanged with the two populations: dynamic clustering and dynamic feature dimensions. Hence, a new *extended* MSE fitness is proposed for our model, which measures quantities of both object tightness (f_T) and cluster separation (f_S):

$$MSE \text{ extended fitness} = \sqrt{n+1} \left(f_T + \frac{1}{f_S} \right)$$

$$f_T = \sum_{i=1}^n \sum_{j=1}^{mi} d(c_i, x_j^i) / n, \quad f_S = \sqrt{k+1} \sum_{i=1}^n d\{c_i, \text{Ave}(\sum_{j=1, j \neq i}^n c_j)\}$$

n : dynamic no. of clusters

k : dynamic no. of features

c_i : the i th cluster centre

$\text{Ave}(A)$: the average value of A

mi : the number of data points belonging to the i_{th} cluster

x_j^i : the j_{th} data point belonging to the i_{th} cluster

$d(a,b)$: the Euclidean distance between points a and b

The square root of the number of clusters and the square root of the number of dimension in *MSE extended fitness* are chosen to be unbiased in the dynamic co-evolutionary environment. The point of the MSE extended fitness is to optimize of the distance criterion by minimizing the within-cluster spread and maximizing the inter-cluster separation.

3.5 Convergence Testing

The number of generations prior to termination depends on whether an acceptable solution is reached or a set number of iterations are exceeded. Most genetic algorithms keep track of the population statistics in the form of population maximum and mean fitness, standard deviation of (maximum or mean) fitness, and minimum cost. Any of these or any combination of these can serve as a convergence test. In PalmPrints, we stop the GA when the maximum fitness does not change by more than .001 for 10 consecutive generations.

3.6 Implementation Results

The *Dpop* population is initialized with 500 members, from which 50 parents were paired from top to bottom. The remaining 400 offspring are produced randomly using

single-point crossover and a mutation rate (μ_d) of 0.02. *Cpop* is initialized at 88 individuals, from which 10 members are selected to produce 10 direct new copies in the next generation. The remaining 68 are generated randomly, using the dimension fine-tuning crossover strategy and a mutation rate (μ_c) of 0.2.

The experiment presented here uses 100 hand images and 84 normalized features. Termination occurred at a maximum of 250 generations, since it is discovered that fitness converged to less than 0.0001 variance prior. The results are promising; the average co-evolutionary clustering fitness is 0.9912 with a significantly low standard deviation of 0.1108. The average number of clusters is 4, with a very low standard deviation of 0.4714. Average hand image misplacement rate is 0.0580, with a low standard deviation of 2.044. Following convergence, the dimension of the feature space is 41, with zero standard deviation. Hence, half of the original 84 features are eliminated. Convergence results are shown in Fig. 4.

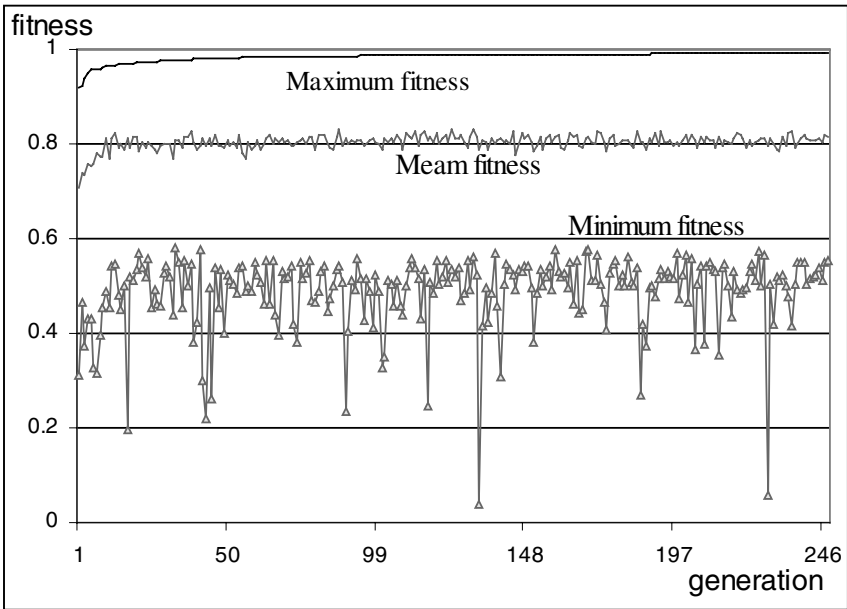


Fig. 4. Convergence results

4 Conclusions

This study is the first to use a genetic algorithm to simultaneously achieve dimensionality reduction and object (hand image) clustering. In order to do this, a cooperative co-evolutionary GA is crafted, one that uses two populations of part-solutions in order to evolve complete highly fit solutions for the whole problem. It does succeed in both its objectives. The results show that the dimensionality of the clustering space is cut in half. The number (4) and quality (0.058) of clusters

produced are also very good. These results open the way towards other cooperative co-evolutionary applications, in which 3 or *more* populations are used to co-evolve solutions and designs consisting of 3 or more loosely-coupled sub-solutions or modules.

In addition to the main contribution of this study, the authors introduce a number of new or modified structural (e.g. palm aspect ratio) and statistical features (e.g. finger 1D contour transformation) that may prove equally useful to others working on the development of biometric-based technologies.

References

1. Fogel, D.B.: Evolutionary Computation: Toward A New Philosophy Of Machine Intelligence. IEEE Press, New York, (1995) 5
2. Haupt, R.L. and Haupt, S.E.: Practical Genetic Algorithms. Wiley Interscience, New York (1998)
3. Lee, C.-Y.: Efficient Automatic Engineering Design Synthesis Via Evolutionary Exploration. PhD thesis (2002), California Institute of Technology, Pasadena, California
4. Maulik, U., Bandyopadhyay S.: Genetic Algorithm-based Clustering Technique. Pattern Recognition 33 (2000) 1455–1465
5. Oh, I.-S., Lee, J.-S. and Moon, B.-R.: Local Search-embedded Genetic Algorithms For Feature Selection. Proc. of International Conf. on Pattern Recognition (2002) 148–151
6. Paredis, J.: Coevolutionary Computation. Artificial Life 2 (1995) 355–375
7. Pena-Reyes, C.A., Sipper M.: Fuzzy CoCo: A Cooperative-Coevolutionary Approach To Fuzzy Modeling. IEEE Transaction on Fuzzy Systems Vol. 9, No.5 (October 2001) 727–737
8. Raymer, M.L., Punch, W.F., Goodman, E.D., Kuhn, L.A. and Jain, A.K.: Dimensionality Reduction Using Genetic Algorithms. IEEE Transaction on Evolutionary Computation, Vol. 4, No.2 (July 2000) 164–171
9. Tseng, L.Y., Yang, S.B.: A Genetic Approach To The Automatic Clustering Problem. Pattern Recognition 34 (2001) 415–424

Coevolution and Linear Genetic Programming for Visual Learning

Krzysztof Krawiec* and Bir Bhanu

Center for Research in Intelligent Systems
University of California, Riverside, CA 92521-0425, USA
{kkrawiec, bhanu}@cris.ucr.edu

Abstract. In this paper, a novel genetically-inspired visual learning method is proposed. Given the training images, this general approach induces a sophisticated feature-based recognition system, by using cooperative coevolution and linear genetic programming for the procedural representation of feature extraction agents. The paper describes the learning algorithm and provides a firm rationale for its design. An extensive experimental evaluation, on the demanding real-world task of object recognition in synthetic aperture radar (SAR) imagery, shows the competitiveness of the proposed approach with human-designed recognition systems.

1 Introduction

Most real-world learning tasks concerning visual information processing are inherently complex. This complexity results not only from the large volume of data that one usually needs to process, but also from its spatial nature, information incompleteness, and, most of all, from the vast number of hypotheses that have to be considered in the learning process and the ‘ruggedness’ of the fitness landscape. Therefore, the design of a visual learning algorithm mostly consists in *modeling* its capabilities so that it is effective in solving the problem. To induce useful hypotheses on one hand and avoid overfitting to the training data on the other, some assumptions have to be made, concerning training data and hypothesis representation, known as *inductive bias* and *representation bias*, respectively. In visual learning, these biases have to be augmented by an extra ‘*visual bias*’, i.e., knowledge related to the visual nature of the information being subject to the learning process. A part of that is general knowledge concerning vision (*background knowledge*, BK), for instance, basic concepts like pixel proximity, edges, regions, primitive features, etc. However, usually a more specific *domain knowledge* (DK) related to a particular *task/application* (e.g., fingerprint identification, face recognition, etc.) is also required.

Currently, most recognition methods make intense use of DK to attain a competitive performance level. This is, however, a double-edged sword, as the more DK the method uses, the more specific it becomes and the less general and

* On a temporary leave from Institute of Computing Science, Poznań University of Technology, Poznań, Poland.

transferable is the knowledge it acquires. The contribution of such over-specific methods to the overall body of knowledge is questionable.

Therefore, in this paper, we propose a general-purpose visual learning method that requires only BK and produces a complete recognition system that is able to classify objects in images. To cope with the complexity of the recognition task, we break it down into components. However, the ability to identify building blocks is a necessary, but not a sufficient, precondition for a successful learning task. To enforce learning in each identified component, we need an *evaluation function* that spans over the space of all potential solutions and guides the learning process. Unfortunately, when no *a priori* definition of module's 'desired output' is available, this requirement is hard to meet. This is why we propose to employ here cooperative coevolution [10], as it does not require the explicit specification of objectives for each component.

2 Related Work and Contributions

No general methodology has been developed so far that effectively automates the visual learning process. Several methods have been reported in the literature; they include blackboard architecture, case-based reasoning, reinforcement learning, and automatic acquisition of models, to mention the most predominant. The paradigm of evolutionary computation (EC) has also found applications in image processing and analysis. It has been found effective for its ability to perform global parallel search in high-dimensional search spaces and to resist the local optima problem. However, in most approaches the learning is limited to parameter optimization. Relatively few results have been reported [5,8,13,14], that perform visual learning in the deep sense, i.e., with a learner being able to synthesize and manipulate an entire recognition system.

The major contribution of this paper is a general method that, given only a set of training images, performs visual learning and yields a complete feature-based recognition system. Its novelty consists mostly in (i) *procedural representation* of features for recognition, (ii) utilization of *coevolutionary computation* for induction of image representation, and (iii) *a learning process* that optimizes the image feature definitions, prior to classifier induction.

3 Coevolutionary Construction of Feature Extraction Procedures

We pose visual learning as the search of the space of image representations (sets of features). For this purpose, we propose to use *cooperative coevolution* (CC) [10], which, besides being appealing from the theoretical viewpoint, has been reported to yield interesting results in some experiments [15]. In CC, one maintains many populations, with individuals in populations encoding only a *part* of the solution to the problem. To undergo evaluation, individuals have to be (temporarily) combined with individuals from the remaining populations to form an *organism* (solution). This joint evaluation scheme forces the populations to cooperate. Except for this evaluation step, other steps of evolutionary algorithm proceed in each population independently.

According to Wolpert's 'No Free Lunch' theorem [17], the choice of this particular search method is irrelevant, as the average performance of any metaheuristic search over a set of all possible fitness functions is the same. In the real world, however, not all fitness functions are equally probable. Most real-world problems are characterized by some features that make them specific. The practical utility of a search/learning algorithm depends, therefore, on its ability to detect and benefit from those features.

The *high complexity* and *decomposable nature* of the visual learning task are such features. Cooperative coevolution seems to fit them well, as it provides the possibility of breaking up a complex problem into components *without specifying explicitly the objectives for them*. The manner in which the individuals from populations cooperate *emerges* as the evolution proceeds. In our opinion, this makes CC especially appealing to the problem of visual learning, where the overall object recognition task is well defined, but there is no *a priori* knowledge about what should be expected at intermediate stages of processing, or such knowledge requires an extra effort from the designer.

In [3], we provide experimental evidence for the superiority of CC-based feature construction over standard EC approach in the standard machine learning setting; here, we extend this idea to visual learning. Following the *feature-based recognition* paradigm, we split the object recognition process into two modules: *feature extraction* and *decision making*. The algorithm learns from a finite training set of examples (images) D in a *supervised* manner, i.e. requires D to be partitioned into finite number of pairwise disjoint decision classes D_r .

In the coevolutionary run, n populations cooperate in the task of building the complete image *representation*, with each population responsible for evolving one component. Therefore, the cooperation here may be characterized as taking place at the *feature level*. In particular, each individual I from a given population encodes a single *feature extraction procedure*. For clarity, details of this encoding are provided in Section 4.

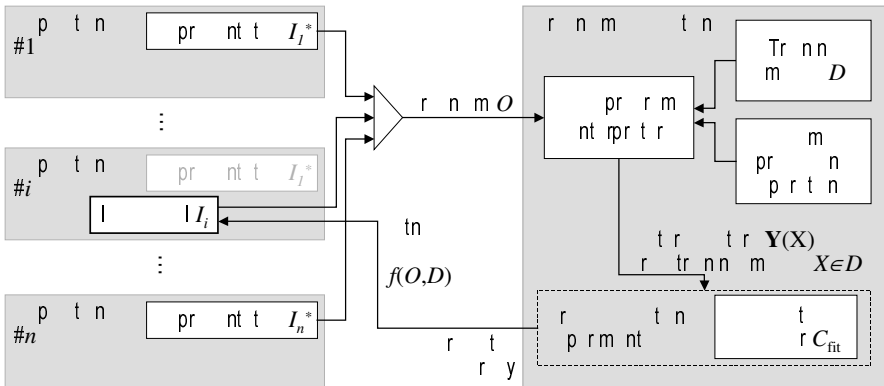


Fig. 1. The evaluation of an individual I_i from i^{th} population.

The coevolutionary search proceeds in all populations independently, except for the evaluation phase, shown in Fig. 1. To evaluate an individual I_j from population $\#j$, we first provide for the remaining part of the representation. For this purpose,

representatives I_i^* are selected from all the remaining populations $i \neq j$. A representative I_i^* of i^{th} population is defined here in a way that has been reported to work best [15]: it is the best individual w.r.t. the previous evaluation. In the first generation of evolutionary run, since no prior evaluation data is given, it is a randomly chosen individual.

Subsequently, I_j is temporarily combined with representatives of all the remaining populations to form an organism

$$O = \langle I_1^*, \dots, I_{j-1}^*, I_j, I_{j+1}^*, \dots, I_n^* \rangle. \quad (1)$$

Then, the feature extraction procedures encoded by individuals from O are ‘run’ (see Section 4) for all images X from the training set D . The feature values $\mathbf{y}$ computed by them are concatenated, building the compound feature vector $\mathbf{Y}$:

$$\mathbf{Y}(X) = \langle \mathbf{y}(I_1^*, X), \dots, \mathbf{y}(I_{j-1}^*, X), \mathbf{y}(I_j, X), \mathbf{y}(I_{j+1}^*, X), \dots, \mathbf{y}(I_n^*, X) \rangle. \quad (2)$$

Feature vectors $\mathbf{Y}(X)$, computed for all training images $X \in D$, together with the images’ decision class labels constitute the dataset:

$$\{ \langle \mathbf{Y}(X), i \rangle : \forall X \in D_i, \forall D_i \} \quad (3)$$

Finally, cross-validation, i.e. multiple train-and-test procedure is carried out on these data. For the sake of speed, we use here a fast classifier C_{fit} that is usually much simpler than the classifier used in the final recognition system. The resulting predictive recognition ratio (see equation 4) becomes the evaluation of the organism O , which is subsequently assigned as the fitness value to $f()$ the individual I_j , concluding its evaluation process:

$$\begin{aligned} f(I_j, D) &= f(O, D) = \\ &= \text{card} \left(\left\{ \langle \mathbf{Y}(X), i \rangle, \forall X \in D_i \wedge C(\mathbf{Y}(X)) = i, \forall D_i, \right\} \right) / \text{card}(D) \end{aligned} \quad (4)$$

where $\text{card}()$ denotes cardinality of a set. Using this evaluation procedure, the coevolutionary search proceeds until some stopping criterion (usually considering computation time) is met. The final outcome of the coevolutionary run is the best found organism/representation O^* .

4 Representation of Feature Extraction Procedures

For representing the feature extraction procedures as individuals in the evolutionary process, we adopt a variety of Linear Genetic Programming (LGP) [1], a hybrid of genetic algorithms (GA) and genetic programming (GP). The individual’s genome is a fixed-length string of bytes, representing a sequential program composed of (possibly parameterized) basic *operations* that work on images and scalar data. This representation combines advantages of both GP and GA, being both procedural and more resistant to the destructive effect of crossover that may occur in ‘regular’ GP [1].

A feature extraction procedure accepts an image X as input and yields a vector $\mathbf{y}$ of scalar values as the result. Its operations are effectively calls to image processing and feature extraction functions. They work on *registers*, and may use them for both input as well as output arguments. *Image registers* store processed images, whereas *real-number registers* keep intermediate scalar results features. Each image register has single channel (grayscale), the same dimensions as the input image X , and maintains a rectangular *mask* that, when used by an operation, limits the processing to its area. For simplicity, the numbers of both types of registers are controlled by the same parameter m .

Each chunk of four consecutive bytes in the genome encodes a single operation with the following components:

- (a) operation code,
- (b) mask flag – decides whether the operation should be global (work on the entire image) or local (limited to the mask),
- (c) mask dimensions (ignored if the mask flag is ‘off’),
- (d) arguments: references to registers to fetch input data and store the result.

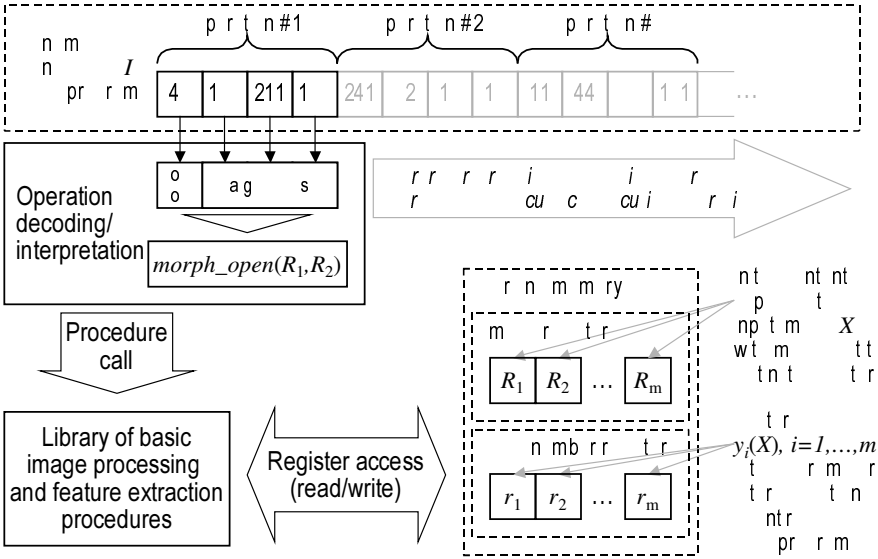


Fig. 2. Execution of LGP code contained in individual’s I genome (for a single image X).

Fig. 2 shows the execution at the moment of executing the following operation: morphological opening (a), applied locally (b) to the mask of size 14×14 (c) to the image fetched from image register pointed by argument #1, and storing the result in image register pointed by argument #2 (d). There are currently 70 operations implemented in the system. They mostly consist of calls to functions from Intel Image Processing and OpenCV libraries, and encompass image processing, mask-related operations, feature extraction, and arithmetic and logic operations.

The processing of a single input image $X \in D$ by the LGP procedure encoded in an individual I proceeds as follows (Fig. 2):

1. *Initialization*: Each of the m image registers is set to X . The masks of images are set to the m most distinctive local features (here: bright ‘blobs’) found in the image. Real-number registers are set to the center coordinates of corresponding masks.
2. *Execution*: the operations encoded by I are carried out one by one, with intermediate results stored in registers.
3. *Interpretation*: the scalar values $y_j(I, X)$, $j=1, \dots, m$, contained in the m real-value registers are interpreted as the output yielded by I for image X . The values are gathered to form an individual’s output vector

$$\mathbf{y}(I, X) = \langle y_1(I, X), \dots, y_m(I, X) \rangle, \quad (5)$$

that is subject to further processing described in Section 3.

5 Architecture of the Recognition System

The overall recognition system consists of: (i) the best feature extraction procedures O^* constructed using the approach described in Sections 3 and 4, and (ii) classifiers trained using those features.

We incorporate a multi-agent methodology that aims to compensate for the suboptimal character of representations elaborated by the evolutionary process and allows us to boost the overall performance.

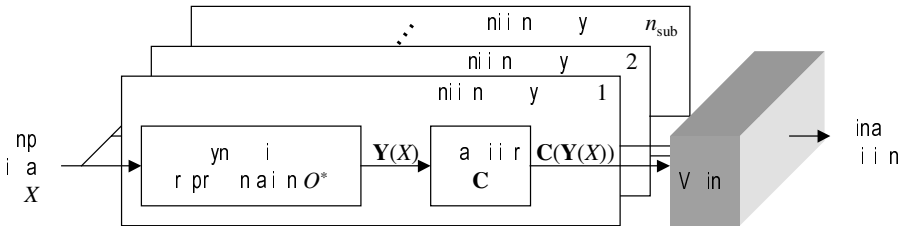


Fig. 3. The top-level architecture of recognition system.

The basic prerequisite for the agents’ fusion to become beneficial is their *diversification*. This may be ensured by using homogenous agents with different parameter settings, homogenous agents with different training data (e.g., bagging [4]), heterogeneous agents, etc. Here, the diversification is naturally provided by the random nature of the genetic search. In particular, we run *many* genetic searches that start from different initial states (initial populations). The best representation O^* evolved in each run becomes a part of a single *subsystem* in the recognition system’s architecture (see Fig. 3). Each subsystem has two major components: (i) a representation O^* , and (ii) a classifier C trained using that representation. As this

classifier training is done once per subsystem, a more sophisticated classifier C may be used here (as compared to the classifier C_{fit} used in the evaluation function).

The subsystems process the input image X independently and output recognition decisions that are further aggregated by a simple majority voting procedure into the final decision. The subsystems are therefore homogenous as far as the structure is concerned; they only differ in the features extracted from the input image and the decisions made. The number of subsystems n_{sub} is a parameter set by the designer.

6 Experimental Results

The primary objective of the computational experiment is to test the scalability of the approach with respect to the number of decision classes and its sensitivity to various types of object distortions. As an experimental testbed, we choose the demanding task of object recognition in synthetic aperture radar (SAR) images. There are several difficulties that make recognition in this modality extremely hard:

- poor visibility of objects – usually only prominent scattering centers are visible,
- low persistence of features under rotation, and
- high levels of noise.

The data source is the MSTAR public database [12] containing real images of several objects taken at different azimuths and at 1-foot spatial resolution. From the original complex (2-channel) SAR images, we extract the magnitude component and crop it to 48×48 pixels. No other form of preprocessing is applied.



Fig. 4. Selected objects and their SAR images used in the learning experiment.

The following parameter settings are used for each coevolutionary run: number of subsystems n_{sub} : 10; classifier C_{fit} used for feature set evaluation: decision tree inducer C4.5 [11]; mutation operator: one-point, probability 0.1; crossover operator: one-point, probability 1.0, cutting allowed at every point; selection operator: tournament selection with tournament pool size = 5; number of registers (image and numeric) m : 2; number of populations n : 4; genome length: 40 bytes (10 operations);

single population size: 200 individuals; time limit for evolutionary search: 4000 seconds (Pentium PC 1.4 GHz processor).

A *compound* classifier **C** is used to boost the recognition performance. In particular, **C** implements the ‘1-vs.-all’ scheme, i.e. it is composed of l *base classifiers* (where l is the number of decision classes), each of them working as a binary (two-class) discriminator between a single decision class and all the remaining classes. To aggregate their outputs, a simple decision rule is used that yields final class assignment only if the base classifiers are consistent and indicate a single decision class. With this strict rule, any inconsistency among the base classifiers (i.e., no class indicated or more than one class indicated) disables univocal decision and the example remains unclassified (assigned to ‘No decision’ category).

The system’s performance is measured using different base classifiers (if not stated otherwise, the classifier uses default parameter settings as specified in [16]):

- support vector machine with polynomial kernels of degree 3 (trained using sequential minimal optimization algorithm [9] with complexity parameter set to 10),
- nonlinear neural networks with sigmoidal units trained using backpropagation algorithm with momentum,
- C4.5 decision tree inducer [11].

Scalability. To investigate the scalability of the proposed approach w.r.t. to the problem size, we use several datasets with increasing numbers of decision classes for a 15-deg. depression angle, starting from $l=2$ decision classes: BRDM2 and ZSU. Consecutive problems are created by adding the decision classes up to $l=8$ in the following order: T62, Zil131, a variant A04 of T72 (T72#A04 in short), 2S1, BMP2#9563, and BTR70#C71.

For i^{th} decision class, its representation D_i in the training data D consists of *two* subsets of images sampled uniformly from the original MSTAR database with respect to a 6-degree azimuth step. Training set D , therefore, always contains $2 \cdot (360/6) = 120$ images from each decision class, so its total size is $120 \cdot l$. The corresponding test set T contains all the remaining images (for a given object and elevation angle) from the original MSTAR collection. In this way, the training and test sets are strictly disjoint. Moreover, the learning task is well represented by the training set as far as the azimuth is concerned. Therefore, there is no need for multiple train-and-test procedures here and the results presented in the following all use this single particular partitioning of MSTAR data.

Let n_c , n_e , and n_u , denote respectively the numbers of test objects correctly classified, erroneously classified, and unclassified by the recognition system. Figure 5(a) presents the *true positive rate*, i.e. $P_p = n_c / (n_c + n_e + n_u)$, also known as *probability of correct identification* (PCI), as a function of the number of decision classes. It can be observed, that the scalability depends heavily on the base classifier, and that SVM clearly outperforms its rivals. For this base classifier, as new decision classes are added to the problem, the recognition performance gradually decreases. The major drop-offs occur when T72 tank and 2S1 self-propelled gun (classes 5 and 6, respectively), are added to the training data; this is probably due to the fact that these objects are visually similar to each other (e.g., both have gun turrets) and significantly resemble the T62 tank (class 3). On the contrary, introducing

consecutive classes 7 and 8 (BMP2 and BTR60) did not affect the performance much; more than this, an improvement of accuracy is even observable for class 7.

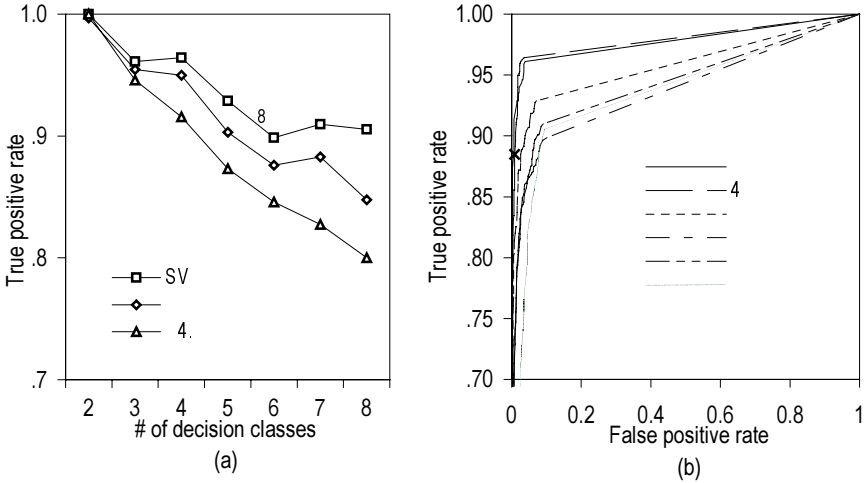


Fig. 5. (a) Test set recognition ratio as a function of number of decision classes. (b) ROC curves for different number of decision classes (base classifier: SVM).

Figure 5(b) shows the *receiver operating characteristics* (ROC) curves obtained, for the recognition systems using SVM as a base classifier, by modifying the confidence threshold that controls whether the classifier votes. The false positive rate is defined here as $P_{fp} = n_c / (n_c + n_v)$. Again, the results support our method: the curves do not drop rapidly as the false positive rate decreases. Therefore, very high *accuracy of classification*, i.e., $n_c / (n_c + n_v)$, may be obtained when accepting a reasonable *rejection rate* $n_v / (n_c + n_v)$. For instance, for 4 decision classes, when $P_{fp} = 0.008$, $P_{tp} = 0.885$ (see marked point in Fig. 5(b)), and, therefore, rejection rate is $1 - (P_{fp} + P_{tp}) = 0.107$, the accuracy of classification equals 0.991.

Object variants. A desirable property of an object recognition system is its ability to recognize different variants of the same object. This task may pose some difficulties, as configurations of vehicles often vary significantly. To provide a comparison with human-designed recognition system, we use the conditions of the experiment reported in [2]. In particular, we synthesized recognition systems using:

- 2 objects: BMP2#C21, T72#132,
- 4 objects: BMP2#C21, T72#132, BTR70#C71, and ZSU23/4.

For both of these cases, the testing set includes two *other* variants of BMP2 (#9563 and #9566), and two *other* variants of T72 (#812 and #s7).

The results of the test set evaluation shown in the confusion matrices (Table 1) suggest that, even when the recognized objects differ significantly from the models provided in the training data, the approach is still able to maintain high performance.

Here the true positive rate P_{tp} equals 0.804 and 0.793, for 2- and 4-class systems, respectively. For the cases where a decision can be made (83.3% and 89.2%, respectively), the values of classification accuracy, 0.966 and 0.940, respectively, are comparable to the forced recognition results of the human-designed recognition algorithms reported in [2], which are 0.958 and 0.942, respectively. Note that in the test, we have not used ‘confusers’, i.e. test images from different classes that those present in the training set, as opposed to [2], where BRDM2 armored personnel carrier has been used for that purpose.

Table 1. Confusion matrices for recognition of object variants.

Test objects		Predicted class							
		2-class system			4-class system				
		BMP2	T72	No	BMP2	T72	BTR	ZSU	No
Object	Serial #	[#C21]	[#132]	decision	[#C21]	[#132]	[#C71]	[#d08]	decision
BMP2	[#9563,9566]	295	18	78	293	27	27	1	43
T72	[#812,s7]	4	330	52	12	323	1	9	41

7 Conclusions

In this contribution, we provide experimental evidence for the possibility of synthesizing, without or with little human intervention, a feature-based recognition system which recognizes 3D objects at the performance level that can be comparable to handcrafted solutions. Let us emphasize that these encouraging results are obtained in the demanding field of SAR imagery, where the acquired images only roughly depict the underlying 3D structure of the object.

There are several major factors that contribute to the overall high performance of the approach. First of all, the paradigm of *coevolution* allows us to decompose the task of representation (feature set) construction into several semi-independent, cooperating subtasks. In this way, we exploit the inherent modularity of the learning process, without the need of specifying explicit objectives for each developed feature extraction procedure. Secondly, the approach manipulates LGP-encoded feature extraction *procedures*, as opposed to most approaches which are usually limited to learning meant as parameter optimization. This allows for learning sophisticated features, which are novel and sometimes very different from expert’s intuition, as may be seen from example shown in Figure 6. And thirdly, the *fusion at feature and decision level* helps us to aggregate sometimes contradictory information sources and build a recognition system that is comparable to human-designed system performance with a bunch of simple components at hand.

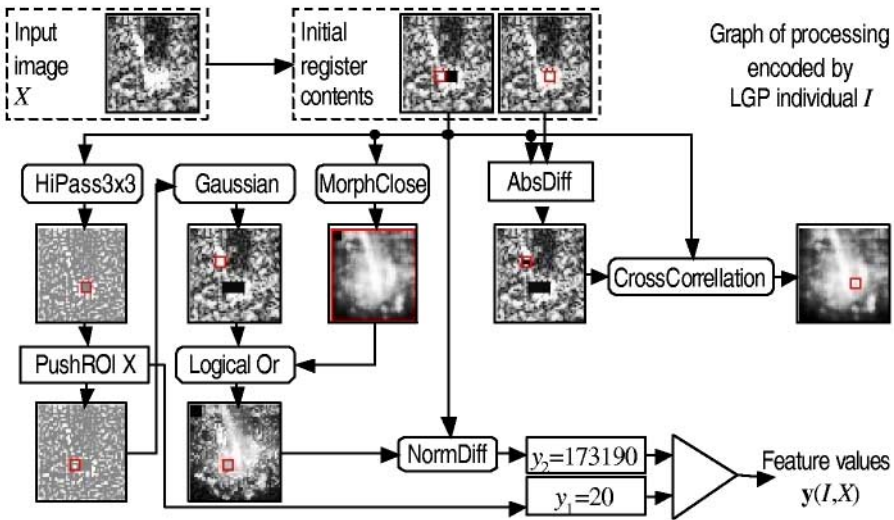


Fig. 6. Processing carried out by one of the evolved procedures shown as a graph (small rectangles in images depict masks; boxes: local operations; rounded boxes: global operations).

Acknowledgements. This research was supported by the grant F33615-99-C-1440. The contents of the information do not necessarily reflect the position or policy of the U. S. Government. The first author is supported by the Polish State Committee for Scientific Research, research grant no. 8T11F 006 19. We would like to thank the authors of software packages: ECJ [7] and WEKA [16] for making their software publicly available.

References

1. Banzhaf, W., Nordic, P., Keller, R., Francine, F.: Genetic Programming. An Introduction. On the automatic Evolution of Computer Programs and its Application. Morgan Kaufmann, San Francisco, Calif. (1998)
2. Bhanu, B., Jones, G.: Increasing the discrimination of SAR recognition models. *Optical Engineering* 12 (2002) 3298–3306
3. Bhanu, B. and Krawiec, K.: Coevolutionary construction of features for transformation of representation in machine learning. *Proc. Genetic and Evolutionary Computation Conference (GECCO 2002)*. AAAI Press, New York (2002) 249–254
4. Breiman, L.: Bagging predictors. *Machine Learning* 24 (1996) 123–140
5. Draper, B., Hanson, A., Riseman, E.: Knowledge-Directed Vision: Control, Learning and Integration. *Proc. IEEE* 84 (1996) 1625–1637
6. Krawiec, K.: On the Use of Pair wise Comparison of Hypotheses in Evolutionary Learning Applied to Learning from Visual Examples. In: Perner, P. (ed.): *Machine Learning and Data Mining in Pattern Recognition*. Lecture Notes in Artificial Intelligence, Vol. 2123. Springer Verlag, Berlin (2001) 307–321.

7. Luke, S.: ECJ Evolutionary Computation System. <http://www.cs.umd.edu/projects/plus/ec/ecj/> (2002)
8. Peng, J., Bhanu, B.: Closed-Loop Object Recognition Using Reinforcement Learning. *IEEE Trans. on PAMI* 20 (1998) 139–154
9. Platt, J.: Fast Training of Support Vector Machines using Sequential Minimal Optimization. In: Schölkopf, B., Burges, C., Smola, A. (eds.): *Advances in Kernel Methods - Support Vector Learning*. MIT Press, Cambridge, Mass. (1998)
10. Potter, M.A., De Jong, K.A.: Cooperative Coevolution: An Architecture for Evolving Coadapted Subcomponents. *Evolutionary Computation* 8 (2000) 1–29
11. Quinlan, J.R.: *C4.5: Programs for machine learning*. Morgan Kaufmann, San Mateo, Calif. (1992)
12. Ross, T., Worell, S., Velten, V., Mossing, J., Bryant, M.: Standard SAR ATR Evaluation Experiments using the MSTAR Public Release Data Set. *SPIE Proc.: Algorithms for Synthetic Aperture Radar Imagery V*, Vol. 3370, Orlando, FL (1998) 566–573
13. Segen, J.: GEST: A Learning Computer Vision System that Recognizes Hand Gestures. In: Michalski, R.S., Tecuci, G., (eds.): *Machine Learning. A Multistrategy Approach. Volume IV*. Morgan Kaufmann, San Francisco, Calif. (1994) 621–634
14. Teller, A., Veloso, M.: A Controlled Experiment: Evolution for Learning Difficult Image Classification. *Proc. 7th Portuguese Conference on Artificial Intelligence*. Springer Verlag, Berlin, Germany (1995) 165–176
15. Wiegand, R.P., Liles, W.C., De Jong, K.A.: An Empirical Analysis of Collaboration Methods in Cooperative Coevolutionary Algorithms. *Proc. Genetic and Evolutionary Computation Conference (GECCO 2001)*. Morgan Kaufmann, San Francisco, Calif. (2001) 1235–1242
16. Witten, I.H., Frank, E.: *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann, San Francisco, Calif. (1999)
17. Wolpert, D., Macready, W.G.: No Free Lunch Theorems for Search. *Tech. Report SFI-TR-95-010*, The Santa Fe Institute (1995)

Finite Population Models of Co-evolution and Their Application to Haploidy versus Diploidy

Anthony M.L. Liekens, Huub M.M. ten Eikelder, and Peter A.J. Hilbers

Department of Biomedical Engineering
Technische Universiteit Eindhoven

P.O. Box 513, 5600MB Eindhoven, The Netherlands

{a.m.l.lieken, h.m.m.t.eikelder, p.a.j.hilbers}@tue.nl

Abstract. In order to study genetic algorithms in co-evolutionary environments, we construct a Markov model of co-evolution of populations with fixed, finite population sizes. In this combined Markov model, the behavior toward the limit can be utilized to study the relative performance of the algorithms. As an application of the model, we perform an analysis of the relative performance of haploid versus diploid genetic algorithms in the co-evolutionary setup, under several parameter settings. Because of the use of Markov chains, this paper provides exact stochastic results on the expected performance of haploid and diploid algorithms in the proposed co-evolutionary model.

1 Introduction

Co-evolution of Genetic Algorithms (GA) denotes the simultaneous evolution of two or more GAs with interdependent or coupled fitness functions. In competitive co-evolution, just like competition in nature, individuals of both algorithms compete with each other to gather fitness. In cooperative co-evolution, individuals have to cooperate to achieve higher fitness. These interactions have previously been modeled in Evolutionary Game Theory (EGT), using replicator dynamics and infinite populations. Similar models have, for example, been used to study equilibriums [2] and comparisons of selection methods [1]. Simulations of competitive co-evolution have previously been used to evolve solutions and strategies for small two-player games, i.e., in [3,4], sorting networks [5], or competitive robotics [6].

In this paper, we provide the construction of a Markov model of co-evolution of two GAs with finite population sizes. After this construction we calculate the relative performances in such a setup, in which a haploid and diploid GA co-evolve with each other.

Commonly, GAs are based on the haploid model of reproduction. In this model, an individual is assumed to carry a single genotype to encode for its phenotype. When two parents are selected for reproduction, recombination of these two genotypes takes place to construct a child for the next generation.

Most higher order species in nature, however, have the characteristic of carrying two sets of alleles that both can encode for the individual's phenotype.

For each of the genes, two (possibly different) alleles are thus present. A dominance relation is defined on each pair of alleles. In a heterozygous gene, i.e., in a gene with 2 different alleles, this dominance relation defines which allele is expressed. A dominance relation can be pure, such that either one of the alleles is always expressed in heterozygous individuals, or it can be partial, such that the result of phenotypic expression is a probability distribution over the alleles. When two diploid parents are selected to reproduce, they produce haploid gamete cells through meiosis, in which each parent's genes are recombined. The haploid gametes are then merged, or fertilized, to form a new diploid child.

In dynamic environments, diploid GAs are hypothesized to perform better than haploid algorithms, since they can build up an implicit long time memory of previously encountered solutions in the recessive parts of the populations' allele pool. These alleles are kept safe from harmful selection. Under the assumption that co-evolution mimics a dynamic environment, we will test this hypothesis with a small problem in this paper, using co-evolution as a special form of dynamic optimization. The Markov model approach yields exact stochastic expectations of performance of haploid and diploid algorithms. Previous accounts of research on the use of diploidy for dynamic optimization, and results of its performance as compared with haploid algorithms, can be found in [5,7,8,9,10]. The methods used in these papers differ from our approach in the fact that we consider exact probability distributions whereas others perform simulation experiments or equilibrium analyses of infinite models. The stochastic method of Markov models, as used in this paper, allows us to provide exact stochastic results and performance expectations, instead of empirical data which is, as we will show later, subject to a large standard deviation. A similar model to the model presented in this paper, discussing stochastic models for dynamic optimization problems, is discussed in [11].

In this study, haploid and diploid populations face one another in co-evolution, which creates a simulation of a comparable situation in the history of life on Earth: The first diploid organisms to appear on Earth had to face haploid life forms in a competition for resources. The dynamics of the co-evolutionary competitive games played by these prehistoric cells are similar to the models presented in this paper. Correct interpretation of the results can give insights whether the earliest diploid life forms were able to compete with haploid life forms.

In this paper, co-evolution, of two competing populations and their governing GAs, is used as a "test bed" to test two algorithms' relative performance in dynamic environments. Indeed, since the fitness of an individual in one of the co-evolving populations is based on the configuration of the opponent population, the fitness landscapes of both populations constantly change, thereby simulating dynamic environments through both populations' interdependent fitness functions. Note that the results can only be used to discuss the algorithms' *relative* performance since the dynamics of one algorithm is explicitly determined by the other algorithm.

2 Models and Methods

In this section, we construct a finite population Markov model of co-evolution. Two finite population Markov chains of simple genetic algorithms, based on the simple GA as described by [12,13], are intertwined through interdependent fitness functions. A discussion of the resulting Markov chain's behavior toward the limit and the interpretation of the limit behavior is also provided.

2.1 Haploid and Diploid Reproduction Schemes

The following constructions are based on the definition of haploid and diploid simple genetic algorithms with finite population sizes as described in [13].

Haploid Reproduction. Let Ω_H be the space of binary bit strings with length l . The bit string serves as a genotype with l loci, that each can hold the alleles 0 or 1. Ω_H serves as the search space for the Haploid Simple Genetic Algorithm (HSGA). Let P_H be a haploid population, $P_H = \{x_0, x_1, \dots, x_{r_H-1}\}$, a multi set with $x_i \in \Omega_H$ for $0 \leq i < r_H$, and $r_H = |P_H|$ the population size. Let π_H denote the set of all possible populations P_H of size r_H .

Let $f_H : \Omega_H \rightarrow \mathbb{R}^+$ denote the fitness function. Let $\varsigma_{f_H} : \pi_H \rightarrow \Omega_H$ represent stochastic selection, proportional to fitness function f_H . Crossover is a genetic operator that takes two parent individuals, and results in a new child individual that shares properties of these parents. Mutation slightly changes the genotype of an individual. Crossover and mutation are represented by the stochastic functions $\chi : \Omega_H \times \Omega_H \rightarrow \Omega_H$ and $\mu : \Omega_H \rightarrow \Omega_H$ respectively.

In a HSGA, a new generation of individuals is created through sexual reproduction of selected parents from the current population. The probability that a haploid individual $i \in \Omega_H$ is generated from a population P_H can be written according to this process as

$$\begin{aligned} \Pr [i \text{ is generated from } P_H] &= \\ \Pr [\mu(\chi(\varsigma_{f_H}(P_H), \varsigma_{f_H}(P_H))) = i] \end{aligned} \tag{1}$$

where it has been shown in [13] that the order of mutation and crossover may be interchanged in equation (1).

Diploid Reproduction. In the Diploid Simple Genetic Algorithm (DSGA), an individual consists of two haploid genomes. An individual of the diploid population is represented by a multi set of two instances of Ω_H , e.g. $\{i, j\}$ with $i, j \in \Omega_H$. The set of all possible diploid instances is denoted by Ω_D , the search space of the DSGA. A diploid population P_D with population size r_D is defined over Ω_D , similar to the definition of a haploid population. Let π_D denote the set of possible populations.

Haploid selection, mutation and crossover are reused in the diploid algorithm. Two more specific genetic operators must be defined. $\delta : \Omega_D \rightarrow \Omega_H$

is the dominance operator. A fitness function f_H defined for the haploid algorithm, can be reused in a fitness function f_D for the diploid algorithm with $f_D(\{i, j\}) = f_H(\delta(\{i, j\}))$ for any $\{i, j\}$ in Ω_D . Another diploid-specific operator is fertilization, which merges two gametes (members of Ω_H) into one diploid individual: $\phi : \Omega_H \times \Omega_H \rightarrow \Omega_D$. Throughout this paper we will assume that $\phi(i, j) = \{i, j\}$ for all i, j in Ω_H . Diploid reproduction can now be written as

$$\begin{aligned} \Pr [\{i, j\} \text{ is generated from } P_D] = \\ \Pr [\phi (\mu (\chi (\varsigma_{f_D}(P_D))), \mu (\chi (\varsigma_{f_D}(P_D)))) = \{i, j\}]. \end{aligned} \quad (2)$$

2.2 Simple Genetic Algorithms

In the simple GA (SGA), a new population P' of fixed size r over search space Ω for the next generation is built according to population P with

$$\begin{aligned} \Pr [\tau(P) = P'] = \\ \frac{r!}{\prod_{i \in \Omega} (\sum_{j \in P'} [i=j])!} \prod_{i \in P'} \Pr [i \text{ is generated from } P] \end{aligned} \quad (3)$$

where $\tau : \pi \rightarrow \pi$ represents the stochastic construction of a new population from and into population space π of the SGA, and $P'(i)$ denotes the number of individuals i in P' . Since the system to create a new generation P' only depends on the previous state P , the SGA is said to be Markovian. The SGA can now be written as a Markov chain with transition matrix T with $T_{P',P} = \Pr [\tau(P) = P']$. If mutation can map any individual to any other individual, all elements of T become strictly positive, and T becomes irreducible and aperiodic. The limit behavior of the Markov chain can then be studied by finding the eigenvector, with corresponding eigenvalue 1, of T .

We will assume uniform crossover, bitwise mutation according to a mutation probability μ , and selection proportional to fitness throughout the paper.

This completes the formal construction of haploid and diploid simple genetic algorithms. More details of this construction can be found in [13].

2.3 Co-evolution of Finite Population Models

Next, we consider the combined co-evolutionary process of two SGAs, respectively defined by population transitions τ_1 and τ_2 , over population search spaces π_1 and π_2 . We assume that the population sizes of both algorithms are fixed and finite, and their generational transitions are executed at the same rate.

In order to make the representative GAs – and thus their fitness functions – interdependent, we need to override the fitness evaluation $f : \Omega \rightarrow \mathbb{R}^+$ of any one of the co-evolving GAs with $f_i : \Omega_i \times \pi_j \rightarrow \mathbb{R}^+$ where Ω_i is the search space of the GA, and π_j is the population state space of the co-evolving GA. As such, the fitness function of an individual in one population becomes dependent on the configuration of the population of the co-evolving GA. Consequently, the

generation probabilities of equation (3) now also depend on the population of the competing algorithm.

The state space π_{co} of the resulting Markov chain of the co-evolutionary algorithm is defined as the Cartesian product of spaces π_1 and π_2 , i.e., $\pi_{co} = \pi_1 \times \pi_2$. All (P, Q) , with $P \in \pi_1, Q \in \pi_2$, are states of the co-evolutionary algorithm. Generally, the transition $\tau_{co} : \pi_{co} \rightarrow \pi_{co}$ in the co-evolutionary Markov chain of two interdependent Markov chains is defined by

$$\Pr [\tau_{co}((P, Q)) = (P', Q')] = \Pr [\tau_1(P) = P' | Q] \cdot \Pr [\tau_2(Q) = Q' | P] \quad (4)$$

where populations P and Q are states of π_1 and π_2 respectively. The dependence of τ_1 and τ_2 on Q and P respectively, gives way for the implementation of a coupled fitness function for either algorithm.

2.4 Limit Behavior

One can show that the combination of irreducible and aperiodic interdependent Markov chains, as defined above, does not generally result in an irreducible and aperiodic Markov chain. Therefore, we cannot simply assume that the Markov chain that defines the co-evolutionary process converges to a unique fixed point.

We can, however, make the following assumptions: If mutation can map any individual – in both of the co-evolving GAs – to any other individual in the algorithm’s search space with a strictly positive probability, then all elements in the transition matrices of both co-evolving Markov chains are always nonzero and strictly positive. As a result from multiplying the transition probabilities in equation (4), all transition probabilities of the co-evolutionary Markov chain are thus strictly positive. This makes the combined Markov chain irreducible and aperiodic, such that the limit behavior of the whole co-evolutionary process can be studied by finding the unique eigenvector, with corresponding eigenvalue 1, of the transition matrix as defined by equation (4), due to the Perron-Frobenius theorem [14].

2.5 Expected Performance

The eigenvector, with corresponding eigenvalue 1, of the co-evolutionary Markov chain describes the fixed point distribution over all possible states (P, Q) of the Markov chain in the limit. As a result, toward the limit, the Markov chain converges to the distribution that describes the overall mean behavior of the co-evolutionary system. If a simulation is run that starts with an initial population according to this distribution, the distribution over the states at all next generations are also according to this fixed point distribution. For each of the states, we can compute the mean fitness of the constituent populations of that state. With this information, and the distribution over all states in the limit, we can make a weighted mean to find the mean fitness of both algorithms in the co-evolutionary system at hand.

More formally, let T denote the $|\pi_{co}| \times |\pi_{co}|$ transition matrix of the co-evolutionary system with transition probabilities

$$T_{(P',Q'),(P,Q)} = \Pr[\tau_{co}((P,Q)) = (P',Q')]$$

as defined by equation (4). Let ξ denote the eigenvector, with corresponding eigenvalue 1, of T . ξ denotes the distribution of states of the co-evolutionary algorithm in the limit, with component $\xi_{(P,Q)}$ denoting the probability of ending up in state $(P,Q) \in \pi_{co}$ in the limit. If $\bar{f}_1(P,Q)$ gives the mean fitness of the individuals in population P , given an opponent population Q , then

$$\bar{f}_1 = \sum_{(P,Q) \in \pi_{co}} \xi_{(P,Q)} \cdot \bar{f}_1(P,Q) \quad \text{with} \quad \bar{f}_1(P,Q) = \frac{1}{|P|} \sum_{i \in P} f_1(i, Q), \quad (5)$$

gives the mean fitness of the populations governing the dynamics of the first algorithm toward the limit, in relation to its co-evolving algorithm. Similarly, the mean fitness of the second algorithm can be computed. We use the mean fitness in the limit as an exact measure of performance of the algorithm, in relation to the co-evolving algorithm. Equation (5) also gives the expected mean fitness of the co-evolving algorithms if simulations of the model are executed.

We will also calculate the variance and standard deviation in order to discuss the significance of the exact results. The variance of the fitness of the first algorithm, according to distribution ξ , is equal to

$$\sigma_{f_1}^2 = \sum_{(P,Q) \in \pi_{co}} \xi_{(P,Q)} \cdot (\bar{f}_1(P,Q) - \bar{f}_1)^2. \quad (6)$$

Similarly to the mean fitness, the variance of the fitness gives an expectation of the variance for simulations of the model.

Given the parameters for fitness determination, selection and reproduction of both co-evolving GAs in the co-evolutionary system, we can now estimate the mean fitness, and discuss the performance of both genetic algorithms, in the context of their competitors' performance.

3 Application

3.1 Competitive Game: Matching Pennies

In order to construct interdependent fitness functions, we can borrow ideas of competitive games from Evolutionary Game Theory (EGT, overviews can be found in [15,16]). EGT studies the dynamics and equilibriums of games played by populations of players. The strategies players employ in the games determine their interdependent fitness.

A common model to study the dynamics – of frequencies of strategies adopted by the populations – is based upon replicator dynamics. This model makes a couple of assumptions, some of which will be discarded in our model. Replicator

dynamics assumes infinite populations, asexual reproduction, complete mixing, i.e., all players are equally likely to interact in the game, and strategies breed true, i.e., strategies are transmitted to offspring proportionally to the payoff achieved. In our finite population model, where two GAs compete against each other, we maintain the assumption that strategies breed true. We also maintain complete mixing, although the stochastic model also represents incomplete mixing with randomly chosen opponent strategies. We now consider finite fixed population sizes with variation and sexual reproduction of strategies.

In the scope of our application, we focus on a family of 2×2 games called “matching pennies.” Consider the payoff matrices for the game in Table 1. Each of the two players in the game either calls ‘heads’ or ‘tails.’ Depending on the players’ calls and their representative values in the payoff matrices, the players receive a payoff. More specifically, the first player receives payoff $1 - L$ if the calls match, and L otherwise. The second player receives 1 minus the first player’s payoff. If L ranges between 0 and 0.5, the first player’s goal therefore is to call the same as the second player, whose goal in turn is to do the inverse. Hence the notion of competition in the game.

Table 1. Payoff matrices of the matching pennies game. One population uses payoff matrix f_1 , where the other players use payoff matrix f_2 . Parameter L denotes the payoff received when the player loses the game, and can range from 0 to 0.5

f_1	heads	tails
heads	$1 - L$	L
tails	L	$1 - L$

f_2	heads	tails
heads	L	$1 - L$
tails	$1 - L$	L

Let a population of players denote a finite sized population consisting of individuals who either call ‘heads’ or ‘tails.’ In our co-evolutionary setup, two GAs evolving such populations P and Q are put against one another. The fitnesses of individuals in population P and Q are based on f_1 and f_2 , from Table 1, respectively. We use complete mixing to determine the fitness of each individual in either of the populations: Let p_{heads} denote the proportion of individuals in population P who call ‘heads,’ and q_{heads} the proportion of individuals in Q to call ‘heads.’ Define p_{tails} and q_{tails} similarly for the proportion of ‘tails’ in the populations. The fitness of an individual i of population P , regarding the constituent strategies of population Q , can now be defined as

$$f_1(i, Q) = \begin{cases} q_{\text{heads}} \cdot (1 - L) + q_{\text{tails}} \cdot L & \text{if } i \text{ calls ‘heads’} \\ q_{\text{tails}} \cdot (1 - L) + q_{\text{heads}} \cdot L & \text{if } i \text{ calls ‘tails’} \end{cases} \quad (7)$$

and that of an individual j in population Q as

$$f_2(j, P) = \begin{cases} p_{\text{heads}} \cdot L + p_{\text{tails}} \cdot (1 - L) & \text{if } j \text{ calls ‘heads’} \\ p_{\text{tails}} \cdot L + p_{\text{heads}} \cdot (1 - L) & \text{if } j \text{ calls ‘tails’} \end{cases} \quad (8)$$

It can easily be verified that the mean fitness of population P always equals 1 minus the mean fitness of population Q , i.e., $\overline{f}_1(P, Q) = 1 - \overline{f}_2(Q, P)$. Similarly, the mean fitness of both algorithms sum up to 1, with $\overline{f}_1 = 1 - \overline{f}_2$, c.f. equation (5). If we assume $0 \leq L < 0.5$, then there exists a unique Nash equilibrium of this game, where both populations call ‘heads’ or ‘tails,’ each with probability 0.5. In this equilibrium, both populations receive a mean fitness of 0.5. No player can benefit by changing her strategy while the other players keep their strategies unchanged. Any deviation from this indicates that one algorithm relatively performs better at the co-evolutionary task at hand than the other. As we want to compare the performance of algorithms in a competitive co-evolutionary setup, this is a viable null hypothesis.

3.2 Haploid versus Diploid

For the matching pennies game, we construct a co-evolutionary Markov chain in which a haploid and diploid GA compete with each other. With this construction, and their transition matrices, we can determine the performance of both algorithms according to the limit behavior of the Markov chain. Depending on the results, either algorithm can be elected as a relatively better algorithm.

Let the length of binary strings in both algorithms be $l = 1$. This is referred to as the single locus, two allele problem, a common, yet small, setup in population genetics. An individual with phenotype 0 calls ‘heads,’ and ‘tails’ if the phenotype is 1. Note that uniform crossover will not recombine genes since there is only one locus, but will rather select one of both parent gametes.

Let π_{co} be the search space of the co-evolutionary system, defined by the Cartesian product of the haploid populations’ search space π_H and diploid populations π_D , such that $\pi_{co} = \pi_H \times \pi_D$. Depending on a fixed population size r for both competing algorithms, $|\pi_{co}| = ((r + 2)(r + 1)^2)/2$ denotes the size of the co-evolutionary state space.

For any state $(P, Q) \in \pi_{co}$, let equations (7) and (8) be the respective fitness functions for the individuals in the haploid and diploid algorithms. Since we want to compare the algorithms’ performance under comparable conditions, both populations are assumed to have the same parameters for recombination and mutation.

3.3 Limit Behavior and Mean Fitness

According to the definition of the co-evolutionary system in equation (4), the transition matrix for a given set of parameters can be calculated. The eigenvector, with corresponding eigenvalue 1, of this transition matrix can be found through iterated multiplication of the transition matrix with an initially distributed stochastic vector. From the resulting eigenvector we can find the mean fitness of the co-evolutionary GAs toward the limit. These means are discussed in the following sections.

We split the presentation of the limit behavior results into two separate sections. In the first section, we discuss the results given the assumption of pure

dominance, i.e., one of both alleles, either 0 or 1 is strictly dominant over the other allele. In the second part, we discuss the results in the case of partial dominance. In this setting, the phenotype of the diploid heterozygous genotype $\{0, 1\}$ is defined by a probability distribution over 0 and 1.

Pure dominance. Let 1 be the dominant allele, and 0 the recessive allele in diploid heterozygous individuals. This implies that diploid individuals with genotype $\{0, 1\}$ have phenotype 1¹.

Figure 1 shows the mean fitness of the haploid algorithm, which is derived from the co-evolutionary systems' limit behavior, using equation (5). The proportion of parameter settings for which diploidy performs better, increases as the population size of the algorithms becomes bigger.

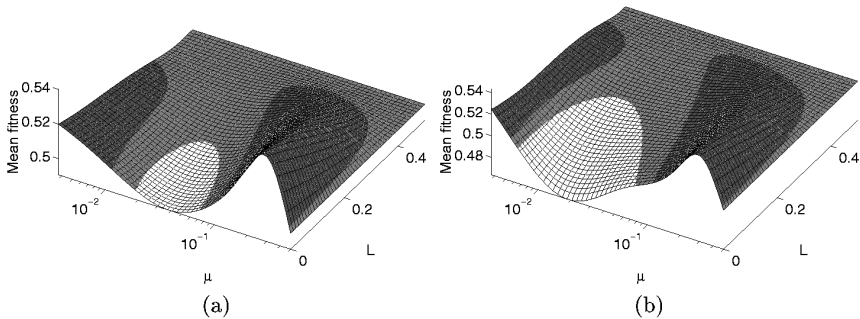


Fig. 1. Exact mean fitness of the haploid GA in the co-evolutionary system, for variable mutation rate μ and payoff parameter L . The mean fitness of the diploid algorithm always equals 1 minus the mean fitness of the haploid algorithm. Population size of both algorithms is fixed to 5 in (a) and 15 in (b). The mesh is colored light as the mean fitness is below 0.4975, i.e. when the diploid algorithm performs better, and dark as the mean fitness is over 0.5025, i.e. for parameters where haploidy performs better.

In our computations, we found a fairly large standard deviation near $\mu = 0$ and $L = 0$. The standard deviation goes to zero as either of the parameters go to 0.5. We discuss the source of this fairly large standard deviation in section 3.4. Because of the large standard deviation, it is very hard to obtain these results with empirical runs of the model. However, it is hard to compute the exact limit behavior of large population systems, since this implies that we need to find the eigenvector of a matrix with $\mathcal{O}(r^6)$ elements for population size r .

Partial dominance. Instead of using a pure dominance scheme in the diploid GA, we can also assign a partial dominance scheme to the dominance operator. In

¹ If we would choose 0 as the dominant allele instead of 1, the co-evolutionary system would yield the exact same performance results, because of symmetries in the matching pennies game. The same holds for exchanging fitness functions f_1 and f_2 .

this dominance scheme, the heterozygous genotype $\{0, 1\}$ has phenotype 0 with probability h , and phenotype 1 with probability $1 - h$. h is called the dominance degree or coefficient. The dominance degree is the measure of dominance of the recessive allele in the case of heterozygosity. Since our model is stochastic, we could also state that the fitness of an heterozygous individual is an intermediate of the fitnesses of both homozygous phenotypes.

The performance results are summarized in Figure 2. The figures show significantly better performance results for the diploid algorithm under small mutation and high selection pressure (small L), in relation to the haploid algorithm. Indeed, if we consider partial dominance instead of pure dominance, the memorized strategies in the recessive alleles of a partial dominant diploid population are tested against the environment, even in heterozygous individuals. The fact that this could lead to lower fitnesses in heterozygous individuals because of interpolation of high and low fitness, does not restrict the diploid algorithm from obtaining a higher mean fitness in the co-evolutionary algorithm. The standard deviation is smaller than in the pure dominance case. This is explained in section 3.4.

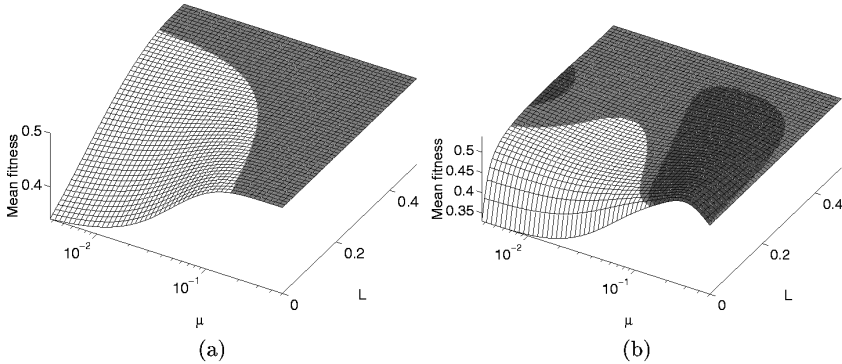


Fig. 2. Mean fitness in the limit of the haploid algorithm similar to Figure 1 for different dominance coefficients, with $r = 15$. Figure (a) applies dominance degree $h = 0.5$ and (b) has dominance degree $h = 0.01$. Figure 1 applies dominance degree $h = 0$

3.4 Source of High Variance

In order to find where the high variance originates, we analyze the distribution of fitness at the fixed point. Dissecting the stable fixed point shows that there are a small number of states with high probability, and many other states with a small probability. More specifically, of these states with a high probability, about half of them have an extremely high mean fitness for one algorithm, where the other half have an extremely low mean fitness. This explains the high variance in the fitness distribution. If we would run a simulation of the model, we would

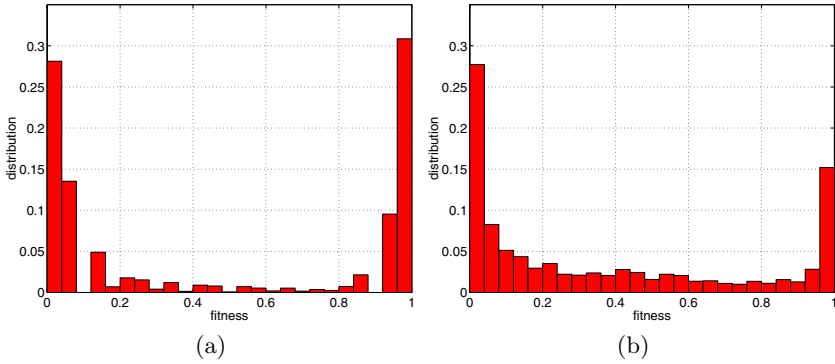


Fig. 3. Histogram showing the distribution of fitness of the haploid genetic algorithm, in the limit. Both figures have parameters $r = 10$, $\mu = 0.01$, $L = 0$. Figure (a) shows the distribution for $h = 0$ and $h = 0.5$ for (b). $\bar{f}_1 = 0.4768$ and $\sigma_{f_1} = 0.4528$ in histogram (a) and $\bar{f}_1 = 0.3699$ and $\sigma_{f_1} = 0.3715$ in (b)

see that the algorithm alternately visits high and low fitness states, and switches relatively fast between these sets of states.

Figure 3 shows that, toward the limit, the mean fitness largely depends on states with both extremely low and high fitnesses, which corresponds with the high standard deviation. Note that the standard deviation is smaller in the case of a higher dominance degree. This is also due to average fitnesses being smeared out in heterozygous individuals because of the higher dominance degree. The relative difference between frequencies of extremely low and high fitnesses also results in a lower variance, as the dominance degree increases.

4 Discussion

This paper shows how a co-evolutionary model of two GAs with finite population size can be constructed. We also provide ways to measure and discuss the relative performance of the algorithms at hand. Because of the use of Markov chains, exact stochastic results can be computed.

The analyses presented in the application of this paper show that, given the matching pennies game, and if pure dominance is assumed, the results are only in favor of diploidy in case of specific parameter settings. Even then, the results are not significant and subject to a large standard deviation. A diploid GA with partial dominance and a strictly positive dominance degree can outperform a haploid GA, if similar conditions hold for both algorithms. These results are expressed best under low mutation pressure and high selection pressure, i.e., when a deleterious mutation has an almost lethal effect on the individual. Diploidy performs relatively better as the population size increases.

Based on these results, we suggest that further research should be undertaken on the usage of diploidy in co-evolutionary GAs. This paper studies a

small problem and small search spaces. Empirical evidence might prove to be a useful tool in studying complexer problems, or larger populations. Scaled up versions – of small situations which can be analyzed exactly – could be used as empirical evidence to support exact predictions. Low significance and high standard deviations might prove that the study of relative performance of GAs in competitive co-evolutionary situations is, however, empirically hard.

References

1. S. G. Ficici, O. Melnik, and J. B. Pollack. A game-theoretic investigation of selection methods used in evolutionary algorithms. In *Proceedings of the 2000 Congress on Evolutionary Computation*, 2000.
2. S. G. Ficici and J. B. Pollack. A game-theoretic approach to the simple coevolutionary algorithm. In *Parallel Problem Solving from Nature VI*, 2000.
3. C. D. Rosin. *Coevolutionary search among adversaries*. PhD thesis, San Diego, CA, 1997.
4. A. Lubberts and R. Miikkulainen. Co-evolving a go-playing neural network. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, pages 14–19, 2001.
5. D. Hillis. Co-evolving parasites improve simulated evolution as an optimization procedure. In *Artificial Life II*. Addison Wesley, 1992.
6. D. Floreano, F. Mondada, and S. Nolfi. Co-evolution and ontogenetic change in competing robots. In *Robotics and Autonomous Systems*, 1999.
7. D. E. Goldberg and R. E. Smith. Nonstationary function optimization using genetic algorithms with dominance and diploidy. In *Second International Conference on Genetic Algorithms*, pages 59–68, 1987.
8. J. Lewis, E. Hart, and G. Ritchie. A comparison of dominance mechanisms and simple mutation on non-stationary problems. In *Parallel Problem Solving from Nature V*, pages 139–148, 1998.
9. K. P. Ng and K. C. Wong. A new diploid scheme and dominance change mechanism for non-stationary function optimization. In *6th Int. Conf. on Genetic Algorithms*, pages 159–166, 1995.
10. R. E. Smith and D. E. Goldberg. Diploidy and dominance in artificial genetic search. *Complex Systems*, 6:251–285, 1992.
11. A. M. L. Liekens, H. M. M. ten Eikelder, and P. A. J. Hilbers. Finite population models of dynamic optimization with alternating fitness functions. In *GECCO Workshop on Evolutionary Algorithms for Dynamic Optimization Problems*, 2003.
12. A. E. Nix and M. D. Vose. Modelling genetic algorithms with markov chains. *Annals of Mathematics and Artificial Intelligence*, pages 79–88, 1992.
13. A. M. L. Liekens, H. M. M. ten Eikelder, and P. A. J. Hilbers. Modeling and simulating diploid simple genetic algorithms. In *Foundations of Genetic Algorithms VII*, 2003.
14. D. P. Bertsekas and J. N. Tsitsiklis. *Parallel and Distributed Computation*. Prentice-Hall, 1989.
15. J. W. Weibull. *Evolutionary Game Theory*. MIT Press, Cambridge, Massachusetts, 1995.
16. J. Hofbauer and K. Sigmund. *Evolutionary Games and Population Dynamics*. Cambridge University Press, 1998.

Evolving Keepaway Soccer Players through Task Decomposition

Shimon Whiteson, Nate Kohl, Risto Miikkulainen, and Peter Stone

Department of Computer Sciences

The University of Texas at Austin

1 University Station C0500

Austin, Texas 78712-1188

{shimon,nate,risto,pstone}@cs.utexas.edu

<http://www.cs.utexas.edu/~shimon,nate,risto,pstone>

Abstract. In some complex control tasks, learning a direct mapping from an agent’s sensors to its actuators is very difficult. For such tasks, decomposing the problem into more manageable components can make learning feasible. In this paper, we provide a task decomposition, in the form of a decision tree, for one such task. We investigate two different methods of learning the resulting subtasks. The first approach, layered learning, trains each component sequentially in its own training environment, aggressively constraining the search. The second approach, coevolution, learns all the subtasks simultaneously from the same experiences and puts few restrictions on the learning algorithm. We empirically compare these two training methodologies using neuro-evolution, a machine learning algorithm that evolves neural networks. Our experiments, conducted in the domain of simulated robotic soccer keepaway, indicate that neuro-evolution can learn effective behaviors and that the less constrained coevolutionary approach outperforms the sequential approach. These results provide new evidence of coevolution’s utility and suggest that solution spaces should not be over-constrained when supplementing the learning of complex tasks with human knowledge.

1 Introduction

One of the goals of machine learning algorithms is to facilitate the discovery of novel solutions to problems, particularly those that might be unforeseen by human problem-solvers. As such, there is a certain appeal to “tabula rasa learning,” in which the algorithms are turned loose on learning tasks with no (or minimal) guidance from humans. However, the complexity of tasks that can be successfully addressed with tabula rasa learning given current machine learning technology is limited.

When using machine learning to address tasks that are beyond this complexity limit, some form of human knowledge must be injected. This knowledge simplifies the learning task by constraining the space of solutions that must be considered. Ideally, the constraints simply enable the learning algorithm to find

the best solutions more quickly. However there is also the risk of eliminating the best solutions from the search space entirely.

In this paper, we consider a multi-agent control task that, given current methods, seems infeasible to learn via a tabula rasa approach. Thus, we provide some structure via a task decomposition in the form of a decision tree. Rather than learning the entire task from sensors to actuators, the agents now learn a small number of subtasks that are combined in a predetermined way.

Providing the decision tree then raises the question of how training should proceed. For example, 1) the subtasks could be learned sequentially, each in its own training environment, thereby adding additional constraints to the solution space. On the other hand, 2) the subtasks could be learned simultaneously from the same experiences. The latter methodology, which can be considered coevolution of the subtasks, does not place any further restrictions on the learning algorithms beyond the decomposition itself.

In this paper, we empirically compare these two training methodologies using neuro-evolution, a machine learning algorithm that evolves neural networks. We attempt to learn agent controllers for a particular domain, namely keepaway in simulated robotic soccer. Our results indicate that neuro-evolution can learn effective keepaway behavior, though constraining the task beyond the tabula rasa approach proves necessary. We also find that the less constrained coevolutionary approach to training the subtasks outperforms the sequential approach. These results provide new evidence of coevolution’s utility and suggest that solution spaces should not be over-constrained when supplementing the learning of complex tasks with human knowledge.

The remainder of the paper is organized as follows. Section 2 introduces the keepaway task as well as the general neuro-evolution methodology. Section 3 fully specifies the different approaches that we compare in this paper. Detailed empirical results are presented in Section 4 and are evaluated in Section 5. Section 6 concludes and discusses future work.

2 Background

This section describes simulated robotic soccer keepaway, the domain used for all experiments reported in this paper. We also review the fundamentals of neuro-evolution, the general machine learning algorithm used throughout.

2.1 Keepaway

The experiments reported in this paper are all in a keepaway subtask of robotic soccer [15]. In keepaway, one team of agents, the *keepers*, attempts to maintain possession of the ball while the other team, the *takers*, tries to get it, all within a fixed region. Keepaway has been used as a testbed domain for several previous machine learning studies. For example, Stone and Sutton implemented keepaway in the RoboCup soccer simulator [14]. They hand-coded low-level behaviors and applied learning, via the Sarsa(λ) method, only to the high-level decision of when

and where to pass. Di Pietro et al. took a similar approach, though they used genetic algorithms and a more elaborate high-level strategy [8]. Machine learning was applied more comprehensively in a study that used genetic programming, though in a simpler grid-based environment [6].

We implement the keepaway task within the SoccerBots environment [1]. SoccerBots is a simulation of the dynamics and dimensions of a regulation game in the RoboCup small-size robot league [13], in which two teams of robots maneuver a golf ball on a field built on a standard ping-pong table. SoccerBots is smaller in scale and less complex than the RoboCup simulator [7], but it runs approximately an order of magnitude faster, making it a more convenient platform for machine learning research.

To set up keepaway in SoccerBots, we increase the size of the field to give the agents enough room to maneuver. To mark the perimeter of the game, we add a large bounding circle around the center of the field. Figure 1 shows how a game of keepaway is initialized. Three keepers are placed just inside this circle at points equidistant from each other. We place a single taker in the center of the field and place the ball in front of a randomly selected keeper.

After initialization, an episode of keepaway proceeds as follows. The keepers receive one point for every pass completed. The episode ends when the taker touches the ball or the ball exits the bounding circle. The keepers and the taker are permitted to go outside the bounding circle.

In this paper, we evolve a controller for the keepers, while the taker is controlled by a fixed intercepting behavior. The keepaway task requires complex behavior that integrates sensory input about teammates, the opponent, and the ball. The agents must make high-level decisions about the best course of action and develop the precise control necessary to implement those decisions. Hence, it forms a challenging testbed for machine learning research.

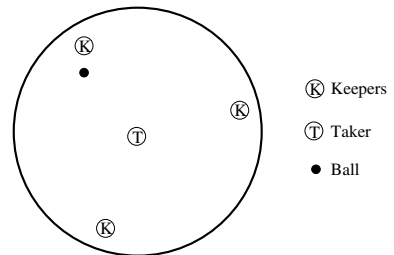


Fig. 1. A game of keepaway after initialization. The keepers try to complete as many passes as possible while preventing the ball from going out of bounds and the taker from touching it.

2.2 Neuro-evolution

We train a team of keepaway players using neuro-evolution, a machine learning technique that uses genetic algorithms to train neural networks [11]. In its simplest form, neuro-evolution strings the weights of a neural network together to form an individual genome. Next, it evolves a population of such genomes by evaluating each one in the task and selectively reproducing the fittest individuals through crossover and mutation.

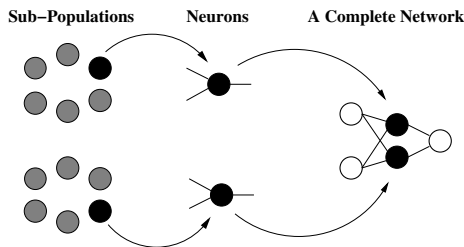


Fig. 2. The Enforced Sub-Populations Method (ESP). The population of neurons is segregated into sub-populations, shown here as clusters of grey circles. One neuron, shown in black, is selected from each sub-population. Each neuron consists of all the weights connecting a given hidden node to the input and output nodes, shown as white circles. The selected neurons together form a complete network which is then evaluated in the task.

The Enforced Sub-Populations Method (ESP) [4] is a more advanced neuro-evolution technique. Instead of evolving complete networks, it evolves sub-populations of neurons. ESP creates one sub-population for each hidden node of the fully connected two-layer feed-forward networks it evolves. Each neuron is itself a genome which records the weights going into and coming out of the given hidden node. As Figure 2 illustrates, ESP forms networks by selecting one neuron from each sub-population to form the hidden layer of a neural network, which it evaluates in the task. The fitness is then passed back equally to all the neurons that participated in the network. Each sub-population tends to converge to a role that maximizes the fitness of the networks in which it appears. ESP is more efficient than simple neuro-evolution because it decomposes a difficult problem (finding a highly fit network) into smaller subproblems (finding highly fit neurons).

In several benchmark sequential decision tasks, ESP outperformed other neuro-evolution algorithms as well as several reinforcement learning methods [2, 3, 4]. ESP is a promising choice for the keepaway task because the basic skills required in keepaway are similar to those at which ESP has excelled before.

3 Method

The goals of this study are 1) to verify that neuro-evolution can learn effective keepaway behavior, 2) to show that decomposing the task is more effective than tabula rasa learning, and 3) to determine whether coevolving the component tasks can be more effective than learning them sequentially.

Unlike soccer, in which a strong team will have forwards and defenders specialized for different roles, keepaway is symmetric and can be played effectively with homogeneous teams. Therefore, in all these approaches, we develop one controller to be used by all three keeper agents. Consequently, all the agents have the same set of behaviors and the same rules governing when to use them,

though they are often using different behaviors at any time. Having identical agents makes learning easier, since each agent learns from the experiences of its teammates as well as its own. In the remainder of this section, we describe the three different methods that we consider for training these agents.

3.1 Tabula Rasa Learning

In the tabula rasa approach, we want our learning method to master the task with minimal human guidance. In keepaway, we can do this by training a single “monolithic” network. Such a network attempts to learn a direct mapping from the agent’s sensors to its actuators. As designers, we need only specify the network’s architecture (i.e. the inputs, hidden units, outputs, and their connectivity) and neuro-evolution does the rest. The simplicity of such an approach is appealing though, in difficult tasks like keepaway, learning a direct mapping may be beyond the ability of our training methods, if not simply beyond the representational scope of the network.

To implement this monolithic approach with ESP, we train a fully connected two-layer feed-forward network with nine inputs, four hidden nodes, and two outputs, as illustrated in Figure 3. This network structure was determined, through experimentation, to be the most effective. Eight of the inputs specify the positions of four crucial objects on the field: the agent’s two teammates, the taker, and the ball. The ninth input represents the distance of the ball from the field’s bounding circle. The inputs to this network and all those considered in this paper are represented in polar coordinates relative to the agent. The four hidden nodes allow the network to learn a compacted representation of its inputs. The network’s two outputs control the agent’s movement on the field: one alters its heading, the other its speed. All runs use sub-populations of size 100.

Since learning a robust keepaway controller directly is so challenging, we facilitate the process through incremental evolution. In incremental evolution, complex behaviors are learned gradually, beginning with easy tasks and advancing through successively more challenging ones. Gomez and Miikkulainen showed that this method can learn more effective and more general behavior than direct evolution in several dynamic control tasks, including prey capture [2] and non-Markovian double pole-balancing [3].

We apply incremental evolution to keepaway by changing the taker’s speed. When evolution begins, the taker can move only 10% as quickly as the keepers. We evaluate each network in 20 games of keepaway and sum its scores (numbers of completed passes) to obtain its fitness. When the population’s average fitness exceeds 50 (2.5 completed passes per episode), the taker’s speed

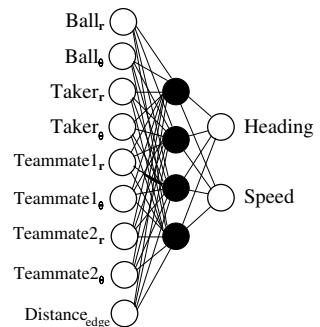


Fig. 3. The monolithic network for controlling keepers. White circles indicate inputs and outputs while black circles indicate hidden nodes.

is incremented by 5%. This process continues until the taker is moving at full speed or the population's fitness has plateaued.

3.2 Learning with Task Decomposition

If learning a monolithic network proves infeasible, we can make the problem easier by decomposing it into pieces. Such task decomposition is a powerful, general principle in artificial intelligence that has been used successfully with machine learning in the full robotic soccer task [12]. In the keepaway task, we can replace the monolithic network with several smaller networks: one to pass the ball, another to receive passes, etc.

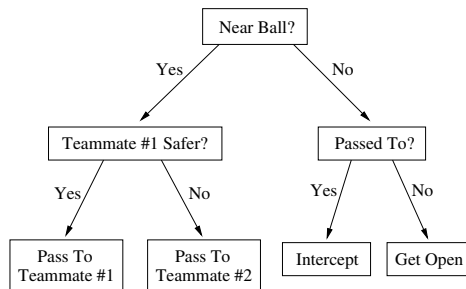


Fig. 4. A decision tree for controlling keepers in the keepaway task. The behavior at each of the leaves is learned through neuro-evolution. A network is also evolved to decide which teammate the agent should pass to.

To implement this decomposition, we developed a decision tree, shown in Figure 4, for controlling each keeper. If the agent is near the ball, it kicks to the teammate that is more likely to successfully receive a pass. If it is not near the ball, the agent tries to get open for a pass unless a teammate announces its intention to pass to it, in which case it tries to receive the pass by intercepting the ball. The decision tree effectively provides some structure (based on human knowledge of the task) to the space of policies that can be explored by the learners.

To implement this decision tree, four different networks must be trained. The networks, illustrated in Figure 5, are described in detail below. As in the monolithic approach, these network structures were determined, through experimentation, to be the most effective.

Intercept: The goal of this network is to get the agent to the ball as quickly as possible. The obvious strategy, running directly towards the ball, is optimal only if the ball is not moving. When the ball has velocity, an ideal interceptor must anticipate where the ball is going. The network has four inputs: two for the ball's current position and two for the ball's current velocity. It has

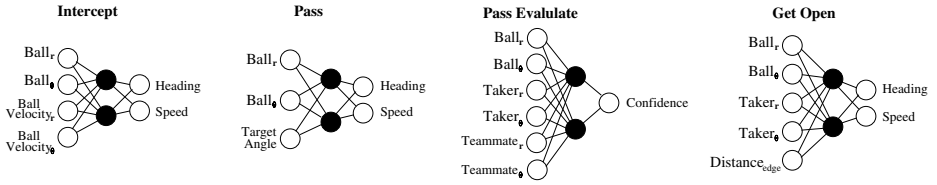


Fig. 5. The four networks used to implement the decision tree shown in Figure 4. White circles indicate inputs and outputs while black circles indicate hidden nodes.

two hidden nodes and two outputs, which control the agent’s heading and speed.

Pass: The pass network is designed to kick the ball away from the agent at a specified angle. Passing is difficult because an agent cannot directly specify what direction it wants the ball to go. Instead, the angle of the kick depends on the agent’s position relative to the ball. Hence, kicking well requires a precise “wind-up” to approach the ball at the correct speed from the correct angle. The pass network has three inputs: two for the ball’s current position and one for the target angle. It has two hidden nodes and two outputs, which control the agent’s heading and speed.

Pass Evaluate: Unlike the other networks, which correspond to behaviors at the leaves of the decision tree, the pass evaluator implements a branch of the tree: the point when the agent must decide which teammate to pass to. It analyzes the current state of the game and assesses the likelihood that an agent could successfully pass to a specific teammate. The pass evaluate network has six inputs: two each for the position of the ball, the taker, and the teammate whose potential as a receiver it is evaluating. It has two hidden nodes and one output, which indicates, on scale of 0 to 1, its confidence that a pass to the given teammate would succeed.

Get Open: The get open network is activated when a keeper does not have a ball and is not receiving a pass. Clearly, such an agent should get to a position where it can receive a pass. However, an optimal get open behavior would not just position the agent where a pass is most likely to succeed. Instead, it would position the agent where a pass would be most strategically advantageous (e.g. by considering future pass opportunities as well). The get open network has five inputs: two for the ball’s current position, two for the taker’s current position, and one indicating how close the agent is to the field’s bounding circle. It has two hidden nodes and two outputs, which control the agent’s heading and speed.

After decomposing the task as described above, we need to evolve networks for each of the four subtasks. These networks can be trained in sequence, through layered learning, or simultaneously, through coevolution. The remainder of this section details these two alternatives.

Layered Learning. One approach to training the components of a task decomposition is layered learning, a bottom-up paradigm in which low-level behaviors are learned prior to high-level ones [16]. Since each component is trained separately, the learning algorithm optimizes over several small solution spaces, instead of one large one. However, since some sub-behaviors must be learned before others, it is not usually possible to train each component in the actual domain. Instead, we must construct a special training environment for each component. The hierarchical nature of layered learning makes this construction easier: since the components are learned from the bottom-up, we can use the already completed sub-behaviors to help construct the next training environment.

In the original implementation of layered learning, each sub-task was learned and frozen before moving to the next layer [16]. However, in some cases it is beneficial to allow some of the lower layers to continue learning while the higher layers are trained [17]. For simplicity, here we freeze each layer before proceeding.

Figure 6 shows one way in which the components of the task decomposition can be trained using layered learning. An arrow from one layer to another indicates that the latter layer depends on the former. A given task cannot be learned until all the layers that point to it have been learned. Hence, learning begins at the bottom, with intercept, and moves up the hierarchy step by step. The training environment for each layer is described below.

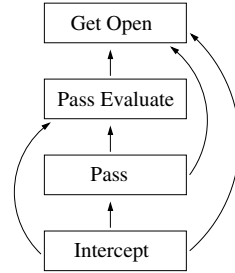


Fig. 6. A layered learning hierarchy for the keepaway task. Each box represents a layer and arrows indicate dependencies between layers. A layer cannot be learned until all the layers it depends on have been learned.

Intercept: To train the interceptor, we propel the ball towards the agent at various angles and speeds. The agent is rewarded for minimizing the time it takes to touch the ball. As the interceptor improves, the initial angle and speed of the ball increase incrementally.

Pass: To train the passer we propel the ball towards the agent and randomly select at which angle we want it to kick the ball. The agent employs the intercept behavior learned in the previous layer until it arrives near the ball, at which point it switches to the pass behavior being evolved. The agent’s reward is inversely proportional to the difference between the target angle and the ball’s actual direction of travel. As the passer improves, the range of angles at which it is required to pass increases incrementally.

Pass Evaluate: To train the pass evaluator, the ball is placed in the center of the field and the pass evaluator is placed just behind it at various angles. Two teammates are situated near the edge of the bounding circle on the other side of the ball at a randomly selected angle. A single taker is placed similarly but nearer to the ball to simulate the pressure it exerts on the passer. The teammates and the taker use the previously learned intercept behavior. We

run the evolving network twice, once for each teammate, and pass to the teammate who receives the higher evaluation. The agent is rewarded only if the pass succeeds.

Get Open: When training the get open behavior, the other layers have already been learned. Hence, the get open network can be trained in a complete game of keepaway. Its training environment is identical to that of the monolithic approach with one exception: during a fitness evaluation the agents are controlled by our decision tree. The tree determines when to use each of the four networks (the three previously trained components and the evolving get open behavior).

At each layer, the results of previous layers are used to assist in training. In this manner, all the components of the task decomposition can be trained and assembled into an effective keepaway controller. However, the behaviors learned with this method are optimized for their training environment, not the keepaway task as a whole. It may sometimes be possible to learn more effective behaviors through coevolution, which we discuss next.

Coevolution. A much less constrained method of learning the keepaway agents' sub-behaviors is to evolve them all simultaneously, a process called coevolution. In general, coevolution can be competitive [5,10], in which case the components are adversaries and one component's gain is another's loss. Coevolution can also be cooperative [9], as when the various components share fitness scores.

In our case, we use an extension of ESP designed to coevolve several cooperating components. This method, called Multi-Agent ESP, has been successfully used to master multi-agent predator-prey tasks [18]. In Multi-Agent ESP, each component is evolved with a separate, concurrent run of ESP. During a fitness evaluation, networks are formed in each ESP and evaluated together in the task. All the networks that participate in the evaluation receive the same score. Therefore, the component ESPs coevolve compatible behaviors that together solve the task.

The training environment for this coevolutionary approach is very similar to that of the get open layer described above. The decision tree still governs each keeper's behavior though the four networks are now all learning simultaneously, whereas three of them were fixed in the layered approach.

4 Empirical Results

To compare monolithic learning, layered learning, and coevolution, we ran seven trials of each method, each of which evolved for 150 generations. In the layered approach, the get open behavior, trained in a full game of keepaway, ran for 150 generations. Additional generations were used to train the lower layers. Figure 7 shows what task difficulty (i.e. taker speed) each method reached during the course of evolution, averaged over all seven runs. This graph shows that decomposing the task vastly improves neuro-evolution's ability to learn effective

controllers for keepaway players. The results also demonstrate the efficacy of coevolution. Though it requires fewer generations to train and less effort to implement, it achieves substantially better performance than the layered approach in this task.

How do the networks trained in these experiments fair in the hardest version of the task? To determine this, we tested the evolving networks from each method against a taker moving at 100% speed. At every fifth generation, we selected the strongest network from the best run of each method and subjected it to 50 fitness evaluations, for a total of 1000 games of keepaway for each network (recall that one fitness evaluation consists of 20 games of keepaway). Figure 8, which shows the results of these tests, further verifies the effectiveness of coevolution. The learning curve of the layered approach appears flat, indicating that it was unable to significantly improve the keepers' performance through training the get open network. However, the layered approach outperformed the monolithic method, suggesting that it made substantial progress when training the lower layers. It is essential to note that neither the layered nor monolithic approaches trained at this highest task difficulty, whereas the best run of coevolution did. Nonetheless, these tests provide additional confirmation that neuro-evolution can truly master complex control tasks once they have been decomposed, particularly when using a coevolutionary approach.

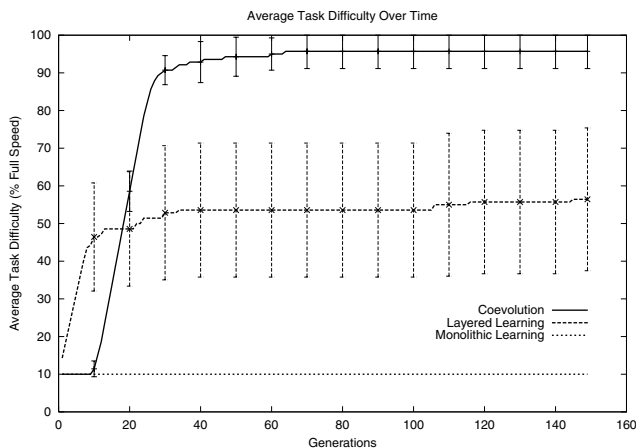


Fig. 7. Task difficulty (i.e. taker speed) of each method over generations, averaged over seven runs. Task decomposition proves essential for reaching the higher difficulties. Only coevolution reaches the hardest task.

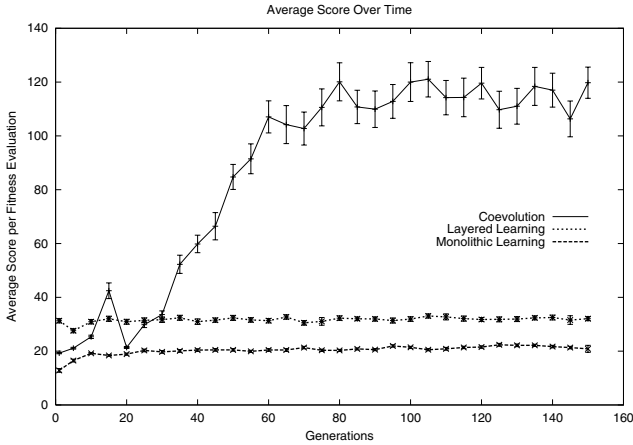


Fig. 8. Average score per fitness evaluation for the best run of each method over generations when the taker moves at 100% speed. These results demonstrate that task decomposition is important in this domain and that coevolution can effectively learn the resulting subtasks.

5 Discussion

The results described above verify that given a suitable task decomposition neuro-evolution can learn a complex, multi-agent control task that is too difficult to learn monolithically. Given such a decomposition, layered learning developed a successful controller, though the less-constrained coevolutionary approach performed significantly better.

By placing fewer restrictions on the solution space, coevolution benefits from greater flexibility, which may contribute to its strong performance. Since coevolution trains every sub-behavior in the target environment, the components have the opportunity to react to each other’s behavior and adjust accordingly. In layered learning, by contrast, we usually need to construct a special training environment for most layers. If any of those environments fail to capture a key aspect of the target domain, the resulting components may be sub-optimal. For example, the interceptor trained by layered learning is evaluated only by how quickly it can reach the ball. In keepaway, however, a good interceptor will approach the ball from the side to make the agent’s next pass easier. Since the coevolving interceptor learned along with the passer, it was able to learn this superior behavior, while the layered interceptor just approached the ball directly. Though it is possible to adjust the layered interceptor’s fitness function to encourage this indirect approach, it is unlikely that a designer would know *a priori* that such behavior is desirable.

The success of coevolution in this domain suggests that we can learn complex tasks simply by providing neuro-evolution with a high-level strategy. However, we suspect that in extremely difficult tasks, the solution space will be too large

for coevolution to search effectively given current neuro-evolution techniques. In these cases, the hierarchical features of layered learning, by greatly reducing the solution space, may prove essential to a successful learning system.

Layered learning and coevolution are just two points on a spectrum of possible methods which differ with respect to how aggressively they constrain learning. At one extreme, the monolithic approach tested in this paper places very few restrictions on learning. At the other extreme, layered learning confines the search by directing each component to a specific sub-goal. The layered and co-evolutionary approaches can be made arbitrarily more constraining by replacing some of the components with hand-coded behaviors. Similarly, both methods can be made less restrictive by requiring them to learn a decision tree, rather than giving them a hand-coded one.

6 Conclusion and Future Work

In this paper we verify that neuro-evolution can master keepaway, a complex, multi-agent control task. We also show that decomposing the task is more effective than training a monolithic controller for it. Our experiments demonstrate that the more flexible coevolutionary approach learns better agents than the layered approach in this domain.

In ongoing research we plan to further explore the space between unconstrained and highly constrained learning methods. In doing so, we hope to shed light on how to determine the optimal method for a given task. Also, we plan to test both the layered and coevolutionary approaches in more complex domains to better assess the potential of these promising methods.

Acknowledgments. This research was supported in part by the National Science Foundation under grant IIS-0083776, and the Texas Higher Education Coordinating Board under grant ARP-0036580476-2001.

References

1. T. Balch. Teambots domain: Soccerbots, 2000.
<http://www-2.cs.cmu.edu/~trb/TeamBots/Domains/SoccerBots>.
2. F. Gomez and R. Miikkulainen. Incremental evolution of complex general behavior. *Adaptive Behavior*, 5:317–342, 1997.
3. F. Gomez and R. Miikkulainen. Solving non-Markovian control tasks with neuroevolution. Denver, CO, 1999.
4. F. Gomez and R. Miikkulainen. Learning robust nonlinear control with neuroevolution. Technical Report AI01-292, The University of Texas at Austin Department of Computer Sciences, 2001.
5. T. Haynes and S. Sen. Evolving behavioral strategies in predators and prey. In G. Weiß and S. Sen, editors, *Adaptation and Learning in Multiagent Systems*, pages 113–126. Springer Verlag, Berlin, 1996.

6. W. H. Hsu and S. M. Gustafson. Genetic programming and multi-agent layered learning by reinforcements. In *Genetic and Evolutionary Computation Conference*, New York, NY, July 2002.
7. I. Noda, H. Matsubara, K. Hiraki, and I. Frank. Soccer server: A tool for research on multiagent systems. *Applied Artificial Intelligence*, 12:233–250, 1998.
8. A. D. Pietro, L. While, and L. Barone. Learning in RoboCup keepaway using evolutionary algorithms. In *GECCO 2002: Proceedings of the Genetic and Evolutionary Computation Conference*, pages 1065–1072, New York, 9-13 July 2002. Morgan Kaufmann Publishers.
9. M. A. Potter and K. A. D. Jong. Cooperative coevolution: An architecture for evolving coadapted subcomponents. *Evolutionary Computation*, 8:1–29, 2000.
10. C. D. Rosin and R. K. Belew. Methods for competitive co-evolution: Finding opponents worth beating. In *Proceedings of the Sixth International Conference on Genetic Algorithms*, pages 373–380, San Mateo, CA, July 1995. Morgan Kaufman.
11. J. D. Schaffer, D. Whitley, and L. J. Eshelman. Combinations of genetic algorithms and neural networks: A survey of the state of the art. In D. Whitley and J. Schaffer, editors, *International Workshop on Combinations of Genetic Algorithms and Neural Networks (COGANN-92)*, pages 1–37. IEEE Computer Society Press, 1992.
12. P. Stone. *Layered Learning in Multiagent Systems: A Winning Approach to Robotic Soccer*. MIT Press, 2000.
13. P. Stone, (ed.), M. Asada, T. Balch, M. Fujita, G. Kraetzschmar, H. Lund, P. Scerri, S. Tadokoro, and G. Wyeth. Overview of RoboCup-2000. In *RoboCup-2000: Robot Soccer World Cup IV*. Springer Verlag, Berlin, 2001.
14. P. Stone and R. S. Sutton. Scaling reinforcement learning toward RoboCup soccer. In *Proceedings of the Eighteenth International Conference on Machine Learning*, pages 537–544. Morgan Kaufmann, San Francisco, CA, 2001.
15. P. Stone and R. S. Sutton. Keepaway soccer: a machine learning testbed. In *RoboCup-2001: Robot Soccer World Cup V*. Springer Verlag, Berlin, 2002.
16. P. Stone and M. Veloso. Layered learning. In *Machine Learning: ECML 2000*, pages 369–381. Springer Verlag, Barcelona, Catalonia, Spain, May/June 2000. Proceedings of the Eleventh European Conference on Machine Learning (ECML-2000).
17. S. Whiteson and P. Stone. Concurrent layered learning. In *Second International Joint Conference on Autonomous Agents and Multiagent Systems*, July 2003. To appear.
18. C. H. Yong and R. Miikkulainen. Cooperative coevolution of multi-agent systems. Technical Report AI01-287, The University of Texas at Austin Department of Computer Sciences, 2001.

A New Method of Multilayer Perceptron Encoding

Emmanuel Blindauer and Jerzy Korczak

Laboratoire des Sciences de l'Image, de l'Informatique et de la Télédétection,
UMR7005, CNRS, 67400 Illkirch, France. {blindauer,jjk}@lsiit.u-strasbg.fr

1 Evolving Neural Networks

One of the central issues in neural network research is how to find an optimal MultiLayer Perceptron architecture. The number of neurons, their organization in layers, as well as their connection scheme have a considerable influence on network learning, and on the capacity for generalization [7]. A solution to find out these parameters is needed: The neuro-evolution ([1,2,4,5]).

The novelty is to emphasize the network performance aspects, and the network simplification achieved by reducing the network topology. All these genetic manipulations on the network architecture should not decrease the neural network performance.

2 Network Representation and Encoding Schemes

The main goal of an encoding scheme is to represent neural networks in a population as a collection of chromosomes. There are many approaches to genetic representation of neural networks [4], [5]. Classical method use to encode the network topology into a single string. But frequently, for large-size problems, these methods do not generate satisfactory results: computing new weights to get satisfactory networks is very costly.

A new encoding method based on the matrix encoding is proposed: A matrix where every element represents a weight of the neural network.

Several operators for a genotype have been proposed: crossover operators and mutation operators. For the classical crossover operation, a new matrix is created from two splitted matrix: the offspring get two different parts, one from each parent. This can be considered as the one point crossover, in a two dimension space A second crossover operator is defined: an exchange of a sub-matrix between the parents is done. For the mutation, several operators are available. The first is the ablation operator. Setting one or several zero in the matrix, we are removing these connections. Setting to zero a partial row or column, we delete several incoming or outgoing connections from the neuron. The second is the grown operator: connection are added. Again, we can control where the connections are added, and know if a neuron is fully connected or not.

With these operators, as matrix elements are the weights of the network, some learning is required to get a new optimal network. As only a few weights have changed, the learning will be faster.

3 Experimentation

The performance have been evaluated on several classical problems. These case studies have been chosen based on the growing complexity of the problem to solve. Each population had 200 individuals. For each individual, 100 epochs were carried out for training. For the genetics parameters, the crossover percent is set to 80%, with a elitist model. 5% of the population can fall under mutation.

Compared with other results from [3], this new method has shown the best, not only in term of network complexity, but also in quality of learning

Table 1. Results of experimentations

	XOR	Parity 3	Parity 4	Parity 5	Heart	Sonar
Number of hidden neurons	2	3	5	8	12	30
Number of connections	6	11	23	38	354	1182
Number of epochs (error)	13	23	80	244	209 (9%)	120 (13%)

4 Conclusion

The experiments have confirmed that, firstly by encoding the network topology and weights the search space is affined; secondly, by the inheritance of connection weights, the learning stage is speeded up considerably. The presented method generates efficient networks in a shorter time compared to actual methods. The new encoding scheme improves the effectiveness of evolutionary process: weights of the neural network included in the genetic encoding scheme and good genetics operators give acceptable results.

References

1. J. Korczak and E. Blindauer, An Approach to Encode Multilayer Perceptrons, [In] *Proceedings of the International Conference on Artificial Neural Networks*, 2002
2. E.Cantú-Paz, C.Kamath, Evolving Neural Networks For The Classification of Galaxies, [In] *Proceedings of the Genetic and Evolutionary Computation Conference*, 2002
3. M.A. Grönroos, *Evolutionary Design Neural Networks*, PhD thesis, Department of Mathematical Sciences, University of Turku, 1998.
4. F. Gruau, *Neural networks synthesis using cellular encoding and the genetic algorithm*, PhD thesis, LIP, Ecole Normale Supérieure, Lyon, 1992.
5. H. Kitano, Designing neural networks using genetic algorithms with graph generation system, *Complex Systems*, 4: 461–476, 1990.
6. F. Radlinski, *Evolutionary Learning on Structured Data for Artificial Neural Networks*, MSC Thesis, Dep. of Computer Science Australian National University, 2002
7. X. Yao, Evolving artificial neural networks. *Proceedings of the IEEE*, 1999.

An Incremental and Non-generational Coevolutionary Algorithm

Ramón Alfonso Palacios-Durazo¹ and Manuel Valenzuela-Rendón²

¹ Lumina Software, apd@luminasoftware.com

<http://www.luminasoftware.com/apd>

Washington 2825 Pte C.P. 64040, Monterrey N.L., Mexico

² ITESM, Monterrey

Centro de Sistemas Inteligentes

valenzuela@itesm.mx,

<http://www-csi.mty.itesm.mx/~mvalenzu>

C.P. 64849 Monterrey, N.L., Mexico

The central idea of coevolution lies in the fact that the fitness of an individual depends on its performance against the current individuals of the opponent population. However, coevolution has been shown to have problems [2,5]. Methods and techniques have been proposed to compensate the flaws in the general concept of coevolution [2]. In this article we propose a different approach to implementing coevolution, called *incremental coevolutionary algorithm* (ICA) in which some of these problems are solved by design. In ICA, the importance of the coexistence of individuals in the same population is as important as the individuals in the opponent population. This is similar to the problem faced by learning classifier systems (LCSs) [1,4]. We take ideas from these algorithms and put them into ICA.

In a coevolutionary algorithm, the fitness landscape depends on the opponent population, therefore it changes every generation. The individuals selected for reproduction are those more promising to perform better against the fitness landscape represented by the opponent population. However, if the complete population of parasites and hosts are recreated in every generation, the offspring of each new generation face a fitness landscape unlike the one they were bred to defeat. Clearly, a generational approach to coevolution can be too disruptive. Since the fitness landscape changes every generation, it also makes sense to incrementally adjust the fitness of individuals in each one. These two ideas define the main approach of the ICA: the use of a non-generational genetic algorithm and the incremental adjustment of the fitness estimation of an individual. The formal definition of ICA can be seen in figure 1.

ICA has some interesting properties. First of all, it is not generational. Each new individual faces a similar fitness landscape than its parents. The fitness landscape changes gradually, allowing an arms race to occur. Since opponents are chosen proportional to their fitness, an individual has a greater chance of facing good opponents. If a particular strength is found in a population, individuals that have it will propagate and will have a greater probability of coming into competition (both because more individuals carry the strength, and because a

greater fitness produces a higher probability of being selected for competition). If the population overspecializes, another strength will propagate to maintain balance. Thus, a natural sharing occurs.

```
(*Define  $A(x, f)$ *) :  $A(x, f) = \tanh(x/f)$ 
Generate random host and parasite population
Initialize fitness of all parasites  $S_p \leftarrow M_p/C_p$  and hosts  $S_h \leftarrow A_s/M_a$ 
repeat
  (* Competition cycle*)
  for  $c \leftarrow 1$  to  $N_c$ 
    Select parasite  $p$  and host  $h$  proportionally to fitness.
     $error \leftarrow \text{abs}(\text{Result of competition between } h \text{ and } p)$ 
     $S_p \leftarrow S_p + M_p A(error, E_{error}) - C_p S_p(t)$ 
     $S_h \leftarrow S_h + A_s (1 - A(error, E_{error})) - M_a S_h$ 
  end-for  $c$ 
  (* 1 step of a GA*)
  Select two parasite parents ( $p_1$  and  $p_2$ ) proportionally to  $S_p$ 
  Create new individual  $p_0$  by doing crossover and mutation
   $S_{p0} \leftarrow (S_{p1} + S_{p2})/2$ 
  Delete parasite with worst fitness and substitute with  $p_0$ 
  Repeat above for host population
until termination criteria met
```

Fig. 1. Incremental coevolutionary algorithm

The equations for incrementally adjusting fitness can be proven to be stable doing an analysis similar to the one used for LCSs [3]. ICA was tested finding trigonometric identities and was found to be robust, able to generate specialization niches and to consistently outperform traditional genetic programming.

References

1. John Holland. Escaping brittleness: The possibilities of general-purpose learning algorithms applied to parallel rule-based systems. *Machine Learning: An Artificial Intelligence Approach*, 2, 1986.
2. Christopher D. Rosin and Richard K. Belew. Methods for competitive co-evolution: Finding opponents worth beating. In Larry Eshelman, editor, *Proceedings of the Sixth International Conference on Genetic Algorithms*, pages 373–380, San Francisco, CA, 1995. Morgan Kaufmann.
3. Manuel Valenzuela-Rendón. *Two Analysis Tools to Describe the Operation of Classifier Systems*. PhD thesis, The University of Alabama, Tuscaloosa, Alabama, 1989.
4. Manuel Valenzuela-Rendón and E. Uresti-Charre. A nongenerational genetic algorithm for multiobjective optimization. In *Proceedings of the Seventh International Conference on Genetic Algorithms*, pages 658–665. Morgan Kaufmann, 1997.
5. Richard A. Watson and Jordan B. Pollack. Coevolutionary dynamics in a minimal substrate. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, pages 702–709, San Francisco, California, USA, 7-11 2001. Morgan Kaufmann.

Coevolutionary Convergence to Global Optima

Lothar M. Schmitt

The University of Aizu, Aizu-Wakamatsu City, Fukushima Prefecture 965-8580, Japan
lothar@u-aizu.ac.jp

Abstract. We discuss a theory for a realistic, applicable scaled genetic algorithm (GA) which converges asymptotically to global optima in a coevolutionary setting involving two species. It is shown for the first time that coevolutionary arms races yielding global optima can be implemented successfully in a procedure similar to simulated annealing.

Keywords: Coevolution; convergence of genetic algorithms; simulated annealing; genetic programming.

In [2], the need for a theoretical framework for coevolutionary algorithms *and* possible convergence theorems in regard to coevolutionary optimization (“arms races”) was pointed out. Theoretical advance for coevolutionary GAs involving two types of creatures seems very limited thus far. [6] largely fills this void¹ in the case of a fixed division of the population among the two species involved even though there is certainly room for improvement. For a setting involving two types of creatures, [6] satisfies *all* goals advocated in [1, p. 270] in regard to finding a theoretical framework for scaled GAs similar to simulated annealing.

[4,5] contain recent substantial advances in theory of coevolutionary GAs for competing agents/creatures of a single type. In particular, the coevolutionary global optimization problem is solved under the condition that (a group of) agents exist that are strictly superior in every population they reside in.

Here and in [6], we continue to use the *well-established notation* of [3,4,5]. The setup considers two sets of creatures $\mathcal{C}^{(0)}$ and $\mathcal{C}^{(1)}$. Elements of $\mathcal{C}^{(0)}$ can, *e.g.*, be thought of as sorting programs while $\mathcal{C}^{(1)}$ can be thought of as unsorted tuples. The two types of creatures $\mathcal{C}^{(j)}$, $j \in \{0, 1\}$, involved in the setup of the coevolutionary GA are being encoded as finite-length strings over arbitrary-size alphabets $\mathcal{A}_j$. Creatures $c \in \mathcal{C}^{(0)}$, $d \in \mathcal{C}^{(1)}$ are evaluated by a duality $\langle c, d \rangle \in \mathbb{R}$. In case of the above example, this expression may represent execution time of a sorting program c on an unsorted tuple d . Any population p is a tuple consisting of $s_0 \geq 4$ creatures of $\mathcal{C}^{(0)}$ followed by $s_1 \geq 4$ creatures of $\mathcal{C}^{(1)}$. This fixed division of the population is done here simply for practical purposes but is, in effect, in accordance with the evolutionary stable strategy in evolutionary game theory. In particular, the model in [6] does not refer to the multi-set model [7].

¹ Possibly, there exist significant theoretical results unknown to the author. Referee 753 claims in regard to [6]: “this elaborate mathematical framework that doesn’t illuminate anything we don’t already know” without giving further reference.

The GA considered in [6] employs *very common* GA operators which are given by detailed, almost procedural definitions including explicit annealing schedules: multiple-spot mutation, practically any known crossover, and scaled proportional fitness selection. Thus, the GA considered in [6] is *standard* and by no means “out of the blue sky”. Work by the authors of [1] and [4, Thms. 8.2–6] show that the annealing procedure considered in [6] is *absolutely necessary for convergence to global optima* and not “highly contrived”. The mutation operator allows for a scalable compromise on the alphabet level between a neighborhood-based search and pure random change (the latter as in [4, Lemma 3.1]). The population-dependent fitness function is defined as follows: if $p = (c_1, \dots, c_{s_0}, d_1, \dots, d_{s_1})$, $\varphi_1 = \pm 1$, then $f(d_i, p) = \exp(\varphi_1 \sum_{\sigma=1}^{s_0} \langle c_\sigma, d_i \rangle)$. The fitness function is defined similarly for $c_1, \dots, c_{s_0}$. The factors $\varphi_{0,1} = \pm 1$ are used to adjust whether the two types of creatures have the same or opposing goals. Referring to the above example, one would set $\varphi_0 = -1$ and $\varphi_1 = 1$ since good sorting programs aim for a short execution time while ‘difficult’ unsorted tuples aim for a long execution time. The fitness function is then scaled with logarithmic growth in the exponent as in [4, Thm. 8.6] or [5, Thm. 3.4.1] with similar lower bounds for the factor $B > 0$ determining the growth.

Under the assumption that a group of *globally strictly maximal* creatures exists that are evaluated superior in any population they reside in, an analogue of [4, Thm. 8.6] [5, Thm. 3.4.1] with similar restriction on population size is shown in [6]. In particular, the coevolutionary GA in [6] is strongly ergodic and converges to a probability distribution over uniform populations containing only globally strictly maximal creatures. [6] is available from this author. As indicated above, this author finds the concerns of referees unacceptable to a large degree.

References

1. Davis, T.E.; Principe, J.C.: A Markov Chain Framework for the Simple GA. *Evol. Comput.* **1** (1993) 269–288
2. DeJong, K.: Lecture on Coevolution. IN: Beyer H.-G. *et al.* (chairs): Seminar ‘*Theory of Evolutionary Computation 2002*’, Max Planck Inst. Comput. Sci. Conf. Cent., Schloß Dagstuhl, Saarland, Germany (2002)
3. Schmitt, L.M. *et al.*: Linear Analysis of Genetic Algorithms. *Theoret. Comput. Sci.* **200** (1998) 101–134
4. Schmitt, L.M.: Theory of Genetic Algorithms. *Theoret. Comput. Sci.* **259** (2001) 1–61
5. Schmitt, L.M.: Asymptotic Convergence of Scaled Genetic Algorithms to Global Optima —A gentle introduction to the theory—. IN: Menon A. (ed.). *The Next Generation Research Issues in Evolutionary Computation*. (in preparation), Kluwer Ser. in Evol. Comput. (Goldberg D.E., ed.). Kluwer, Dordrecht, The Netherlands (2003) (to appear)
6. Schmitt, L.M.: Coevolutionary Convergence to Global Optima. Tech. Rep. 2003-2-001, The University of Aizu, Aizu-Wakamatsu, Japan (2003) 1–12
7. Vose M.D.: *The Simple Genetic Algorithm: Foundations and Theory*. MIT Press, Cambridge, MA, USA (1999)

Generalized Extremal Optimization for Solving Complex Optimal Design Problems

Fabiano Luis de Sousa¹, Valeri Vlassov¹, and Fernando Manuel Ramos²

¹ Instituto Nacional de Pesquisas Espaciais – INPE/DMC – Av. dos Astronautas, 1758
12227-010 São José dos Campos, SP – Brazil
{fabiano, vlassov}@dem.inpe.br

² Instituto Nacional de Pesquisas Espaciais – INPE/LAC – Av. dos Astronautas, 1758
12227-010 São José dos Campos, SP – Brazil
fernando@lac.inpe.br

Recently, Boettcher and Percus [1] proposed a new optimization method, called Extremal Optimization (EO), inspired by a simplified model of natural selection developed to show the emergence of Self-Organized Criticality (SOC) in ecosystems [2]. Although having been successfully applied to hard problems in combinatorial optimization, a drawback of the EO is that for each new optimization problem assessed, a new way to define the fitness of the design variables has to be created [2]. Moreover, to our knowledge it has been applied so far to combinatorial problems with no implementation to continuous functions.

In order to make the EO easily applicable to a broad class of design optimization problems, Sousa and Ramos [3,4] have proposed a generalization of the EO that was named the Generalized Extremal Optimization (GEO) method. It is of easy implementation, does not make use of derivatives and can be applied to unconstrained or constrained problems, non-convex or disjoint design spaces, with any combination of continuous, discrete or integer variables. It is a global search meta-heuristic, as the Genetic Algorithm (GA) and the Simulated Annealing (SA), but with the *a priori* advantage of having only one free parameter to adjust. Having been already tested on a set of test functions, commonly used to assess the performance of stochastic algorithms, the GEO proved to be competitive to the GA and the SA, or variations of these algorithms [3,4].

The GEO method was devised to be applied to complex optimization problems, such as the optimal design of a heat pipe (HP). This problem has difficulties such as an objective function that presents design variables with strong non-linear interactions, subject to multiple constraints, being considered unsuitable to be solved by traditional gradient based optimization methods [5]. To illustrate the efficacy of the GEO on dealing with such kind of problems, we used it to optimize a HP for a space application with the goal of minimizing the HP's total mass, given a desirable heat transfer rate and boundary conditions on the condenser. The HP uses a mesh type wick and is made of Stainless Steel. A total of 18 constraints were taken into account, which included operational, dimensional and structural ones. Temperature dependent fluid properties were considered and the calculations were done for steady state conditions, with three fluids being considered as working fluids: ethanol, methanol and ammonia. Several runs were performed under different values of heat transfer

rate and temperature at the condenser. Integral optimal characteristics were obtained, which are presented in Figure 1.

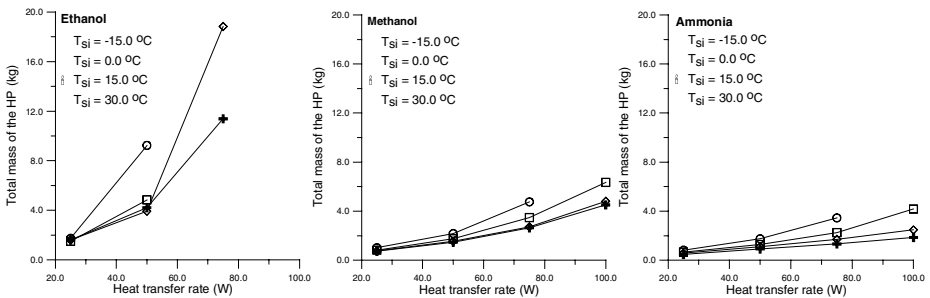


Fig. 1. Minimum HP mass found for ethanol, methanol and ammonia, at different operational conditions.

It can be seen from these results, that for moderate heat transfer rates (up to 50 W), the ammonia and methanol HPs display similar results in terms of optimal mass, while for high heat transfer rates (as for $Q = 100$ W), the HP filled with ammonia shows considerably better performance. In practice, this means that for applications which require the transport of moderate heat flow rates, cheaper methanol HPs can be used, whereas at higher heat transport rates, the ammonia HP should be utilized. It can be also seen, that the higher the heat to be transferred, the higher the HP total mass. Although this is an expected result, the apparent non-linearity of the HP mass with Q (more pronounced as the temperature on the external surface of the condenser T_{si} is increased), means that for some applications there is a theoretical possibility that the use of two HPs of a given heat transfer capability can yield a better performance, in terms of mass optimization, than the use of a single HP with double capability. This non-linearity of the optimal characteristics has an important significance in design practice and, thus, should be further investigated. These results highlight the potential of the GEO to be used as a design tool. In fact, it can be said that the GEO method is a good candidate to be incorporated to the designer's tools suitcase.

References

1. Boettcher, S. and Percus, A. G.: Optimization with Extremal Dynamics, *Physical Review Letters*, Vol. 86 (2001) 5211–5214.
2. Bak, P. and Sneppen, K., "Punctuated Equilibrium and Criticality in a Simple Model of Evolution", *Physical Review Letters*, Vol. 71, Number 24, pp. 4083–4086, 1993.
3. Sousa, F.L. and Ramos, F.M.: Function Optimization Using Extremal Dynamics. Proceedings of the 4th International Conference on Inverse Problems in Engineering, Rio de Janeiro, Brazil, (2002).
4. Sousa, F.L., Ramos, F.M., Paglione, P. and Girardi, R.M.: A New Stochastic Algorithm for Design Optimization. Accepted for publication in the *AIAA Journal*.
5. Rajesh, V.G. and Ravindran K.P.: Optimum Heat Pipe Design: A Nonlinear Programming Approach. *International Communications in Heat and Mass Transfer*, Vol. 24, No. 3, (1997) 371–380.

Coevolving Communication and Cooperation for Lattice Formation Tasks

Jekanthan Thangavelautham, Timothy D. Barfoot, and
Gabriele M.T. D'Eleuterio

Institute for Aerospace Studies University of Toronto
4925 Dufferin Street, Toronto, Ontario, Canada, M3H 5T6
thangav@ecf.utoronto.ca, {tim.barfoot,gabriele.deleuterio}@utoronto.ca

Abstract. Reactive multi-agent systems are shown to coevolve with explicit communication and cooperative behavior to solve lattice formation tasks. Comparable agents that lack the ability to communicate and cooperate are shown to be unsuccessful in solving the same tasks. The control system for these agents consists of identical cellular automata lookup tables handling communication, cooperation and motion subsystems.

1 Introduction

In nature, social insects such as bees, ants and termites collectively manage to construct hives and mounds, without any centralized supervision [1]. The agents in our simulation are driven by a decentralized control system and can take advantage of communication and cooperation strategies to produce a desired ‘swarm’ behavior. A decentralized approach offers some inherent advantages, including fault tolerance, parallelism, reliability, scalability and simplicity in agent design [2].

Our initial test has been to evolve a homogenous multi-agent system able to construct simple lattice structures. The lattice formation task involves redistributing a preset number of randomly scattered objects (blocks) in a 2-D grid world into a desired lattice structure. The agents move around the grid world and manipulate blocks using reactive control systems with input from simulated vision sensors, contact sensors and inter-agent communication. A global consensus is achieved when the agents arrange the blocks into one indistinguishable lattice structure (analogous to the heap formation task [3]). The reactive control system triggers one of four basis behaviors, namely move, manipulate object, pair-up (link) and communicate based on the state of numerous sensors.

2 Results and Discussion

For the GA run, the 2-D world size was a 16×16 grid with 24 agents, 36 blocks and a training time of 3000 time steps. Shannon’s entropy function was used as a fitness evaluator for the 3×3 tilling pattern task. After 300 generations, the GA run converged to a reasonably high average fitness value (about 99). The

agents learn to explicitly cooperate within the first 5-10 generations. From our findings, it appears the evolved solution perform well for much larger problem sizes of up to 100×100 grids as expected, due to our decentralized approach.

Within a coevolutionary process it would be expected for competing populations (or subsystems) to spur an 'arms race' [4]. The steady convergence in physical behaviors appears to exhibit this process. The communication protocol that had evolved from the GA run consists of a set of non-coherent signals with a mutually agreed upon meaning. A comparable agent was developed which lacked the ability to communicate and cooperate for solving the 3×3 tiling pattern task. Each agent had 7 vision sensors, which meant 4374 lookup table entries compared to the 349 entries for the agent discussed earlier. After having modified various genetic parameters, it was found the GA run never converged. For this particular case, techniques employing communication and cooperation have reduced the lookup table size by a factor 12.5 and have made the GA run computational feasible.

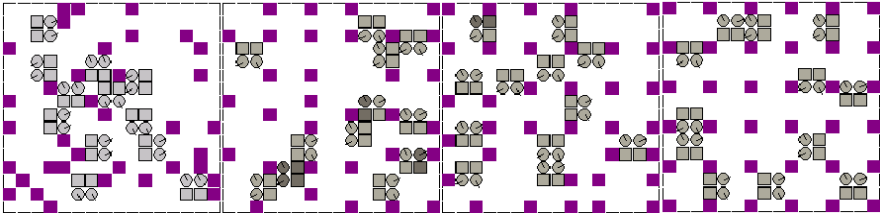


Fig. 1. Snapshot of the system taken at various time steps (0, 100, 400, 1600). The 2-D world size is a 16×16 grid with 28 agents and 36 blocks. At time step 0, neighboring agents are shown 'unlinked' (light gray) and by 100 time steps all 28 agents manage to 'link' (gray or dark gray). Agents shaded in dark gray carry a block. After 1600 time steps (far right), the agents come to a consensus and form one lattice structure.

References

1. Kube, R., Zhang, H.: Collective Robotics Intelligence : From Social Insects to robots. In Proc. Of Simulation of Adaptive Behavior (1992) 460–468
2. Cao, Y.U., Fukunaga, A., Kahng, A. : Cooperative Mobile Robotics : Antecedents and Directions. : In Autonomous Robots, Vol.4. Kluwer Academic Pub., Boston (1997) 1–23
3. Barfoot, T., D'Eleuterio, G.M.T.: An Evolutionary Approach to Multi-agent Heap Formation. In proceedings of the Congress on Evolutionary Computation (1999)
4. Koza, J.R.: Genetic Programming: On the Programming of Computers by Means of Natural Selection. The MIT Press. Cambridge, MA, (1992)
5. J. Thangavelautham, T.D. Barfoot, G. M. T. D'Eleuterio: Coevolving Communication and Cooperation for Lattice Formation Tasks. University of Toronto Institute for Aerospace Studies Technical Report. Toronto, Ont. (2003)

Efficiency and Reliability of DNA-Based Memories

Max H. Garzon, Andrew Neel, and Hui Chen

Computer Science, University of Memphis
373 Dunn Hall, Memphis, TN 38152-3240
{mgarzon, aneel, hchen2}@memphis.edu

Abstract. Associative memories based on DNA-affinity have been proposed [2]. Here, the performance, efficiency, reliability of DNA-based memories is quantified through simulations *in silico*. Retrievals occur reliably (98%) within very short times (milliseconds) despite the randomness of the reactions and regardless of the number of queries. The capacity of these memories is also explored in practice and compared with previous theoretical estimates. Advantages of implementations of the same type of memory in special purpose chips *in silico* is proposed and discussed.

1 Introduction

DNA oligonucleotides have demonstrated to be a feasible and useful medium for computing applications since Adleman's original work [1], which created a field now known as biomolecular computing (BMC). Potential applications range from increasing speed through massively parallel computations [13], to new manufacturing techniques in nanotechnology [18], and to the creation of memories that can store very large amounts of data and fit into minuscule spaces [2], [15]. The apparent enormous capacity of DNA (over million fold compared to conventional electronic media) and the enormous advances in recombinant biotechnology to manipulate DNA *in vitro* in the last 20 years make this approach potentially attractive and promising. Despite much work in the field, however, difficulties still abound in bringing these applications to fruition due to inherent difficulties in orchestrating a large number of individual molecules to perform a variety of functions in the environment of virtual test tubes, where the complex machinery of the living cell is no longer present to organize and control the numerous errors pulling computations by molecular populations away from their intended targets.

In this paper, we initiate a quantitative study of the potential, limitations, and actual capacity of memories based or inspired by DNA. The idea of using DNA to create large associative memories goes back to Baum [2], where he proposed to use DNA recombination as the basic mechanism for content-addressable storage of information so that retrieval could be accomplished using the basic mechanism of DNA hybridization affinity. Content is to be encoded in single stranded molecules in solution (or their complements.) Queries can be obtained by dropping in the tube a DNA primer

Watson-Crick complement of the (partial) information known about a particular record using the same coding scheme as in the original memory, appropriately marked (e.g., using magnetic beads, or fluorescent tags). Retrieval is completed by extension and/or retrieval (e.g., by sequencing) of any resulting double strands after appropriate reaction times have been allowed for hybridization to take effect. As pointed out by Baum [2], and later Reif & LaBean [15], many questions need to be addressed before an associative memory based on this idea can be regarded as feasible, let alone actually built.

Further methods were proposed in [15] for input/output from/to databases represented in wet DNA (such as genomic information obtained from DNA-chip optical readouts, or synthesis of strands based on such output) and suggested methods to improve the capabilities and performance of the queries of such DNA-based memories. The proposed hybrid methods, however, require major pre-processing of the entire database contents (through clustering and vector quantization) and post-processing to complete the retrieval by the DNA memory (based on the identification of the clusters centers.) This is a limitation when the presumed database approaches the expected sizes to be an interesting challenge to conventional databases, or when the data already exists in wet DNA, because of the prohibitive (and sometimes even impossible) cost of the transduction process to and from electronics. Inherent issues in the retrieval *per se*, such as the reliability of the retrieval *in-vitro* and the appropriate concentrations for optimal retrieval times and error rates remain unclear.

We present an assessment of the efficiency and reliability of queries in DNA-based memories in Section 3, after a description of the experimental design and the data collected for this purpose in Section 2. In Section 3, we also present very preliminary estimates of their capacity. Finally, section 4 summarizes the results and discusses the possibility of building analogous memories *in silico* inspired by the original ideas *in vitro*, as suggested by the experiments reported here. A preliminary analysis of some of these results has been presented in [7], but here we present further results and a more complete analysis.

2 Experimental Design

The experimental data used in this paper has been obtained by simulations in the virtual test tube of Garzon et al [9]. Recently, driven by efficiency and reliability considerations, the ideas of BMC have been implemented *in silico* by using computational analogs of DNA and RNA molecules [8]. Recent results show that these protocols produce results that closely resemble, and in many cases are indistinguishable from, the protocols they simulate in wet tubes [7]. For example, Adleman's experiment has been experimentally reproduced and scaled in virtual test tubes with random graphs of up to 15 vertices while producing results correct with no probability of a false positive error and a probability of a false negative of at most 0.4%. Virtual test tubes have also matched very well the results obtained *in vitro* by more elaborate and newer protocols, such as the selection protocol for DNA library design of Deaton et Al. [4]. Therefore,

there is good evidence that virtual test tubes provide a reasonable and reliable estimate of the events in wet tubes (see [7] for a more detailed discussion.)

Virtual test tubes thus can serve as a reasonable pre-requisite methodology to estimate the performance and experimental validation prior to construction of such a memory, a validation step that is now standard in the design of conventional solid-state memories. Moreover, as will be seen below in the discussion of the results, virtual test tubes offer a much better insight into the nature of the reaction *kinetics* than corresponding experiments *in vitro*, which, when possible (such as Cot curves to measure the diversity of a DNA pool), incur much larger cost and effort.

2.1 Virtual Test Tubes

Our experimental runs were implemented using the virtual test tube *Edna* of Garzon et al. [7],[8],[9] that simulates BMC protocols *in silico*. *Edna* provides an environment where DNA analogs can be manipulated much more efficiently, can be programmed and controlled much more easily, at much lower costs, and produce comparable results to those obtained in a real test tube [7]. Users simply need to create object-oriented programming classes (in C++) specifying the objects to be used and their interactions. The basic design of the entities that were put in *Edna* represent each nucleotide within the DNA as a single character and the entire strand of DNA as a string, which may contain single- or double-stranded sections, bulges, and loops or higher secondary structures. An unhybridized strand represents a strand of DNA from the 5'-end to the 3'-end. These strands encode library records in the database, or queries containing partial information that identify the records to be retrieved.

The interactions among objects in *Edna* represent chemical reactions by hybridization and ligation resulting in new objects such as dimers, duplexes, double strands, or more complicated complexes. They can result in one or both entities being destroyed and a new entity possibly being created. In our case, we wanted to allow the entities that matched to hybridize to each other to effect a retrieval, per Baum's design [2]. *Edna* simulates the reactions in successive *iterations*. One iteration moves the objects randomly in the tube's container (the RAM really) and updates their status according to the specified interactions with neighbor objects, based on proximity parameters that can be varied within the interactions. The hybridization reactions between strands were performed according to the h-measure [8] of hybridization likelihood. Hybridization was allowed if the h-measure was under a given threshold, which is the number of mismatches allowed (including frame-shifts) and so roughly codes for stringency in the reaction conditions. A threshold of zero enforces perfect matches in retrieval, whereas a larger value permits more flexible and associative retrieval. These requirements essentially ensured good enough matches along the sections of the DNA that were relevant for the associative recall.

The efficiency of the test tube protocols (in our case, retrievals) can be measured by counting the number of iterations necessary to complete the reactions or achieve the desired objective; alternatively, one can measure the wall clock time. The *number of iterations* taken until a match is found has the advantage of being indifferent to the

speed of the machine(s) running the experiment. This intrinsic measure was used because one iteration is representative of a unit of real-time for *in vitro* experiments. The relationship between simulation results in simulation and equivalent results *in vitro* has been discussed in [7]. Results of the experiments *in silico* can be used to yield realistic estimates of those *in vitro*. Essentially, one iteration of the test tube corresponds to the reaction time of one hybridization in the wet tube, which is of the order of one millisecond [17]. However, the number of iterations cannot be a complete picture because iterations will last longer as more entities are put in the test tube. For this reason, *processor time* (wall clock) was also measured. The wall clock time depends on the speed and power of the machine(s) running *Edna* and ranged anywhere from seconds to days for the single processors and 16 PC cluster that were used to run the experiments used below.

2.2 Libraries and Queries

We assume we have at our disposal a library of non-cross hybridizing (nxh) strands representing the records in the databases. The production of such large libraries has been addressed elsewhere [4], [10]. Well-chosen DNA word designs that will make this perfectly possible in large numbers of DNA strands directly, even in real test tubes, will likely be available within a short time. The exact size of such a library will be discussed below. The nxh property of the library will also ensure that retrievals will be essentially noise-free (no false positives), module the flexibility built into the retrieval parameters (here h-distance). We will also assume that a record may also contain an additional segment (perhaps double-stranded [2]) encoding supplementary information beyond the label or segment actively used for associative recall, although this is immaterial for assumptions and results in this paper. The library is assumed to reside in the test tube, where querying takes place.

Queries are strings objects encoding, and complementary of, the available information to be searched for. The selection operation uses probes to mark strands by hybridizing part of the probe with part of the “probed” strand. The number of unique strands available to be probed is, in principle, the entire library, although we consider below more selective retrieval modes based on temperature gradients. Strictly speaking, the probe consists of two logical sections: the *query* and *tail*. The tail is the portion of the strand that is used with *in vitro* experiments to physically retrieve the marked DNA from the test tube (e.g., biotin-streptavidin-coated beads or fluorescent tags [16]). The query is the portion of the strand that is expected to hybridize with strands from the library to form a double-stranded entity. We will only be concerned with the latter below, as the former becomes important only at the implementation stage, or just be identical to the duplex formed during retrieval.

When a probe comes close enough to a library or probe strand in the tube so that any hybridization between the two strands is possible, an encounter (which triggers a check for hybridization) is said to have occurred. The number of encounters can vary greatly depending directly on the concentration of probes and library strands. It appears that higher concentration reduce retrieval time, but this is only true to a point

since results below show that too much concentration will interfere with the retrieval process. In other words, a large number of encounters may cause unnecessary hybridization attempts that will slow down the simulation. Further, too many neighbor strands may hinder the movement of the probe strands in search of their match. Probing is considered complete when probe copies have formed enough retrieval duplexes with library strands that should be retrieved (perhaps none) according to stringency of the retrieval (here the h-distance threshold.) In single probes with high stringency (perfect matches), probing can be halted when one successful hybridization occurs. Lesser stringency and multiple simultaneous probes require longer times to complete the probe. The question arises how long is long enough to complete the probes with high reliability.

2.3 Test Libraries and Experimental Conditions

The experiments used mostly a library consisting of the full set of 512 non-complementary 5-mer strands, although other libraries obtained through the software package developed based on the thermodynamic model of Deaton et Al. [5] were also tried with consistent results. This is a desirable situation to benchmark retrieval performance since the library is saturated (maximum size) and retrieval times would be worst-case. The probes were chosen to be random probes of 5-mers. The stringency was highest (h-distance 0), so exact matches were required. The experiment began by placing variable concentrations (number of copies) of the library and the probes into the tube of constant size. Once placed in the tube, the simulation begins. It stops when the first hybridization is detected. For the purposes of these experiments, there existed no error margin thus preventing close matches from hybridizing. Introduction of more flexible thresholds does not affect the results of the experiments.

In the first batch of experiments, we collected data to quantify the efficiency of the retrieval process (time, number of encounters, and attempted hybridizations) with single queries between related strands and its variance in hybridization attempts until successful hybridization. Three successive batches of experiments were designed to determine the optimal concentrations with which the retrieval was both successful and efficient, as well as to determine the effect on retrieval times of multiple probes in a single query. The experiments were performed between 5 and 100 times each and the results averaged. The complexity and variety of experiments has limited the quantity of runs possible for each experiment. Over a total of over 2000 experiments were run continuously over the course of many weeks.

3 Analysis of Results

Below are the results of the experiments, with some analysis of the data gathered.

3.1 Retrieval Efficiency

Figure 1 shows the results of the first experiment at various concentrations averaged over five runs. The most hybridization attempts occurred when the concentration of probes is between 50-60 copies and the concentration of library strands was between 20-30 copies. Figure 2 represents the variability (as measured by the standard deviation) of the experimental data. Although, there exists an abnormally high variance in some deviations in the population, most data points exist with deviations less than 5000. This high variance can be partially explained by the probabilistic chance of any two matching strands encountering each other by following a random walk. Interestingly enough, the range of 50-60 probe copies and 20-30 library copies exhibits minimum deviations.

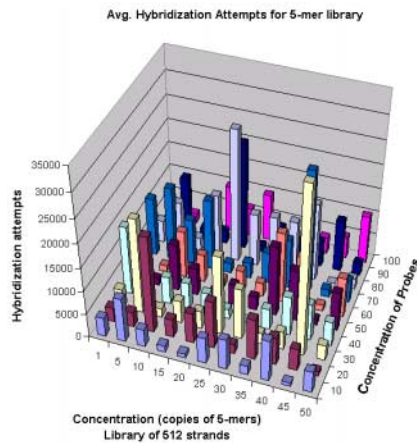


Fig. 1. Retrieval difficulty (hybridization attempts) based on concentration.

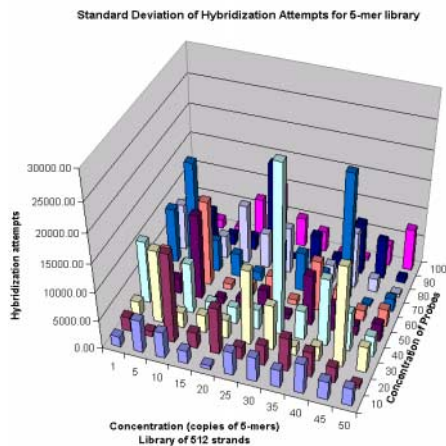


Fig. 2. Variability in retrieval difficulty (hybridization attempts) based on concentration.

3.2 Optimal Concentrations

Figure 3 shows the average retrieval times as measured in tube iterations. The number of iterations decreases as the number of probes and library strands increase, to a point. One might think at first that the highest available probe and library concentration is desirable. However, Fig. 1 indicates a diminishing return in that the number of hybridization attempts increases as the probe and library concentration increase. In order for the experiments *in silico* to be representative of the wet test tube experiments, a compromise must be made. Therefore, if the ranges of concentrations determined from Fig. 1 are used, the number of tube iterations remains under 200. Fig. 4 shows only minimum deviations once the optimal concentration has been achieved. The larger deviations at the lower concentrations can be accounted for by the highly randomized nature of the test tube simulation. These results on optimal concentration are consistent and further supported by comparison with the results in Fig. 1.

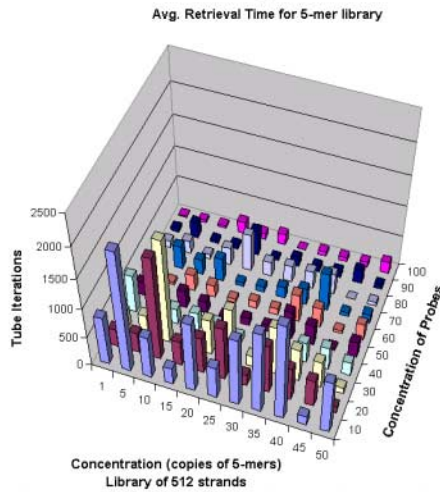


Fig. 3. Retrieval times (number of iterations) based on concentration.

As a comparison, in a second batch of experiments with a smaller (much sparser) library of 64 32-mers obtained by a genetic algorithm [9], the same dependent measures were tested. The results (averaged over 100 runs) are similar, but are displayed in a different form below. In Figure 5, the retrieval times ranged from nearly 0 through 5,000 iterations. For low concentrations, retrieval times were very large and exhibited great variability. As the concentration of probe strands exceeds a threshold of about 10, the retrieval times drop under 100 iterations, assuming a library strand concentration of about 10 strands.

Finally, Figure 6 shows that the retrieval time increases only logarithmically with the number of multiple queries and tends to level off in the range within which probes don't interfere with one another.

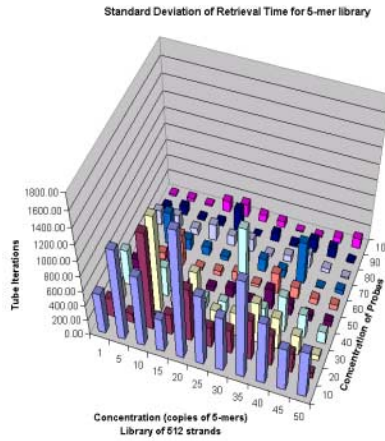


Fig. 4. Variability in retrieval times (number of iterations) based on concentration.

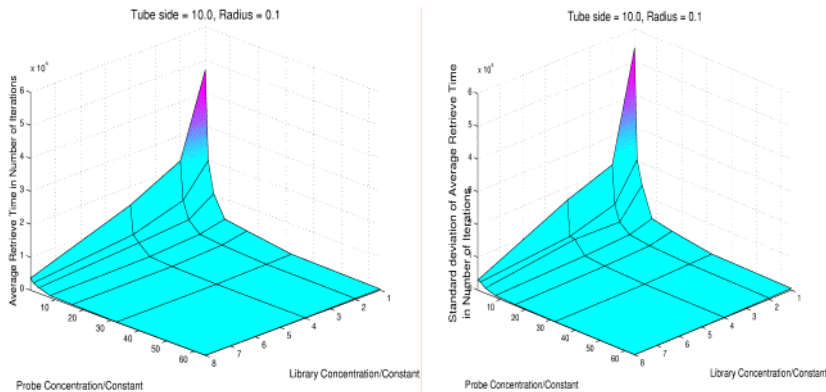


Fig. 5. Retrieval times and optimal concentration on sparser library.

In summary, these results permit a preliminary estimate of optimal and retrieval times for queries in DNA associative memories. For a library of size N , a good concentration of library for optimal retrieval time appears to be in the order of $O(\log N)$. Probe strands require the same order, although probably a smaller number will suffice. The variability in the retrieval time also decreases for optimal concentrations. Although not reported here in detail due to space constraints, similar phenomena were observed for multiple probes. We surmise that this hold true up to $O(\log N)$ simultaneous probes, past which probes begin to interfere with one another causing a substantial increase in retrieval time. Based on benchmarks obtained by comparing simulations in *Edna* with

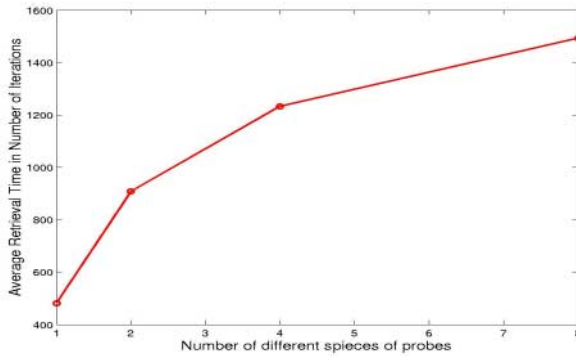


Fig. 6. Retrieval times (number of iterations) based on multiple simultaneous of queries.

wet tube experiments [7], we can estimate the actual retrieval time itself in all these events to be in the order of 1/10 of a second for libraries in the range of 1 to 100 millions strands in a wet tube.

It is worth noticing that similar results may be expected for memory updates. Adding a record is straightforward in DNA-based memories (assuming that the new record is noncrosshybridizing with the current memory), one can just drop it in the solution. Deleting a record requires making sure that all copies of the records are retrieved (full stringency for perfect recall) and expunged, which reduces deletion to the problem above. Additional experiments were performed that verified this conclusion. The problem of adding new crosshybridizing records is of a different nature and was not addressed in this project.

3.3 DNA-Based Memory Capacity

An issue of paramount importance is the capacity of the memories considered in this paper. Conventional memories and even memories developed with other technologies have impressive sizes despite apparent shortcomings such as address-based indexing and sequential search retrievals. DNA-based memories need to offer a definitive advantage to make them competitive. Candidates are massive size, associative retrieval, and straightforward implementation by recombinant biotechnology. We address below only the first aspect.

Baum [2] claimed that it seemed DNA-based memories could be made with a capacity larger than the brain, but warned that preventing undesirable cross-hybridization may reduce the potential capacity of 4^n strands for a library made of n -mers. Later work on error-prevention has confirmed that the reduction will be orders of magnitude smaller [6]. Based on combinatorial constraints, [14] combinatorially obtained some theoretical lower bounds and upper bounds of the number of equi-length DNA strands. However, from the practical point of view, the question still remains of determining the size of the largest memories based on oligonucleotides in effective use (20 to 150-mers).

A preliminary estimation of the runs has been made in several ways. First, a greedy search of small DNA spaces (up to 9-mers) in [10] by exhaustive searches averaged a number of 100 code words or less at a minimum h-distance apart of 4 or more, in a space of at least 4^{10} strands, regardless of the random order in which they the entire spaces were searched. Using the more realistic (but still approximate) thermodynamic model of Deaton et Al. [5], similar greedy searches turned up libraries of about 1,400 10-mers with nonnegative pairwise Gibbs energies (given by the model.) An *in vitro* selection protocol proposed by Deaton et Al. [4] has been tested experimentally and is expected to produce large libraries. The difficulty is that quantifying the size of the libraries obtained by the selection protocol is yet an unresolved problem given the expected size for 20-mers. In a separate experiment simulating this selection protocol, *Edna* has produced libraries of about 100 to 150 n -mers ($n=10, 11, 12$) starting with a full size DNA space of all n -mers (crosshybridizing) as the seed populations. Further several simulations of the selection protocol with random seeds of 1024 20-mers as initial population have consistently produced libraries of no more than 150 20-mers. A linear extrapolation to the size of the entire population is too risky because the greedy searches show that sphere packing allows high density in the beginning, but tends to add more strands very sparsely toward the end of the process. The true growth rate of the library size as a function of strand size n remains a truly intriguing question.

4 Summary and Conclusions

The reliability and efficiency of DNA-based associative memories has been explored quantitatively through simulation of reactions *in silico* on a virtual test tube. They show that there the region of optimal concentrations for library and probe strands to minimize retrieval time and avoid excessive concentrations (which tend to lengthen retrieval times) is about $O(\log N)$, where N is the size of the library. Further the retrieval time is highly dependent on reactions conditions and the probe, but tends to stabilize at optimal concentrations. Furthermore, these results remain essentially unchanged for simultaneous multiple queries if they remain small compared to the library size (within $O(\log N)$.) Previous benchmarks of the virtual tube provide a good level of confidence that these results extrapolate well to wet tubes with real DNA. The retrieval times in that case can be estimated in the order of 1/10 of a second. The important question of how the memory capacity grows as a function of strand size is certainly sub-exponential, but remains a truly intriguing open question.

An interesting possibility is suggested by the results presented here. The experiments were run in simulation. It is thus conceivable that conventional memories could be designed in hardware using special-purpose chips of the software simulations. The chips would run according to the parallelism inherent in VLSI circuits. One iteration could be run in nanoseconds with current technology. Therefore, once can obtain the advantages of DNA-based associative recall at varying threshold of stringency *in silico*, while retaining the speed, implementation, and manufacturing facilities of solid-state memories. A further exploration of this idea will be fleshed out elsewhere.

References

1. L.M. Adleman: Molecular Computation of Solutions to Combinatorial Problems. *Science* **266** (1994) 1021–1024
2. E. Baum, Building An Associative Memory Vastly Larger Than The Brain. *Science* **268** (1995) 583–585.
3. A. Condon, G. Rozenberg (eds.): DNA Computing (Revised Papers). In: Proc. of the 6th International Workshop on DNA-based Computers, 2000. Springer-Verlag Lecture Notes in Computer Science **2054** (2001)
4. R. Deaton, R., J. Chen, H. Bi, M. Garzon, H. Rubin, D.H. Wood. A PCR-Based Protocol for In-Vitro Selection of Non-Crosshybridizing Oligonucleotides (2002). In [11], 105–114
5. R.J. Deaton, J. Chen, H. Bi, J.A. Rose: A Software Tool for Generating Non-crosshybridizing Libraries of DNA Oligonucleotides. In [11], pp. 211–220.
6. R. Deaton, M. Garzon, R. E. Murphy, J. A. Rose, D. R. Franceschetti, S.E. Stevens, Jr. The Reliability and Efficiency of a DNA Computation. *Phys. Rev. Lett.* **80** (1998) 417–420
7. M. Garzon, D. Blain, K. Bobba, A. Neel, M. West: Self-Assembly of DNA-like structures *in silico*. Journal of Genetic Programming and Evolvable Machines **4**:2 (2003), in press.
8. M. Garzon: Biomolecular Computation *in silico*. *Bull. of the European Assoc. For Theoretical Computer Science* EATCS (2003), in press.
9. M. Garzon, C. Oehmen: Biomolecular Computation on Virtual Test Tubes. In: N. Jonoska and N. Seeman (eds.): Proc. of the 7th International Workshop on DNA-based Computers, 2001. Springer-Verlag Lecture Notes in Computer Science **2340** (2002) 117–128
10. M. Garzon, R. Deaton, P. Neathery, R.C. Murphy, D.R. Franceschetti, E. Stevens Jr.: On the Encoding Problem for DNA Computing. In: Proc. of the Third DIMACS Workshop on DNA-based Computing, U of Pennsylvania. (1997) 230–237
11. M. Hagiya, A. Ohuchi (eds.): Proceedings of the 8th Int. Meeting on DNA Based Computers, Hokkaido University, 2002, Springer-Verlag Lecture Notes in Computer Science **2568** (2003)
12. J. Lee, S. Shin, S.J. Augh, T.H. Park, B. Zhang: Temperature Gradient-Based DNA Computing for Graph Problems with Weighted Edges. In [11], pp. 41–50.
13. R. Lipton: DNA Solutions of Hard Computational Problems. *Science* **268** (1995) 542–544
14. A. Marathe, A. Condon, R. Corn: On Combinatorial Word Design. In: E. Winfree and D. Gifford (eds.): DNA Based Computers V, DIMACS Series in Discrete Mathematics and Theoretical Computer Science. **54** (1999) 75–89
15. J.H. Reif, T. LaBean. Computationally Inspired Biotechnologies: Improved DNA Synthesis and Associative Search Using Error-Correcting Codes and Vector Quantization In [3], pp. 145–172
16. K.A. Schmidt, C.V. Henkel, G. Rozenberg: DNA computing with single molecule detection. In [3], 336.
17. J.G. Wetmur: Physical Chemistry of Nucleic Acid Hybridization. In: H. Rubin and D.H. Wood (eds.): Proc. DNA-Based Computers III, U. of Pennsylvania, 1997. DIMACS series in Discrete Mathematics and Theoretical Computer Science **48** (1999) 1–23
18. E. Winfree, F. Liu, L.A. Wenzler, N.C. Seeman: Design and self-assembly of two-dimensional DNA crystals. *Nature* **394** (1998) 539–544

Evolving Hogg's Quantum Algorithm Using Linear-Tree GP

André Leier and Wolfgang Banzhaf

University of Dortmund, Dept. of Computer Science, Chair of Systems Analysis,
44221 Dortmund, Germany
{andre.leier, wolfgang.banzhaf}@cs.uni-dortmund.de

Abstract. Intermediate measurements in quantum circuits compare to conditional branchings in programming languages. Due to this, quantum circuits have a natural linear-tree structure. In this paper a Genetic Programming system based on linear-tree genome structures developed for the purpose of automatic quantum circuit design is introduced. It was applied to instances of the 1-SAT problem, resulting in evidently and “visibly” scalable quantum algorithms, which correspond to Hogg’s quantum algorithm.

1 Introduction

In theory certain computational problems can be solved on a quantum computer with a lower complexity than possible on classical computers. Therefore, in view of its potential, design of new quantum algorithms is desirable, although no working quantum computer beyond experimental realizations has been built so far. Unfortunately, the development of quantum algorithms is very difficult, since they are highly non-intuitive and their simulation on conventional computers is very expensive.

The use of genetic programming to evolve quantum circuits is not a novel approach. It was elaborated first in 1997 by Williams and Gray [21]. Since then, various other papers [5,1,15,18,17,2,16,14,20] dealt with quantum computing as an application of genetic programming or genetic algorithms, respectively. The primary goal of most GP experiments, described in this context, was to demonstrate the feasibility of automatic quantum circuit design. Different GP schemes and representations of quantum algorithms were considered and tested on various problems.

The GP system described in this paper uses linear-tree structures and was build to achieve more “degrees of freedom” in the construction and evolution of quantum circuits compared to stricter linear GP schemes (like in [14,18]). A further goal was to evolve quantum algorithms for the k -SAT problem (only for $k = 1$ up to now). In [9,10] Hogg has already introduced quantum search algorithms for 1-SAT and highly constrained k -SAT. An experimental implementation of Hogg’s 1-SAT algorithm for logical formulas in three variables is demonstrated in [13].

The following section briefly outlines some basics of quantum computing essential to understand the mathematical principles on which the simulation of quantum algorithms depends. Section 3 of this paper discusses previous work on automatic quantum circuit design. Section 4 describes the linear-tree GP scheme used here. The results of evolving quantum algorithms for the 1-SAT problem are presented in Sect. 5. The last section summarizes our results and draws conclusions.

2 Quantum Computing Basics

Quantum computing is the result of a link between quantum mechanics and information theory. It is computation based on quantum principles, that is quantum computers use coherent atomic-scale dynamics to store and to process information [19]. The basic unit of information is the qubit which, unlike a classical bit can exist in a superposition of the two classical states 0 and 1, i.e. with a certain probability p , resp. $1 - p$, the qubit is in state 0, resp. 1. In the same way an n -qubit quantum register can be in a superposition of its 2^n classical states. The state of the quantum register is described by a 2^n -dimensional complex vector $(\alpha_0, \alpha_1, \dots, \alpha_{2^n-1})^t$, where α_k is the probability amplitude corresponding to the classical state k . The probability for the quantum register being in state k is $|\alpha_k|^2$ and from the normalization condition of probability measures it follows $\sum_{k=0}^{2^n-1} |\alpha_k|^2 = 1$. It is common usage to write the classical states (the so-called computational basis states) in the 'ket' notation of quantum computing, as $|k\rangle = |a_{n-1}a_{n-2} \dots a_0\rangle$, where $a_{n-1}a_{n-2} \dots a_0$ is the binary representation of k . Thus, the general state of an n -qubit quantum computer can be written as $|\psi\rangle = \sum_{k=0}^{2^n-1} \alpha_k |k\rangle$.

The quantum circuit model of computation describes quantum algorithms as a sequence of unitary – and therefore reversible – transformations (plus some non-unitary measurement operators), also called quantum gates, which are applied successively to an initialized quantum state. Usually this state to an n -qubit quantum circuit is $|0\rangle^{\otimes n}$. A unitary transformation operating on n qubits is a $2^n \times 2^n$ matrix U , with $U^\dagger U = I$. Each quantum gate is entirely determined by it's gate type, the qubits, it is acting on, and a certain number of real-valued (angle) parameters. Figure 1 shows some basic gate types working on one or two qubits. Similar to the universality property of classical gates, small sets of quantum gates are sufficient to compute any unitary transformation to arbitrary accuracy. For example, single qubit and *CNOT* gates are universal for quantum computation, just as *H*, *CNOT*, *Phase* $[\pi/4]$ and *Phase* $[\pi/2]$ are. In order to be applicable to an n -qubit quantum computer (with a 2^n -dimensional state vector) quantum gates operating on less than n qubits have to be adapted to higher dimensions. For example, let U be an arbitrary single-qubit gate applied to qubit q of an n -qubit register. Then the entire n -qubit transformation is composed of the tensor product

$$\underbrace{I \otimes \dots \otimes I}_{n-(q+1)} \otimes U \otimes \underbrace{I \dots \otimes I}_q$$

$$H = 1/\sqrt{2} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \quad \text{Phase}[\phi] = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\phi} \end{pmatrix} \quad CNOT = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

$$Rx[\phi] = \begin{pmatrix} \cos \phi & i \sin \phi \\ i \sin \phi & \cos \phi \end{pmatrix} \quad Ry[\phi] = \begin{pmatrix} \cos \phi & \sin \phi \\ -\sin \phi & \cos \phi \end{pmatrix} \quad Rz[\phi] = \begin{pmatrix} e^{-i\phi} & 0 \\ 0 & e^{i\phi} \end{pmatrix}$$

Fig. 1. Some basic unitary 1- and 2-qubit transformations: Hadamard-gate H , a *Phase*-gate with angle parameter ϕ , a *CNOT*-gate, some rotation gates $Rx[\phi]$, $Ry[\phi]$, $Rz[\phi]$ with rotation angle ϕ .

Calculating the new quantum state requires 2^{n-1} matrix-vector-multiplications of the 2×2 matrix U . It is easy to see, that the costs of simulating quantum circuits on conventional computers grow exponentially with the number of qubits.

Input gates sometimes known as oracles enable the encoding of problem instances. They may change from instance to instance of a given problem, while the “surrounding” quantum algorithm remains unchanged. Consequently, a proper quantum algorithm solving the problem has to achieve the correct outputs for all oracles representing problem instances. In quantum algorithms like Grover’s [6] or Deutsch’s [3,4], oracle gates are permutation matrices computing Boolean functions (Fig. 2, left matrix). Hogg’s quantum algorithm for k -SAT [9,10] uses a special diagonal matrix, encoding the number of conflicts in assignment s , i. e. the number of false clauses for assignment s in the given logical formula at position (s, s) (Fig. 2, right matrix).

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \quad \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & i & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & i & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & i & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -i \end{pmatrix}$$

Fig. 2. Examples for oracle matrices. Left matrix: implementation of the *AND* function of two inputs. The right-most qubit is flipped, if the two other qubits are ‘1’. This gate is also called a *CCNOT*. Right matrix: a diagonal matrix with coefficients $(i^{c(000)}, \dots, i^{c(111)})$, where $c(s)$ is the number of conflicts of assignment s in the formula $\bar{v}_1 \wedge \bar{v}_2 \wedge \bar{v}_3$. For example, the assignment $(v_1 = \text{true}, v_2 = \text{false}, v_3 = \text{true})$ makes two clauses false, i. e. $c(101) = 2$ and $i^2 = -1$.

Quantum information processing is useless without readout (measurement). When the state of a quantum computer is measured in the computational basis, result ' k ' occurs with probability $|\alpha_k|^2$. By measurement the superposition collapses to $|k\rangle$. A partial measurement of a single qubit is a projection into the subspace, which corresponds to the measured qubit. The probability p of measuring a single qubit q with result '0' ('1') is the sum of the probabilities for all basis states with qubit $q = 0$ ($q = 1$). The post-measurement state is just the superposition of these basis states, re-normalized by the factor $1/\sqrt{p}$. For example, measuring the first (right-most) qubit of $|\psi\rangle = \alpha_0|00\rangle + \alpha_1|01\rangle + \alpha_2|10\rangle + \alpha_3|11\rangle$ gives '1' with probability $|\alpha_1|^2 + |\alpha_3|^2$, leaving the post-measurement state $|\psi'\rangle = 1/\sqrt{|\alpha_1|^2 + |\alpha_3|^2}(\alpha_1|01\rangle + \alpha_3|11\rangle)$.

According to the *quantum principle of deferred measurement*, "measurements can always be moved from an intermediate stage of a quantum circuit to the end of the circuit" [12]. Of course, such a shift has to be compensated by some other changes in the quantum circuit.

Note, that quantum measurements are irreversible operators, though it is usual to call these operators measurement *gates*. To get a deeper insight into quantum computing and quantum algorithms the following references might be of interest to the reader: [12],[7],[8].

3 Previous Work in Automatic Quantum Circuit Design

Williams and Gray focus in [21] on demonstrating a GP-based search heuristic more efficient than the exhaustive enumeration strategy which finds a correct decomposition of a given unitary matrix U into a sequence of simple quantum gate operations. In contrast, however, to subsequent GP schemes for the evolution of quantum circuits, a unitary operator solving the given problem had to be known in advance.

Extensive investigations concerning the evolution of quantum algorithms were done by Spector et al. [15,18,17,1,2]. In [18] they presented three different GP schemes for quantum circuit evolution: the standard tree-based GP (TGP) and both stack-based and stackless linear genome GP (SBLGP/SLLGP). These were applied to evolve algorithms for Deutsch's two-bit early promise problem, using TGP, the scaling majority-on problem, using TGP as well, the quantum four-item database search problem, using SBLGP, and the two-bit-AND-OR problem, using SLLGP. Better-than-classical algorithms could be evolved for all but the scaling majority-on problem.

Without doing a thorough comparison Spector et al. pointed out some pros and cons of the three GP schemes: The tree structure of individuals in TGP simplifies the evolution of scalable quantum circuits, as it seems to be predestined for "adaptive determination of program size and shape" [18]. A disadvantage of the tree representation are its higher costs in time, space and complexity. Furthermore, possible return-value/side-effect interactions may make evolution more complicated for TGP. The linear representation in SBLGP/SLLGP seems to be better suited for evolution, because the quantum algorithms are itself se-

quential (in accordance with the principle of deferred measurement). Moreover, the genetic operators in linear GP are simpler to implement and memory requirements are clearly reduced compared to TGP. The return-value/side-effect interaction is eliminated in SBGL, since the algorithm-building functions do not return any values. Overall, Spector et al. stated that, applied to their problems, results appeared to emerge more quickly with SBLGP than with TGP. If scalability of the quantum algorithms would be not so important, the SLLGP approach should be preferred.

In [17] and [2] a modified SLLGP system was applied to the 2-bit-AND-OR problem, evolving an improved quantum algorithm. The new system is steady-state rather than generational as its predecessor system, supports true variable-length genomes and enables distributed evolution on a workstation cluster. Expensive genetic operators allow for “local hill-climbing search [...] integrated into the genetic search process”. For fitness evaluation the GP system uses a standardized lexicographic fitness function consisting of four fitness components: the number of fitness cases on which the quantum program “failed” (MISSES), the number of expected oracle-gates in the quantum circuit (EXPECTED-QUERIES), the maximum probability over all fitness cases of getting the wrong result (MAX-ERROR) and the number of gates (NUM-GATES).

Another interesting GP scheme is presented in [14] and its function is demonstrated by generating quantum circuits for the production of two to five maximally entangled qubits. In this scheme gates are represented by a gate type and by bit-strings coding the qubit operands and gate parameters. Qubit operands and parameters have to be interpreted corresponding to the gate type. Assigning a further binary key to each gate type the gate representation is completely based on bit strings, where appropriate genetic operators can be applied to.

4 The Linear-Tree GP Scheme

The steady-state GP system described here is a linear-tree GP scheme, introduced first in [11]. The structure of the individuals consists of linear program segments, which are sequences of *unitary* quantum gates, and branchings, caused by single qubit measurement gates. Depending on the measurement result (‘0’ or ‘1’), the corresponding (linear) program branch, the ‘0’- or ‘1’-branch, is executed. Since measurement results occur with certain probabilities, usually both branches have to be evaluated. Therefore, the quantum gates in the ‘0’- and ‘1’-branch have to be applied to their respective post-measurement states. From the branching probabilities the probabilities for each final quantum state can be calculated.

In this way linear-tree GP naturally supports the use of measurements as an intermediate step in quantum circuits. Measurement gates can be employed to conditionally control subsequent quantum gates, like an “if-then-else”-construct in a programming language. Although the principle of deferred measurement suggests the use of purely sequential individual structures, the linear-tree structure may simplify legibility and interpretation of quantum algorithms.

The maximum number of possible branches is set by a global system parameter; without using any measurement gates the GP system becomes very similar to the modified SLLGP version in [17]. From there, we adopted the idea of using fitness components with certain weights: MISSES, MAX-ERROR and TOTAL-ERROR (the summed error over all fitness cases) are used in this way. A penalty function based on NUM-GATES and a global system parameter is used to increase slightly the fitness value for any existing gate in the quantum circuit. In order to restrict the evolution, in particular at the beginning of a GP run, fitness evaluation of an individual is aborted if the number of MISSES exceeds a certain value, set by another global system parameter. The bitlength of gate parameters (interpreted as a fraction of 2π) was fixed to 12 bits which restricts angle resolution. This corresponds to current precisions for NMR experiments. The genetic operators used here are RANDOM-INSERTION, RANDOM-DELETION and RANDOM-ALTERATION, each referred to a single quantum gate, plus LINEAR-XOVER and TREE-XOVER. A GP run terminates when the number of tournaments exceeds a given value (in our experiments, 500000 tournaments) or the fitness of a new best individual under-runs a given threshold.

It should be emphasized that the GP system is not designed to *directly* evolve scalable quantum circuits. Rather, by scalability we mean that the algorithm does not only work on n but also on $n+1$ qubits. At least for the 1-SAT problem, scalability of the solutions became “visible”, as is shown below.

5 Evolving Quantum Circuits for 1-SAT

The 1-SAT problem for n variables, solved by classical heuristics in $O(n)$ steps, can be solved even faster on a quantum computer. Hogg's quantum algorithm, presented in [9,10], finds a solution in a single search step, using a clever input matrix (see Sect. 2 and Fig. 2). Let R denote this input matrix, with $R_{ss} = i^{c(s)}$ where $c(s)$ is the number of conflicts in the assignment s of a given logical 1-SAT formula in n variables. Thus, the problem description is entirely encoded in this input matrix. Furthermore, let be U the matrix defined by $U_{rs} = 2^{-n/2}(-1)^{d(r,s)}$, where $d(r,s)$ is the Hamming distance between r and s . Then the entire algorithm is the sequential application of Hadamard gates applied to n qubits ($H^{\otimes n}$) initially in state $|0\rangle$, R and U . It can be proven, that the final quantum state is the (equally weighted) superposition of all assignments s with $c(s) = 0$ conflicts.¹ A final measurement will lead, with equal probability, to one of the 2^{n-m} solutions, where m denotes the number of clauses in the 1-SAT formula.

We applied our GP system on problem instances of $n = 2.4$ variables. The number of fitness cases (the number of formulas) is $\sum_{k=1}^n \binom{n}{k} 2^k$ in total. Each fitness case consists of an input state (always $|0\rangle^{\otimes n}$), an input matrix for the formula and the desired output. For example,

¹ For all 1-SAT (and also maximally constrained 2-SAT) problems Hogg's algorithm finds a solution with probability one. Thus, an incorrect result definitely indicates the problem is not soluble [9].

$$(|00\rangle, \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & i & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & i \end{pmatrix}, |-0\rangle)$$

is the fitness case for the 1-SAT formula $\bar{v}_2$ in two variables v_1, v_2 . Here, the ‘ $-$ ’ in $|-0\rangle$ denotes a “don’t care”, since only the rightmost qubit is essential to the solutions $\{v_1 = \text{true/false}, v_2 = \text{false}\}$. That means, an equally weighted superposition of all solutions is not required.

Table 1 gives some parameter settings for GP runs applied to the 1-SAT problem.

Table 1. Parameter settings for the 1-SAT problem with $n = 4$. ^{*)}After evolving solutions for $n = 2$ and $n = 3$, intermediate measurements seemed to be irrelevant for searching 1-SAT quantum algorithms, since at least the evolved solutions did not use them. Without intermediate measurements (gate type M), which constitute the tree structure of quantum circuits, tree crossover is not applicable. In GP runs for $n = 2, 3$ the maximum number of measurements was limited by the number of qubits.

Population Size	5000
Tournament Size	16
Basic Gate Types	$H, Rx, Ry, Rz, C^k NOT, M$
Max. Number of Gates	15
Max. Number of Measurements	0 ^{*)}
Number of Input Gates	1
Mutation Rate	1
Crossover (XO) Rate	0.1
Linear XO Probability	1 ^{*)}
Deletion Probability	0.3
Insertion Probability	0.3
Alteration Probability	0.4

For the two-, three- and four-variable 1-SAT problem 100 GP runs were done recording the best evolved quantum algorithm of each run. Finally the over-all best quantum algorithm was determined. For each problem instance our GP system evolved solutions (Figs. 3 and 4) that are essentially identical to Hogg’s algorithm. This can be seen at a glance, when noting that $U = Rx[3/4\pi]^{\otimes n}$.²

The differences in fitness values of the best algorithms of each GP run, were negligible, though they differed in length and structure, i. e. in the arrangement of gate-types. Most quantum algorithms did not make use of intermediate measurements. Details of the performance and convergence of averaged fitness values over all GP runs can be seen in the three graphs of Fig. 5.

² Note, that U is equal to $Rx[3/4\pi]^{\otimes n}$ up to a global phase factor, which of course has no influence on the final measurement results.

Misses: 0
Max. Error: 8.7062e-05
Total Error: 0.0015671
Oracle Number: 1
Gate Number: 10
Fitness Value: 0.00025009

Individual:
H 0
H 1
H 2
INP
RX 6.1083 0
RX 2.6001 0
RX 3.0818 0
RX 2.3577 1
RX 2.3562 2
RZ 0.4019 1

Fig. 3. Extract from the GP system output: After 100 runs this individual was the best evolved solution to 1-SAT with three variables. Here, INP denotes the specific input matrix R .

		H 0
	H 0	H 1
H 0	H 1	H 2
H 1	H 2	H 3
INP	INP	INP
Rx[3/4 Pi] 0	Rx[3/4 Pi] 0	Rx[3/4 Pi] 0
Rx[3/4 Pi] 1	Rx[3/4 Pi] 1	Rx[3/4 Pi] 1
	Rx[3/4 Pi] 2	Rx[3/4 Pi] 2
		Rx[3/4 Pi] 3

Fig. 4. The three best, slightly hand-tuned quantum algorithms to 1-SAT with $n = 2, 3, 4$ (from left to right) after 100 evolutionary runs each. Postprocessing was used to eliminate introns, i. e. gates which have no influence on the quantum algorithm or the final measurement results respectively, and to combine two or more rotation gates of the same sort into one single gate. Here, the angle parameters are stated more precisely in fractions of π . INP denotes the input gate R as specified in the text. Without knowledge of Hogg’s quantum algorithm, there would be strong evidence for the scalability of this evolved algorithm.

Further GP runs with different parameter settings hinted at strong parameter dependencies. For example, an adequate limitation of the maximum number of gates leads rapidly to good quantum algorithms. In contrast, stronger limitations (somewhat above the length of the best evolved quantum algorithm) made convergence of the evolutionary process more difficult. We experimented also

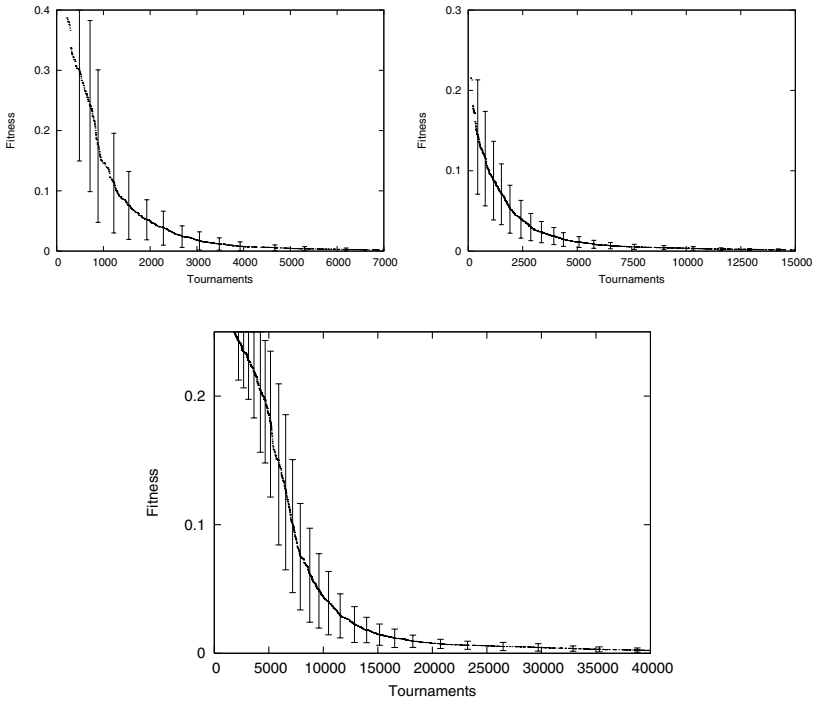


Fig. 5. Three graphs illustrating the course of 100 evolutionary runs for quantum algorithms for the two-, three- and four-variable 1-SAT problem. Errorbars show the standard deviation for the averaged fitness values of the 100 best evolved quantum algorithms after a certain number of tournaments. The dotted line marks averaged fitness values. Convergence of the evolution is obvious.

with different gate sets. Unfortunately, for larger gate sets “visible” scalability was not detectable. GP runs on input gates implementing a logical 1-SAT formula as a permutation matrix, which is a usual problem representation in other quantum algorithms, did not lead to acceptable results, i.e. quantum circuits with zero error probability. This may be explained with the additional problem-specific information (the number of conflicts for each assignment) encoded in the matrix R . The construction of Hogg’s input representation from some other representation matrices does not need to be hard for GP at all, but it may require some more ancillary qubits to work. Note, however, that due to the small number of runs with these parameter settings the results do not have statistical evidence.

6 Conclusions

The problems of evolving novel quantum algorithms are evident. Quantum algorithms can be simulated in acceptable time only for very few qubits without excessive computer power. Moreover, the number of evaluations per individual to calculate its fitness are given by the number of fitness-cases usually increases exponentially or even super-exponentially. As a direct consequence, automatic quantum circuit design seems to be feasible only for problems with sufficiently small instances (in the number of required qubits). Thus the examination of scalability becomes a very important topic and has to be considered with special emphasis in the future.

Furthermore, as Hogg's k -SAT quantum algorithm shows, a cleverly designed input matrix is crucial for the outcome of a GP-based evolution. For the 1-SAT problem, the additional tree structure in the linear-tree GP scheme did not take noticeable effect, probably because of the simplicity of the problem solutions.

Perhaps, genetic programming and quantum computing will have a brighter common future, as soon as quantum programs do not have to be simulated on classical computers, but can be tested on true quantum computers.

Acknowledgement. This work is supported by a grant from the Deutsche Forschungsgemeinschaft (DFG). We thank C. Richter and R. Stadelhofer for numerous discussions and helpful comments.

References

- [1] H. Barnum, H. Bernstein, and L. Spector, *Better-than-classical circuits for OR and AND/OR found using genetic programming*, 1999, LANL e-preprint quant-ph/9907056.
- [2] H. Barnum, H. Bernstein, and L. Spector, *Quantum circuits for OR and AND of ORs*, J. Phys. A: Math. Gen., 33 (2000), pp. 8047–8057.
- [3] D. Deutsch, *Quantum theory, the Church-Turing principle and the universal quantum computer*, Proc. R. Soc. London A, 400 (1985), pp. 97–117.
- [4] D. Deutsch and R. Jozsa, *Rapid solution of problems by quantum computation*, Proc. R. Soc. London A, 439 (1992), pp. 553–558.
- [5] Y. Ge, L. Watson, and E. Collins, *Genetic algorithms for optimization on a quantum computer*, in Proceedings of the 1st International Conference on Unconventional Models of Computation (UMC), C. Calude, J. Casti, and M. Dinneen, eds., DMTCS, Auckland, New Zealand, Jan. 1998, Springer, Singapur, pp. 218–227.
- [6] L. Grover, *A fast quantum mechanical algorithm for database search*, in Proceedings of the 28th Annual ACM Symposium on Theory of Computing (STOC), ACM, ed., Philadelphia, Penn., USA, May 1996, ACM Press, New York, pp. 212–219, LANL e-preprint quant-ph/9605043.
- [7] J. Gruska, *Quantum Computing*, McGraw-Hill, London, 1999.
- [8] M. Hirvensalo, *Quantum Computing*, Natural Computing Series, Springer-Verlag, 2001.
- [9] T. Hogg, *Highly structured searches with quantum computers*, Phys. Rev. Lett., 80 (1998), pp. 2473–2476.

- [10] T. Hogg, *Solving highly constrained search problems with quantum computers*, J. Artificial Intelligence Res., 10 (1999), pp. 39–66.
- [11] W. Kantschik and W. Banzhaf, *Linear-tree GP and its comparison with other GP structures*, in Proceedings of the 4th European Conference on Genetic Programming (EUROGP), J. Miller, M. Tomassini, P. Lanzi, C. Ryan, A. Tettamanzi, and W. Langdon, eds., vol. 2038 of LNCS, Lake Como, Italy, Apr. 2001, Springer, Berlin, pp. 302–312.
- [12] M. Nielsen and I. Chuang, *Quantum Computation and Quantum Information*, Cambridge University Press, 2000.
- [13] X. Peng, X. Zhu, X. Fang, M. Feng, M. Liu, and K. Gao, *Experimental implementation of Hogg's algorithm on a three-quantum-bit NMR quantum computer*, Phys. Rev. A, 65 (2002).
- [14] B. Rubinstein, *Evolving quantum circuits using genetic programming*, in Proceedings of the 2001 Congress on Evolutionary Computation, IEEE, ed., Seoul, Korea, May 2001, IEEE Computer Society Press, Silver Spring, MD, USA, pp. 114–151. The first version of this paper already appeared in 1999.
- [15] L. Spector, *Quantum computation - a tutorial*, in GECCO-99: Proceedings of the Genetic and Evolutionary Computation Conference, W. Banzhaf, J. Daida, A. Eiben, M. H. Garzon, V. Honavar, M. Jakiela, and R. Smith, eds., Orlando, Florida, USA, Jul. 1999, Morgan Kaufmann Publishers, San Francisco, pp. 170–197.
- [16] L. Spector, *The evolution of arbitrary computational processes*, IEEE Intelligent Systems, (2000), pp. 80–83.
- [17] L. Spector, H. Barnum, H. Bernstein, and N. Swamy, *Finding a better-than-classical quantum AND/OR algorithm using genetic programming*, in Proceedings of the 1999 Congress on Evolutionary Computation, P. Angeline, Z. Michalewicz, M. Schoenauer, X. Yao, and A. Zalzal, eds., Washington DC, USA, Jul. 1999, IEEE Computer Society Press, Silver Spring, MD, USA, pp. 2239–2246.
- [18] L. Spector, H. Barnum, H. Bernstein, and N. Swamy, *Quantum Computing Applications of Genetic Programming*, in Advances in Genetic Programming, L. Spector, U.-M. O'Reilly, W. Langdon, and P. Angeline, eds., vol. 3, MIT Press, Cambridge, MA, USA, 1999, pp. 135–160.
- [19] A. Steane, *Quantum computation*, Reports on Progress in Physics, 61 (1998), pp. 117–173, LANL e-preprint quant-ph/9708022.
- [20] A. Surkan and A. Khuskivadze, *Evolution of quantum algorithms for computer of reversible operators*, in Proceedings of the 2002 NASA/DoD Conference on Evolvable Hardware (EH), IEEE, ed., Alexandria, Virginia, USA, Jul. 2002, IEEE Computer Society Press, Silver Spring, MD, USA, pp. 186–187.
- [21] C. Williams and A. Gray, *Automated Design of Quantum Circuits*, in Explorations in Quantum Computing, C. Williams and S. Clearwater, eds., Springer, New York, 1997, pp. 113–125.

Hybrid Networks of Evolutionary Processors

Carlos Martín-Vide¹, Victor Mitrana^{2*}, Mario J. Pérez-Jiménez^{3**}, and
Fernando Sancho-Caparrini³

¹ Rovira i Virgili University,
Research Group in Mathematical Linguistics,
Pça. Imperial Tàrraco 1, 43005 Tarragona, Spain,
`cmv@correu.urv.es`

² University of Bucharest, Faculty of Mathematics and Computer Science,
Str. Academiei 14, 70109 Bucharest, Romania,
`mitrana@funinf.math.unibuc.ro`

³ University of Seville,
Department of Computer Science and Artificial Intelligence,
`{Mario.Perez,Fernando.Sancho}@cs.us.es`

Abstract. A hybrid network of evolutionary processors consists of several processors which are placed in nodes of a virtual graph and can perform one simple operation only on the words existing in that node in accordance with some strategies. Then the words which can pass the output filter of each node navigate simultaneously through the network and enter those nodes whose input filter was passed. We prove that these networks with filters defined by simple random-context conditions, used as language generating devices, are able to generate all linear languages in a very efficient way, as well as non-context-free languages. Then, when using them as computing devices, we present two linear solutions of the Common Algorithmic Problem.

1 Introduction

This work is a continuation of the investigation started in [1] and [2] where one has considered a mechanism inspired from cell biology, namely networks of evolutionary processors, that is networks whose nodes are very simple processors able to perform just one type of point mutation (insertion, deletion or substitution of a symbol). These nodes are endowed with filters which are defined by some membership or random context condition.

Another source of inspiration is a basic architecture for parallel and distributed symbolic processing, related to the Connection Machine [13] as well as

* Corresponding author. This work, done when this author was visiting the Department of Computer Science and Artificial Intelligence of the University of Seville, was supported by the Generalitat de Catalunya, Direcció General de Recerca (PIV2001-50)

** Work supported by the project TIC2002-04220-C03-01 of the Ministerio de Ciencia y Tecnología of Spain, cofinanced by FEDER funds

the Logic Flow paradigm [6]. This consists of several processors, each of them being placed in a node of a virtual complete graph, which are able to handle data associated with the respective node. Each node processor acts on the local data in accordance with some predefined rules, and, then local data becomes a mobile agent which can navigate in the network following a given protocol. Only such data can be communicated which can pass a filtering process. This filtering process may require to satisfy some conditions imposed by the sending processor, by the receiving processor or by both of them. All the nodes send simultaneously their data and the receiving nodes handle also simultaneously all the arriving messages, according to some strategies, see, e.g., [7,13].

Starting from the premise that data can be given in the form of strings, [4] introduces a concept called network of parallel language processors in the aim of investigating this concept in terms of formal grammars and languages. Networks of language processors are closely related to grammar systems, more specifically to parallel communicating grammar systems [3]. The main idea is that one can place a language generating device (grammar, Lindenmayer system, etc.) in any node of an underlying graph which rewrite the strings existing in the node, then the strings are communicated to the other nodes. Strings can be successfully communicated if they pass some output and input filter.

Mechanisms introduced in [1] and [2] simplify as much as possible the networks of parallel language processors defined in [4]. Thus, in each node is placed a very simple processor, called evolutionary processor, which is able to perform a simple rewriting operation only, namely either insertion of a symbol or substitution of a symbol by another, or deletion of a symbol. Furthermore, filters used in [4] are simplified in some versions defined in [1,2].

In spite of these simplifications, these mechanisms are still powerful. In [2] networks with at most six nodes having filters defined by the membership to a regular language condition are able to generate all recursively enumerable languages no matter the underlying structure. This result does not surprise since similar characterizations have been reported in the literature, see, e.g., [5,11,10,12,14]. Then one considers networks with nodes having filters defined by random context conditions which seem to be closer to the biological possibilities of implementation. Even in this case, rather complex languages like non-context-free ones, can be generated.

However, these very simple mechanisms are able to solve hard problems in polynomial time. In [1] it is presented a linear solution for an NP-complete problem, namely the Bounded Post Correspondence Problem, based on networks of evolutionary processors able to substitute a letter at any position in the string but insert or delete a letter in the right end only. This restriction was discarded in [2], but the new variants were still able to solve in linear time another NP-complete problem, namely the “3-colorability problem”.

In the present paper, we consider hybrid networks of evolutionary processors in which each deletion or insertion node has its own working mode (at any position, in the left end, or in the right end) and its own way of defining the input and output filter. Thus, in the same network one may co-exist nodes in

which deletion is done at any position and nodes in which deletion is done in the right end only. Also the definition of the filters of two nodes, though both are random context ones, may differ.

This model may be viewed as a biological computing model in the following way: each node is a cell having a genetic information encoded in DNA sequences which may evolve by local evolutionary events, that is point mutations (insertion, deletion or substitution of a pair of nucleotides). Each node is specialized just for one of these evolutionary operations. Furthermore, the biological data in each node is organized in the form of arbitrarily large multisets of strings (each string appears in an arbitrarily large number of copies), each copy being processed in parallel such that all the possible evolutionary events that can take place do actually take place. Definitely, the computational process described here is not exactly an evolutionary process in the Darwinian sense. But the rewriting operations we have considered might be interpreted as mutations and the filtering process might be viewed as a selection process. Recombination is missing but it was asserted that evolutionary and functional relationships between genes can be captured by taking into consideration local mutations only [17]. Furthermore, we were not concerned here with a possible biological implementation, though a matter of great importance.

The paper is organized as follows: in the next section we recall the some basic notions from formal language theory and define the hybrid networks of evolutionary processors. Then, we briefly investigate the computational power of these networks as language generating devices. We prove that all regular languages over an n -letter alphabet can be generated in an efficient way by networks having the same underlying structure and show that this result can be extended to linear languages. Furthermore, we provide a non-context-free language which can be generated by such networks. The last section is dedicated to hybrid networks of evolutionary processors viewed as computing (problem solving) devices; we present two linear solutions of the so-called Common Algorithmic Problem. The latter one needs linearly bounded resources (symbols and rules) as well.

2 Preliminaries

We start by summarizing the notions used throughout the paper. An *alphabet* is a finite and nonempty set of symbols. The cardinality of a finite set A is written $\text{card}(A)$. Any sequence of symbols from an alphabet V is called *string* (*word*) over V . The set of all strings over V is denoted by V^* and the empty string is denoted by ε . The length of a string x is denoted by $|x|$ while the number of occurrences of a letter a in a string x is denoted by $|x|_a$. Furthermore, for each nonempty string x we denote by $\text{alph}(x)$ the minimal alphabet W such that $x \in W^*$.

We say that a rule $a \rightarrow b$, with $a, b \in V \cup \{\varepsilon\}$ is a *substitution rule* if both a and b are not ε ; it is a *deletion rule* if $a \neq \varepsilon$ and $b = \varepsilon$; it is an *insertion rule* if $a = \varepsilon$ and $b \neq \varepsilon$. The set of all substitution, deletion, and insertion rules over an alphabet V are denoted by Sub_V , Del_V , and Ins_V , respectively.

Given a rule as above σ and a string $w \in V^*$, we define the following *actions* of σ on w :

- If $\sigma \equiv a \rightarrow b \in Sub_V$, then

$$\sigma^*(w) = \sigma^r(w) = \sigma^l(w) = \begin{cases} \{ubv : \exists u, v \in V^* (w = uav)\}, \\ \{w\}, \text{ otherwise} \end{cases}$$

- If $\sigma \equiv a \rightarrow \varepsilon \in Del_V$, then

$$\sigma^*(w) = \begin{cases} \{uv : \exists u, v \in V^* (w = uav)\}, \\ \{w\}, \text{ otherwise} \end{cases}$$

$$\sigma^r(w) = \begin{cases} \{u : w = ua\}, \\ \{w\}, \text{ otherwise} \end{cases} \quad \sigma^l(w) = \begin{cases} \{v : w = av\}, \\ \{w\}, \text{ otherwise} \end{cases}$$

- If $\sigma \equiv \varepsilon \rightarrow a \in Ins_V$, then

$$\sigma^*(w) = \{uav : \exists u, v \in V^* (w = uv)\}, \quad \sigma^r(w) = \{wa\}, \quad \sigma^l(w) = \{aw\}.$$

$\alpha \in \{*, l, r\}$ expresses the way of applying an evolution rule to a word, namely at any position ($\alpha = *$), in the left ($\alpha = l$), or in the right ($\alpha = r$) end of the word, respectively. For every rule σ , action $\alpha \in \{*, l, r\}$, and $L \subseteq V^*$, we define the α -action of σ on L by $\sigma^\alpha(L) = \bigcup_{w \in L} \sigma^\alpha(w)$. Given a finite set of rules M , we define the α -action of M on the word w and the language L by:

$$M^\alpha(w) = \bigcup_{\sigma \in M} \sigma^\alpha(w) \quad \text{and} \quad M^\alpha(L) = \bigcup_{w \in L} M^\alpha(w),$$

respectively.

In what follows, we shall refer to the rewriting operations defined above as *evolutionary operations* since they may be viewed as linguistical formulations of local gene mutations. For two disjoint subsets P and F of an alphabet V and a word over V , we define the predicates

$$\begin{aligned} \varphi^{(1)}(w; P, F) &\equiv P \subseteq \text{alph}(w) \wedge F \cap \text{alph}(w) = \emptyset \\ \varphi^{(2)}(w; P, F) &\equiv \text{alph}(w) \subseteq P \\ \varphi^{(3)}(w; P, F) &\equiv P \subseteq \text{alph}(w) \wedge F \not\subseteq \text{alph}(w) \end{aligned}$$

The construction of these predicates is based on *random-context conditions* defined by the two sets P (*permitting contexts*) and F (*forbidding contexts*).

For every language $L \subseteq V^*$ and $\beta \in \{(1), (2), (3)\}$, we define:

$$\varphi^\beta(L, P, F) = \{w \in L \mid \varphi^\beta(w; P, F)\}.$$

An *evolutionary processor over V* is a tuple (M, PI, FI, PO, FO) , where:

- Either $(M \subseteq Sub_V)$ or $(M \subseteq Del_V)$ or $(M \subseteq Ins_V)$. The set M represents the set of evolutionary rules of the processor. As one can see, a processor is “specialized” in one evolutionary operation, only.
- $PI, FI \subseteq V$ are the *input* permitting/forbidding contexts of the processor, while $PO, FO \subseteq V$ are the *output* permitting/forbidding contexts of the processor.

We denote the set of evolutionary processors over V by EP_V .

A *hybrid network of evolutionary processors* (HNEP for short) is a 7-tuple $\Gamma = (V, G, N, C_0, \alpha, \beta, i_0)$, where:

- V is an alphabet.
- $G = (X_G, E_G)$ is an undirected graph with the set of vertices X_G and the set of edges E_G . G is called the *underlying graph* of the network.
- $N : X_G \rightarrow EP_V$ is a mapping which associates with each node $x \in X_G$ the evolutionary processor $N(x) = (M_x, PI_x, FI_x, PO_x, FO_x)$.
- $C_0 : X_G \rightarrow 2^{V^*}$ is a mapping which identifies the initial configuration of the network. It associates a finite set of words with each node of the graph G .
- $\alpha : X_G \rightarrow \{*, l, r\}$; $\alpha(x)$ gives the action mode of the rules of node x on the words existing in that node.
- $\beta : X_G \rightarrow \{(1), (2), (3)\}$ defines the type of the *input/output filters* of a node. More precisely, for every node, $x \in X_G$, the following filters are defined:

$$\text{input filter: } \rho_x(\cdot) = \varphi^{\beta(x)}(\cdot; PI_x, FI_x),$$

$$\text{output filter: } \tau_x(\cdot) = \varphi^{\beta(x)}(\cdot; PO_x, FO_x).$$

That is, $\rho_x(w)$ (resp. τ_x) indicates whether or not the string w can pass the input (resp. output) filter of x . More generally, $\rho_x(L)$ (resp. $\tau_x(L)$) is the set of strings of L that can pass the input (resp. output) filter of x .

- $i_0 \in X_G$ is the *output node* of the HNEP.

We say that $\text{card}(X_G)$ is the size of Γ . If $\alpha(x) = \alpha(y)$ and $\beta(x) = \beta(y)$ for any pair of nodes $x, y \in X_G$, then the network is said to be *homogeneous*. In the theory of networks some types of underlying graphs are common, e.g., rings, stars, grids, etc. We shall investigate here networks of evolutionary processors with their underlying graphs having these special forms. Thus a HNEP is said to be a *star*, *ring*, or *complete* HNEP if its underlying graph is a star, ring, grid, or complete graph, respectively. The star, ring, and complete graph with n vertices is denoted by S_n , R_n , and K_n , respectively.

A *configuration* of a HNEP Γ as above is a mapping $C : X_G \rightarrow 2^{V^*}$ which associates a set of strings with every node of the graph. A configuration may be understood as the sets of strings which are present in any node at a given moment. A configuration can change either by an *evolutionary step* or by a *communication step*. When changing by an evolutionary step, each component $C(x)$ of the configuration C is changed in accordance with the set of evolutionary rules M_x associated with the node x and the way of applying these rules $\alpha(x)$. Formally, we say that the configuration C' is obtained in *one evolutionary step* from the configuration C , written as $C \Rightarrow C'$, iff

$$C'(x) = M_x^{\alpha(x)}(C(x)) \text{ for all } x \in X_G.$$

When changing by a communication step, each node processor $x \in X_G$ sends one copy of each string it has, which is able to pass the output filter of x , to all the node processors connected to x and receives all the strings sent by any node processor connected with x providing that they can pass its input filter.

Formally, we say that the configuration C' is obtained in *one communication step* from configuration C , written as $C \vdash C'$, iff

$$C'(x) = (C(x) - \tau_x(C(x))) \cup \bigcup_{\{x,y\} \in E_G} (\tau_y(C(y)) \cap \rho_x(C(y))) \text{ for all } x \in X_G.$$

Let Γ an HNEP, the computation in Γ is a sequence of configurations $C_0, C_1, C_2, \dots$, where C_0 is the initial configuration of Γ , $C_{2i} \Longrightarrow C_{2i+1}$ and $C_{2i+1} \vdash C_{2i+2}$, for all $i \geq 0$. By the previous definitions, each configuration C_i is uniquely determined by the configuration C_{i-1} . If the sequence is finite, we have a finite computation. If one uses HNEPs as language generating devices, then the result of any finite or infinite computation is a language which is collected in the output node of the network.

For any computation $C_0, C_1, \dots$, all strings existing in the output node at some step belong to the language generated by the network. Formally, the language generated by Γ is $L(\Gamma) = \bigcup_{s \geq 0} C_s(i_0)$.

The time complexity of computing a finite set of strings Z is the minimal number s such that $Z \subseteq \bigcup_{t=0}^s C_t(i_0)$.

3 Computational Power of HNEP as Language Generating Devices

First, we compare these devices with the simplest generative grammars in the Chomsky hierarchy. In [2], one proves that the families of regular and context-free languages are incomparable with the family of languages generated by homogeneous networks of evolutionary processors. HNEPs are more powerful, namely

Theorem 1. *Any regular language can be generated by any type (star, ring, complete) of HNEP.*

Proof. Let $A = (Q, V, \delta, q_0, F)$ be a deterministic finite automaton; without loss of generality we may assume that $\delta(q, a) \neq q_0$ holds for each $q \in Q$ and each $a \in V$. Furthermore, we assume that $\text{card}(V) = n$. We construct the following complete HNEP (the proof for the other underlying structures is left to the reader):

$$\Gamma = (U, K_{2n+3}, N, C_0, \alpha, \beta, f).$$

The alphabet U is defined by $U = V \cup V' \cup Q \cup \{s_a \mid s \in Q, a \in V\}$, where $V' = \{a' \mid a \in V\}$. The set of nodes of the complete underlying graph is $\{x_0, x_1, x_f\} \cup V \cup V'$, and the other parameters are given in Table 1, where s and b are generic states from Q and symbols from V , respectively. One can easily prove by induction that

1. $\delta(q, x) \in F$ for some $q \in Q \setminus \{q_0\}$ if and only if $xq \in C_{8|x|}(0)$.
 2. x is accepted by A ($x \in L(A)$) if and only if $x \in C_p(f)$ for any $p \geq 8|x| + 1$.
- Therefore, $L(A)$ is exactly the language generated by Γ .

Table 1.

Node	M	PI	FI	PO	FO	C_0	α	β
x_0	$\{q \rightarrow s_b\}_{\delta(s,b)=q}$	$\emptyset$	$\{s_b\}_{s,b} \cup \{b'\}_b$	$\emptyset$	$\emptyset$	F	$*$	(1)
$a \in V$	$\varepsilon \rightarrow a'$	$\{s_a\}_s \cup V$	Q	U	$\emptyset$	$\emptyset$	l	(2)
$a' \in V'$	$\{s_a \rightarrow s\}_s$	$\{a'\}$	Q	$\emptyset$	$\emptyset$	$\emptyset$	$*$	(1)
x_1	$\{b' \rightarrow b\}_b$	$\emptyset$	$\{s_b\}_{s,b}$	$\emptyset$	$\emptyset$	$\emptyset$	$*$	(1)
x_f	$q_0 \rightarrow \varepsilon$	$\{q_0\}$	V'	$\emptyset$	V	$\emptyset$	r	(1)

Surprisingly enough, the size of the above HNEP, hence its underlying structure, does not depend on the number of states of the given automaton. In other words, this structure is common to all regular languages over the same alphabet, no matter the state complexity of the automata recognizing them. Furthermore, all strings of the same length are generated simultaneously.

Since each linear grammar can be transformed into an equivalent linear grammar with rules of the form $A \rightarrow aB$, $A \rightarrow Ba$, $A \rightarrow \varepsilon$ only, the proof of the above theorem can be adapted for proving the next result.

Theorem 2. *Any linear language can be generated by any type of HNEP.*

We do not know whether these networks are able to generate all context-free languages, but they can generate non-context-free languages as shown below.

Theorem 3. *There are non-context-free languages that can be generated by any type of HNEP.*

Proof. We construct the following complete HNEP which generates the non-context-free language $L = \{wcx \mid x \in \{a, b\}^*, w \text{ is a permutation of } x\}$:

$$\Gamma = (V, K_9, N, C_0, \alpha, \beta, y_2),$$

where $V = \{a, b, a', b', X_a, X_b, X\}$, $X_{K_9} = \{y_0, y_1, y_2, y_a, y_b, \bar{y}_a, \bar{y}_b, \tilde{y}_a, \tilde{y}_b\}$, and the other parameters are given in Table 2, where u is a generic symbol in $\{a, b\}$. The working mode of this network is rather simple. In the node y_0 there are generated strings of the form X^n for any $n \geq 1$. They can leave this node as soon as they receive a D at their right end, the only node able to receive them being y_1 . In y_1 , either X_a or X_b is added to their right end. Thus, for a given n , the strings X^nDX_a and X^nDX_b are produced in y_1 . Let us follow what happens with the strings X^nDX_a , a similar analysis applies to the strings X^nDX_b as well. So, X^nDX_a goes to y_a where any occurrence of X is replaced by a' in different identical copies of X^nDX_a . In other words, y_a produces each string $X^ka'X^{n-k-1}DX_a$, $0 \leq k \leq n-1$. All these strings are sent out but no node, except $\bar{y}_a$, can receive them. Here, X_a is replaced by a and the obtained strings are sent to $\tilde{y}_a$ where a is substituted to a' . As long as the strings contains occurrences of X , they follow the same itinerary, namely $y_1, y_u, \bar{y}_u, \tilde{y}_u$, $u \in \{a, b\}$, depending on what symbol X_a or X_b is added in y_1 .

After a finite number of such cycles, when no occurrence of X is present in the strings, they are received by y_2 where D is replaced by c in all of them, and

Table 2.

Node	M	PI	FI	PO	FO	C_0	α	β
y_0	$\{\varepsilon \rightarrow X, \varepsilon \rightarrow D\}$	$\emptyset$	$\{a', b', a, b, X_a, X_b\}$	$\{D\}$	$\emptyset$	$\{\varepsilon\}$	r	(1)
y_1	$\{\varepsilon \rightarrow X_a, \varepsilon \rightarrow X_b\}$	$\emptyset$	$\{X_a, X_b, a', b'\}$	$\emptyset$	$\emptyset$	$\emptyset$	r	(1)
y_u	$\{X \rightarrow u'\}$	$\{X_u\}$	$\{a', b'\}$	$\emptyset$	$\emptyset$	$\emptyset$	$*$	(1)
$\bar{y}_u$	$\{X_u \rightarrow u\}$	$\{u'\}$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$*$	(1)
$\tilde{y}_u$	$\{u' \rightarrow u\}$	$\{u'\}$	$\{X_a, X_b\}$	$\emptyset$	$\emptyset$	$\emptyset$	$*$	(1)
y_2	$\{D \rightarrow c\}$	$\emptyset$	$\{X, a', b', X_a, X_b\}$	$\emptyset$	$\{a, b\}$	$\emptyset$	$*$	(1)

they remain in this node for ever. By these explanations, the node y_2 collects all strings of L and any string which arrives in this node belongs to L .

A more precise characterization of the family of languages generated by HNEPs remains to be done.

4 Solving Problems with HNEPs

HNEPs may be used for solving problems in the following way. For any instance of the problem the computation in the associated HNEP must be finite. In particular, this means that there is no node processor specialized in insertions. If the problem is a decision problem, then at the end of the computation, the output node provides all solutions of the problem encoded by strings, if any, otherwise this node will never contain any word. If the problem requires a finite set of words, this set will be in the output node at the end of the computation. In other cases, the result is collected by specific methods which will be indicated for each problem.

In [2] one provides a complete homogeneous NEP of size $7m + 2$ which solves in $O(m + n)$ time an (n, m) -instance of the “3-colorability problem” with n vertices and m edges. In the sequel, following the descriptive format for three NP-complete problems presented in [9] we present a solution to the *Common Algorithmic Problem*. The three problems are:

1. The *maximum independent set*: Given an undirected graph $G = (X, E)$, where X is the finite set of vertices and E is the set of edges given as a family of sets of two vertices, find the cardinality of a maximal subset (with respect to inclusion) of X which does not contain both vertices connected by any edge in E .
2. The *vertex cover* problem: Given an undirected graph find the cardinality of a minimal set of vertices such that each edge has at least one of its extremes in this set.
3. The *satisfiability* problem: For a given set P of Boolean variables and a finite set U of clauses over P , does a truth assignment for the variables of P exist satisfying all the clauses of U ?

For detailed formulations and discussions about their solutions, the reader is referred to [8].

These problems can be viewed as special cases of the following algorithmic problem, called the Common Algorithmic Problem (CAP) in [9]: let S be a finite set and F be a non-empty family of subsets of S . Find the cardinality of a maximal subset of S which does not include any set belonging to F . The sets in F are called *forbidden sets*. We say that (F, S) is an $(\text{card}(S), \text{card}(F))$ -instance of CAP

Let us show how the three problems mentioned above can be obtained as special cases of CAP. For the first problem, we just take $S = X$ and $F = E$. The second problem is obtained by letting $S = X$ and F contains all sets $o(x) = \{x\} \cup \{y \in X \mid \{x, y\} \in E\}$. The cardinality one looks for is the difference between the cardinality of S and the solution of the CAP. The third problem is obtained by letting $S = P \cup P'$, where $P' = \{p' \mid p \in P\}$, and $F = \{F(C) \mid C \in U\}$, where each set $F(C)$ associated with the clause C is defined by

$$F(C) = \{p' \mid p \text{ appears in } C\} \cup \{p \mid \neg p \text{ appears in } C\}.$$

From this it follows that the given instance of the satisfiability problem has a solution if and only if the solution of the constructed instance of the CAP is exactly the cardinality of P .

First, we present a solution of the CAP based on homogeneous HNEPs.

Theorem 4. *Let $(S = \{a_1, a_2, \dots, a_n\}, F = \{F_1, F_2, \dots, F_m\})$, be an (n, m) -instance of the CAP. It can be solved by a complete homogeneous HNEP of size $m + 2n + 2$ in $O(m+n)$ time.*

Proof. We construct the complete homogeneous HNEP

$$\Gamma = (U, K_{m+2n+2}, N, C_0, \alpha, \beta).$$

Since the result will be collected in a way which will be specified later, the output node is missing. The alphabet of the network is

$$U = S \cup \bar{S} \cup S' \cup \{Y, Y_1, Y_2, \dots, Y_{m+1}\} \cup \{b\} \cup \{Z_0, Z_1, \dots, Z_n\} \cup \{Y'_1, Y'_2, \dots, Y'_{m+1}\} \cup \{X_1, X_2, \dots, X_n\},$$

where $\bar{S}$ and S' are copies of S obtained by taking the barred and primed copies of all letters from S , respectively. The nodes of the underlying graph are:

$$x_0, x_{F_1}, x_{F_2}, \dots, x_{F_m}, x_{a_1}, x_{a_2}, \dots, x_{a_n}, y_0, y_1, \dots, y_n.$$

The mapping N is defined by:

$$\begin{aligned} N(x_0) &= (\{X_i \rightarrow a_i, X_i \rightarrow \bar{a}_i \mid 1 \leq i \leq n\} \cup \{Y \rightarrow Y_1\} \cup \{Y'_i \rightarrow Y_{i+1} \mid \\ &\quad 1 \leq i \leq m\}, \{Y'_i \mid 1 \leq i \leq m\}, \emptyset, \emptyset, \{X_i \mid 1 \leq i \leq n\} \cup \{Y\}), \\ N(x_{F_i}) &= (\{\bar{a} \rightarrow a' \mid a \in F_i\}, \{Y_i\}, \emptyset, \emptyset, \emptyset), \text{ for all } 1 \leq i \leq m, \\ N(x_{a_j}) &= (\{a'_j \rightarrow \bar{a}_j\} \cup \{Y_i \rightarrow Y'_i \mid 1 \leq i \leq m\}, \{a'_j\}, \emptyset, \emptyset, \{a'_j\} \cup \{Y_i \mid \\ &\quad 1 \leq i \leq m\}), \text{ for all } 1 \leq j \leq n, \\ N(y_n) &= (\{\bar{a}_i \rightarrow b \mid 1 \leq i \leq n\} \cup \{Y_{m+1} \rightarrow Z_0\}, \{Y_{m+1}\}, \emptyset, \{Z_0, b\}, \bar{S}), \\ N(y_{n-i}) &= (\{b \rightarrow Z_i\}, \{Z_{i-1}\}, \emptyset, \{b, Z_i\}, \emptyset), \text{ for all } 1 \leq i \leq n. \end{aligned}$$

The initial configuration C_0 is defined by

$$C_0(x) = \begin{cases} \{X_1 X_2 \dots X_n Y\} & \text{if } x = x_0 \\ \emptyset, & \text{otherwise} \end{cases}$$

Finally, $\alpha(x) = *$ and $\beta(x) = (1)$, for any node x .

A few words on how the HNEP above works: in the first $2n$ steps, in the first node one obtains 2^n different words $w = x_1 x_2 \dots x_n Y$, where each x_i is either a_i or $\bar{a}_i$. Each such string w can be viewed as encoding a subset of S , namely the set containing all symbols of S which appear in w . After replacing Y by Y_1 in all these strings they are sent out and x_{F_1} is the only node which can receive them. After one rewriting step, only those strings encoding subsets of S which do not include F_1 will remain in the network, the others being lost. The strings which remain are easily recognized since they have been obtained by replacing a barred copy of symbol with a primed copy of the same symbol. This means that this symbol is not in the subset encoded by the string but in F_1 . In the nodes x_{a_i} the modified barred symbols are restored and the symbol Y'_1 is substituted for Y_1 . Now, the strings go to the node x_0 where Y_2 is substituted for Y'_1 and the whole process above resumes for F_2 . This process lasts for $8m$ steps.

The last phase of the computation makes use of the nodes y_j , $0 \leq j \leq n$. The number we are looking for is given by the largest number of symbols from S in the strings from y_n . It is easy to note that the strings which cannot leave y_{n-i} have exactly $n-i$ such symbols, $0 \leq i \leq n$. Indeed, only the strings which contains at least one occurrence of b can leave y_n and reach y_{n-1} . Those strings which do not contain any occurrence of b have exactly n symbols from S . In y_{n-1} , Z_1 is substituted for an occurrence of b and those strings which still contain b leave this node for y_{n-2} and so forth. The strings which remain here contain $n-1$ symbols from S . Therefore, when the computation is over, the solution of the given instance of the CAP is the largest j such that y_j is nonempty. The last phase is over after at most $4n+1$ steps. By the aforementioned considerations, the total number of steps is at most $8m+4n+3$, hence the time complexity of solving each instance of the CAP of size (n, m) is $O(m+n)$.

As far as the time and memory resources the HNEP above uses, the total number of symbols is $2m+5n+4$ and the total number of rules is

$$mn + m + 5n + 2 + \sum_{i=1}^m \text{card}(F_i) \in \Theta(mn)$$

The same problem can be solved in a more economic way, regarding especially the number of rules, with HNEPs, namely

Theorem 5. *Any instance of the CAP can be solved by a complete HNEP of size $m+n+1$ in $O(m+n)$ time.*

Proof. For the same instance of the CAP as in the previous proof, we construct the complete HNEP $\Gamma = (U, K_{m+n+1}, N, C_0, \alpha, \beta)$.

The alphabet of the network is $U = S \cup S' \cup \{Y_1, Y_2, \dots, Y_{m+1}\} \cup \{b\} \cup \{Z_0, Z_1, \dots, Z_n\}$. The other parameters of the network are given in Table 3.

Table 3.

Node	M	PI	FI	PO	FO	C_0	α	β
x_0	$\{a'_i \rightarrow a_i\}_i$ $\{a'_i \rightarrow T\}_i$	$\{a'_1\}$	$\emptyset$	$\emptyset$	$\{a'_i\}_i$	$\{a'_1 \dots a'_n Y_1\}$	*	(1)
x_{F_j}	$\{Y_j \rightarrow Y_{j+1}\}$	$\{Y_j\}$	F_j	$\emptyset$	U	$\emptyset$	*	(3)
y_n	$\{T \rightarrow Z_0\}$	$\{Y_{m+1}\}$	$\emptyset$	$\{T\}$	$\emptyset$	$\emptyset$	*	(1)
y_{n-i}	$\{T \rightarrow Z_i\}$	$\{Z_{i-1}\}$	$\emptyset$	$\{T\}$	$\emptyset$	$\emptyset$	*	(1)

In the table above, i ranges from 1 to n and j ranges from 1 to m . The reasoning is rather similar to that from the previous proof. The only notable difference concerns the phase of selecting all strings which do not contain any symbol from any set F_j . This selection is simply accomplished by the way of defining the filters of the nodes x_{F_j} . The time complexity is now $2m + 4n + 1 \in O(m + n)$, while the needed resources are: $m + 3n + 3$ symbols and $m + 3n + 1$ rules.

5 Concluding Remarks and Future Work

We have considered a mechanism inspired from cell biology, namely hybrid networks of evolutionary processors, that is networks whose nodes are very simple processors able to perform just one type of point mutation (insertion, deletion or substitution of a symbol). These nodes are endowed with a filter which is defined by some random context conditions which seem to be close to the possibilities of biological implementation. A rather suggestive view of these networks is that of a group of connected cells that are similar to each other and have the same purpose, that is a *tissue*. It is worth mentioning some similarities with the membrane systems defined in [16]. In that work, the underlying structure is a tree and the (biological) data is transferred from one region to another by means of some rules. A more closely related protocol of transferring data among regions in a membrane system was considered in [15].

We finish with a natural question: We are conscious that our mechanisms have likely no biological relevance. Then why to study them? We believe that by combining our knowledge about behavior of cell populations with advanced formal theories from computer science, we could try to define computational models based on the interacting molecular entities. To this aim we need to accomplish the followings:

- (1) Understanding which features of the behavior of molecular entities forming a biological system can be used for designing computing networks with an underlying structure inspired from that of the biological system.
- (2) Understanding how to control the data navigating in the networks via precise protocols
- (3) Understanding how to effectively design the networks.

The results obtained in this paper suggest that these mechanisms might be a reasonable example of global computing due to the real and massively parallelism

involved in molecular interactions. Therefore, they deserve a deep theoretical investigation as well as an investigation of biological limits of implementation in our opinion.

References

1. Castellanos, J., Martín-Vide, C., Mitrana, V., Sempere, J.: Solving NP-complete problems with networks of evolutionary processors. IWANN 2001 (J. Mira, A. Prieto, eds.), LNCS 2084, Springer-Verlag (2001) 621–628.
2. Castellanos, J., Martín-Vide, C., Mitrana, V., Sempere, J.: Networks of evolutionary processors. Submitted (2002).
3. Csuhaj-Varjú, E., Dassow, J., Kelemen, J., Păun, G.: *Grammar Systems*, Gordon and Breach, 1993.
4. Csuhaj-Varjú, E., Salomaa, A.: Networks of parallel language processors. New Trends in Formal Languages (Gh. Păun, A. Salomaa, eds.), LNCS 1218, Springer Verlag (1997) 299–318.
5. Csuhaj-Varjú, E., Mitrana, V.: Evolutionary systems: a language generating device inspired by evolving communities of cells. Acta Informatica **36** (2000) 913–926.
6. Errico, L., Jesshope, C.: Towards a new architecture for symbolic processing. Artificial Intelligence and Information-Control Systems of Robots '94 (I. Plander, ed.), World Sci. Publ., Singapore (1994) 31–40.
7. Fahlman, S.E., Hinton, G.E., Sejnowski, T.J.: Massively parallel architectures for AI: NETL, THISTLE and Boltzmann machines. Proc. AAAI National Conf. on AI, William Kaufman, Los Altos (1983) 109–113.
8. Garey, M., Johnson, D.: *Computers and Intractability. A Guide to the Theory of NP-completeness*, Freeman, San Francisco, CA, 1979.
9. Head, T., Yamamura, M., Gal, S.: Aqueous computing: writing on molecules. Proc. of the Congress on Evolutionary Computation 1999, IEEE Service Center, Piscataway, NJ (1999) 1006–1010.
10. Kari, L.: *On Insertion and Deletion in Formal Languages*, Ph.D. Thesis, University of Turku, 1991.
11. Kari, L., Păun, G., Thierrin, G., Yu, S.: At the crossroads of DNA computing and formal languages: Characterizing RE using insertion-deletion systems. Proc. 3rd DIMACS Workshop on DNA Based Computing, Philadelphia (1997) 318–333.
12. Kari, L., Thierrin, G.: Contextual insertion/deletion and computability. Information and Computation **131** (1996) 47–61.
13. Hillis, W.D.: *The Connection Machine*, MIT Press, Cambridge, 1985.
14. Martín-Vide, C., Păun, G., Salomaa, A.: Characterizations of recursively enumerable languages by means of insertion grammars. Theoretical Computer Science **205** (1998) 195–205.
15. Martín-Vide, Mitrana, V., Păun, G.: On the power of valuations in P systems. Computacion y Sistemas **5** (2001) 120–128.
16. Păun, G.: Computing with membranes. J. Comput. Syst. Sci. **61**(2000) 108–143.
17. Sankoff, D. et al.: Gene order comparisons for phylogenetic inference: Evolution of the mitochondrial genome. Proc. Natl. Acad. Sci. USA **89** (1992) 6575–6579.

DNA-Like Genomes for Evolution *in silico*

Michael West, Max H. Garzon, and Derrel Blain

Computer Science, University of Memphis

373 Dunn Hall, Memphis, TN 38152

{mrwest1, mgarzon}@memphis.edu, derrelb@earthlink.net

Abstract. We explore the advantages of DNA-like genomes for evolutionary computation *in silico*. Coupled with simulations of chemical reactions, these genomes offer greater efficiency, reliability, scalability, new computationally feasible fitness functions, and more dynamic evolutionary algorithms. The prototype application is the decision problem of HPP (the Hamiltonian Path Problem.) Other applications include pre-processing of protocols for biomolecular computing and novel fitness functions for evolution *in silico*.

1 Introduction

The advantages of using DNA molecules for advances in computing, known as biomolecular computing (BMC), have been widely discussed [1], [3]. They range from increasing speed by using massively parallel computations to the potential storage of huge amounts of data fitting into minuscule spaces. Evolutionary algorithms have been used to find word designs to implement computational protocols [4]. More recently, driven by efficiency and reliability considerations, the ideas of BMC have been explored for computation *in silico* by using computational analogs of DNA and RNA molecules [5]. In this paper, a further step with this idea is taken by exploring the use of DNA-like genomes and online fitness for evolutionary computation.

The idea of using sexually split genomes (based on pair attraction) has hardly been explored in evolutionary computation and genetic algorithms. Overwhelming evidence from biology shows that “the [evolutionary] essence of sex is Mendelian recombination” [11]. DNA is the basic genomic representation of virtually all life forms on earth. The closest approach of this type is the DNA-based computing approach of Adleman [1]. We show that an interesting and intriguing interplay can exist between the ideas of biomolecular-based and silicon-based computation. By enriching Adleman’s solution to the Hamiltonian Path Problem (HPP) with fitness-based selection in a population of potential solutions, we show how these algorithms can exploit biomolecular and traditional computing techniques for improving solutions to HPP on conventional computers. Furthermore, it is conceivable that these fitness functions may be implemented *in vitro* in the future, and so improve the efficiency and reliability of solutions to HPP with biomolecules as well.

In Section 2, we describe the experiments performed for this purpose, including the programming environment and the genetic algorithms based on DNA-like genomes. In Section 3, we discuss the results of the experiments. A preliminary analysis of some of these results has been presented in [5], but here we present further results and a more complete analysis. Finally, we summarize the results, discuss the implications of genetic computation, and envision further work.

2 Experimental Design

As our prototype we took the problem that was used by Adleman [1], the Hamiltonian Path Problem (HPP), for a proof-of-concept to establish the feasibility of DNA-based computation. An instance of the problem is a digraph and a given source and destination; the problem is to determine whether there exists a path from the source to the destination that passes through each vertex in the digraph exactly once. Solutions to this problem have a wide-ranging impact in combinatorial optimization areas such as route planning and network efficiency.

In Adleman's solution [1], the problem is solved by encoding vertices of the graph with unique strands of DNA and encoding edges so that their halves will hybridize with the end vertex molecules. Once massive numbers of these molecules are put in a test tube, they will hybridize in multiple ways and form longer molecules ultimately representing all possible paths in the digraph. To find a Hamiltonian path, various extraction steps are taken to filter out irrelevant paths, such as those not starting at the source vertex or ending at the destination. Good paths must also have exactly as many vertices as there are in the graph, and each vertex has to be unique within the final path. Any paths remaining represent desirable solution Hamiltonian paths.

There have been several improvements on this technique. In [10], the authors attempt to automate Adleman's solution so that the protocols more intelligently construct promising paths. Another improvement [2] uses reflective PCR to restrict or eliminate duplicated vertices in paths. In [8], the authors extend Adleman's solution, by adding weights associated with melting temperatures to solve another NP-complete problem, the Traveling Salesman Problem (TSP).

We further these genetic techniques by adding several on-line fitness functions for an implementation *in silico*. By rewriting these biomolecular techniques within the framework of traditional computing, we hope to begin the exploration of algorithms based on concepts inspired by BMC. In this case, a large population of possible solutions is evolved in a process that is also akin to a developmental process. Specifically, a population of partially formed solutions is maintained that could react (hybridize), in a pre-specified manner, with other partial solutions within the population to form a more complete (fitter) solution. Several fitness functions ensure that the new solution inherits the good traits of the mates in the hybridization. For potential future implementation *in vitro*, the fitness functions are kept consistent with biomolecular computing by placing the genomes within a simulation of a test tube to allow for random movement and interaction. Fitness evaluation is thus more attuned to developmental

and environmental conditions than customary fitness functions solely dependent on genome composition.

2.1 Virtual Test Tubes

The experimental runs were implemented using an electronic simulation of a test tube, the virtual test tube *Edna* of Garzon et al. [5], [7] which simulates BMC protocols *in silico*. As compared to a real test tube, *Edna* provides an environment where DNA analogs can be manipulated much more efficiently, can be programmed and controlled much more easily, cost much less, and produce results comparable to real test tubes [5]. Users simply need to create object-oriented programming classes (in C++) specifying the *objects* to be used and their *interactions*. The basic design of the entities that are put in *Edna* represents each nucleotide within DNA strands as a single character and the entire strand of DNA as a string, which may contain single- or double-stranded sections, bulges, and other secondary structures. An unhybridized strand represents a strand of DNA from the 5'-end to the 3'-end. In addition to the actual DNA strand composition, other statistics were also saved such as the vertices making up the strand and the number of encounters since extension.

The interactions among objects in *Edna* are chemical reactions through hybridizations and ligations resulting in longer paths. They can result in one or both reactants being destroyed and a new entity possibly being created. In our case, we wanted to allow the entities that matched to hybridize to each other's ends so that an edge could hybridize to its adjacent vertex. We called this reaction *extension* since the path, vertex, or edge represented by one entity is extended by the path, vertex, or edge represented by the other entity, in analogy with the PCR reaction used with DNA. *Edna* simulates the reactions in successive *iterations*. One iteration moves the objects randomly in the tube's container (the RAM really) and updates their status according to the specified interactions based on proximity parameters that can be varied within the interactions. The hybridization reactions between strands were controlled by the h-distance [6] of hybridization affinity. Roughly speaking, the h-distance between two strands provides the number of Watson-Crick mismatching pairs in a best alignment of the two strands; strands at distance 0 are complementary, while the hybridization affinity decreases as the h-distance increases. Extension was allowed if the h-distance was zero (which would happen any time the origin or destination of a path hybridized with one of its adjacent edges); or half the length of a single vertex or edge (such as when any vertex encountered an adjacent edge); or, more generally, when two paths, both already partially hybridized, encountered each other, and each had an unhybridized segment (of length equal to half the length of a vertex or edge) representing a matching vertex and edge. These requirements essentially ensured perfect matches along the sections of the DNA that were supposed to hybridize. Well-chosen DNA encodings make this perfectly possible in real test tubes [4].

The complexity of the test tube protocols can be measured by counting the number of iterations necessary to complete the reactions or achieve the desired objective. Alternatively, one can measure the wall clock time. The *number of iterations* taken be-

fore a correct path is found has the advantage of being indifferent to the speed of the machine(s) running the experiment. However, it cannot be a complete picture because each iteration will last longer as more entities are put in the test tube. For this reason, *processor time* (wall clock) was also measured.

2.2 Fitness Functions

Our genetic approach to solving HPP used fitness functions to be enforced online as the reactions proceeded. The first stage, which was used as a benchmark, included checks that vertices did not repeat themselves, called *promise* fitness. This original stage also enforced a constant number of the initial vertices and edges in the test tube in order to ensure an adequate supply of vertices and edges to form paths as needed.

Successive refinements improve on the original by using three types of fitnesses: *extension* fitness, *demand* fitness, and *repetition* fitness, as described below. The goal in adding these fitnesses was to improve the efficiency of path formation. The purpose of the fitnesses implemented here was to bring down the number of iterations it took to find a solution since Edna's speed, although parallel, decreases with more DNA. Toward this goal, we aimed at increasing the opportunity for an object to encounter another object that is likely to lead to a correct path. This entailed increasing the quantity of entities that seemed to lead to a good path (were more fit) and decreasing the concentration of those entities that were less fit. By removing the unlikely paths, we moved to improve the processor time by lowering the overall concentration in the test tube. At this point, the only method to regulate which of its adjacent neighbors an entity encounters is by adjusting the concentration and hence adjusting the probability that its neighbors are of a particular type.

Promise Fitness. As part of the initial design, we limited the type of extensions that were allowed to occur beyond the typical requirement of having matching nucleotides and an h-distance as described above. Any two entities that encountered each other could only hybridize if they did not contain any repeated vertices. It was checked during the encounter by comparing a list of vertices that were represented by each strand of DNA. A method similar to this was proposed in [2] to work *in vitro*. As a consequence, much of the final screening otherwise needed to find the correct path was eliminated. Searching for a path can stop once one is found that contains as many vertices as are in the graph. Since all of the vertices are guaranteed to be unique, this path is guaranteed to pass through all of the vertices in the graph. Because the origin and destination are encoded as half the length of any other vertex, the final path's strand can only have them on the two opposite ends and hence the path travels from the origin to the destination.

Constant Concentration Enhancement. The initial design also kept the concentration of the initial vertices and edges constant. Simply put, whenever vertices and edges encountered each other and were extended, neither of the entities was removed although the new entity was still put into the test tube. It is as if the two original entities were copied before they hybridized and all three were returned to the mixture. The same mechanism was used when the encountering objects were not single vertices or edges but instead were paths. This, however, did not guarantee that the concentration of any type of path remained constant since new paths could still be created. The motivation behind this enhancement was to allow all possible paths to be created without worrying about running out of some critical vertex or edge. It also removed some of the complications about different initial concentrations of certain vertices or edges and what paths may be more likely to be formed. However, this fitness, while desirable and enforceable *in silico* (although not easily *in vitro* just yet) creates a huge number of molecules that made the simulation slow and inefficient.

Extension Fitness. The most obvious paths to be removed are lazy paths that are not being extended. These paths could be stuck in dead-ends where no extension to a Hamiltonian path is possible. To make finding them easier, all paths were allowed to have the same, limited number of encounters without being extended (an initial lifespan) which, when met, would result in their being removed from the tube. If, however, a path was extended before meeting its lifespan then the lifespan of both reacting objects was increased by 50%. The new entity created during an extension received the larger lifespan of its two parents.

Demand Fitness. The concentration of vertices and edges in the tube can be tweaked based on the demand for each entity to participate in reactions. The edges that are used most often (e.g., bridge edges) have a high probability of being in a correct Hamiltonian path since they are likely to be a single or critical connection between sections of the graph. Hence we increase the concentration of edges that are used the most often. Since all vertices must be in a correct solution, those vertices that are not extended often have a disadvantage in that they are less likely to be put into the final solution. In order to remedy this, vertices that are not used often have their concentration increased. The number of encounters and the number of extensions for each entity was stored so a ratio of extensions to encounters was used to implement demand fitness. To prevent the population of vertices and edges from getting out of control, we set a maximum number of any individual vertex or edge to eight unless otherwise noted.

Repetition Fitness. To prevent the tube from getting too full with identical strands, repetition fitness was implemented. It filtered out low performing entities that were repeated often throughout the tube. Whenever an entity encountered another entity, the program checked to see if they encoded the same information. If they did, then they did not extend, and they increased their count of encounters with the same path. Once a path encountered a duplicate of itself too many times, it was removed if it was a low enough performer in terms of its ratio of extensions to encounters.

2.3 Test Graphs and Experimental Conditions

Graphs for the experiments were made using Model A of random graphs [12]. Given a number of vertices, an edge existed between two vertices with probability given by a parameter $p = (0.2, 0.4, \text{ or } 0.6)$ of including an edge (more precisely, an arc) from the set of all possibilities. For positive instances, one witness Hamiltonian path was placed randomly connecting source to destination. For negative instances, the vertices were divided into two random sets, one containing the origin and one containing the destination; no path was allowed to connect the origin set to the set containing the destination, although the reverse was allowed so that the graph may be connected.

The input to *Edna* was a set of non-crosshybridizing strands of size 64 consisting of 20-oligomers designed by a genetic algorithm using the h-distance as fitness criterion. One copy of each vertex and edge was placed initially in the tube. The quality of the encoding set is such that even under a mildly stringent hybridization criterion, two sticky ends will not hybridize unless they're perfect Watson-Crick complements. In the first set of experiments, the retrieval time was measured in a variety of conditions including variable library concentration, variable probe concentrations, and joint variable concentration. At first, we permitted only paths that were *promising* to become Hamiltonian. Later, other fitness constraints were added to make the path assembly process smarter as discussed below with the results.

Each experiment was broken down into many different runs of the application all with related configurations. All of the experiments went through several repetitions where one or two parameters were slightly changed so that we could evaluate the differences over these parameters (number of vertices and edge density), although we sometimes changed other parameters such as maximum concentration allowed, maximum number of repeated paths, or tube size. Unless otherwise noted, all repetitions were run 30 times with the same parameters, although a different randomly generated graph was used for each run. We report below the averages of the various performance measures. A run was considered unsuccessful if it went through 3000 iterations without finding a correct solution, in which case the run was not included within the averages. We began with the initial implementation as discussed above and added each fitness so that each could be studied without the other fitnesses interfering. Finally we investigated the scalability of our algorithms by adding a population control parameter and running the program on graphs with more vertices.

3 Analysis of Results

The initial implementation provided us with a benchmark from which to judge the fitness efficiency. In terms of iterations (Fig. 1, left) and processor time (Fig. 1, right), the results of this first experiment are not at all surprising. Both measures increase as the number of vertices increases. There is also a noticeable trend where the 40% edge densities take the most time. Edge density of 20% is faster because the graph contains fewer possible paths to search through whereas 60% edge density shows a decrease in time of search because the additional edges provide significantly more correct solu-

tions. It should be noted that altogether there were only two unsuccessful attempts, both with 9 vertices, one at 20% edge density and the other at 40% edge density. This places the probability of success with these randomized graphs above 99%.

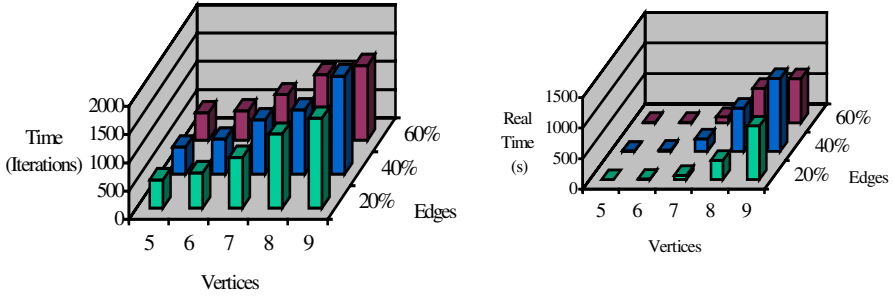


Fig. 1. Successful completion time for the baseline runs (only unique vertices and constant concentration restrictions in force) in number of iterations (left) and processor time (right)

The first comparison made was with extension fitness. The test was done with the initial lifespan set to 150 and the maximum lifespan also set to 150. As seen in Fig. 2, the result cut the number of iterations 54% for 514 fewer iterations on average.

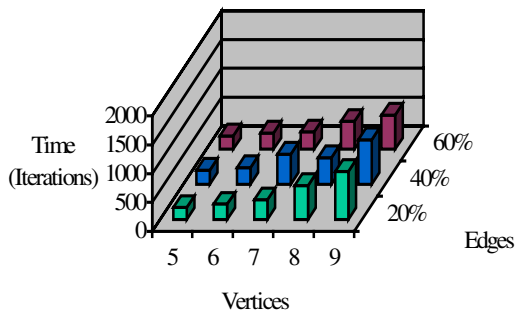


Fig. 2. Successful completion times with extension fitness

From what data is available at this time, demand fitness did not show as impressive an improvement as extension fitness although it still seemed to help. The greatest gain from this fitness is expected to be for graphs with larger numbers of vertices where small changes in the number of vertices and edges will have more time to have a large effect. The number of iterations recorded, on average, can be seen in Fig. 3. The minimum ratio of extensions to encounters before an edge was copied, the edge ratio, was set to .17. The maximum ratio of extensions to encounters below which a vertex

was copied, the vertex ratio, was set to .07. Although it was not measured, the processor time for this fitness seemed to be considerably greater than that of the other fitnesses.

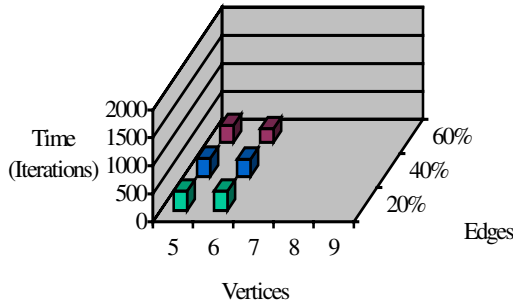


Fig. 3. Successful completion times with demand fitness

The last fitness to be implemented, repetition fitness, provided a 49% decrease in iterations resulting in 465 less iterations on average (Fig. 4). The effect seems to become especially pronounced as the number of vertices increases.

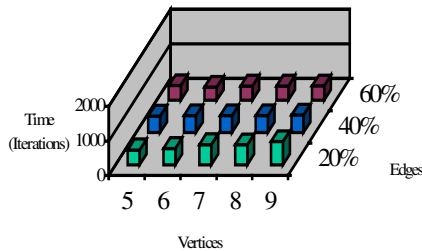


Fig. 4. Successful completion times with the addition of repetition fitness

Finally, we combined all of the fitnesses together. The results can be seen in Fig. 5 in terms of iterations (left) and in terms of processor time (right). Note that the scale for both graphs changed from the comparable ones above. We also increased the radius of each entity from one to two. The initial lifespan of entities was 140, and it was allowed to reach a maximum lifespan of 180. The edge ratio was set to .16, and the vertex ratio was set to .07. For demand fitness, the number of paths allowed was 20, and the removal ratio was .04. All of the fitnesses running together resulted in decreasing the number of iterations by 93% for 880 iterations less, on average. The processor time was cut by 69% saving, on average, 219.90 seconds per run.

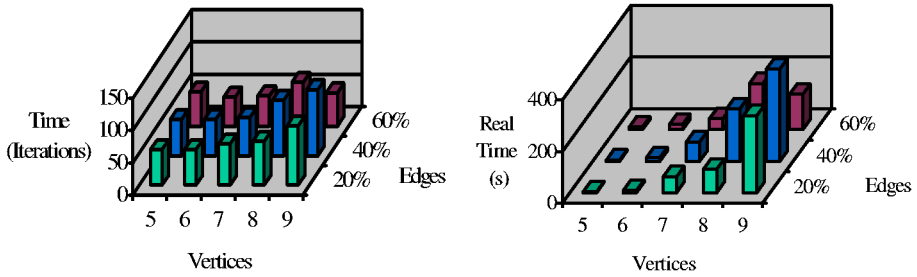


Fig. 5. Successful completion time with all fitnesses running in terms of number of iterations (left) and running time (right)

An important objective of these experiments is to explore the limits of Adleman's approach, at least *in silico*. What is the largest problem that could be solved? In order to allow the program to run on graphs with large numbers of vertices, we put an upper limit on the number of entities present in the tube at any time. Each entity, of course, takes up a certain amount of memory and processing time so this limitation would help keep the program's memory usage in check. Unfortunately, when the limit on the number of entities is reached, the fitnesses, if they are configured with reasonable settings, will not remove very many paths during each iteration meaning that many new paths cannot be added. The dark red line in Fig. 6 shows the results; as the number of entities in the tube reaches the maximum, only a small number of entities are removed, thus not allowing room for many new entities to be created and preventing new, possibly good paths, from forming.

It is necessary to not only limit the population but also to control it. The desired effect would be for the fitnesses to be aggressive as the entity count nears the maximum and reasonable as it falls back down to some minimum. Additionally it would be advantageous for the more aggressive settings to be applied to shorter paths and not longer ones since the shorter paths can be remade much faster than the longer ones. Longer paths have more "memory" of what may constitute a good solution. In order to achieve this, once the maximum number of vertices was reached a population control parameter was multiplied by the values of the extension and repetition fitnesses. The population control parameter is made up of two parts: the *vertex effect*, used on paths with less vertices so that they are more likely to be effected by the population control parameter, and the *entities effect*, used to change the population control parameter as the number of entities in the tube changes. The vertex effect is calculated by:

$$\alpha + (1 - \alpha) * (\text{number of vertices in path} / \text{largest number of vertices in any path}) \quad (1)$$

such that α is configurable. The entities effect is:

$$(\text{max entities} - \text{actual entities in the tube}) / (\text{max entities} - \text{min entities}) \quad (2)$$

The population control parameter is then calculated using the vertex effect and entities effect with:

$$\text{Entities Effect} + (1 - \text{Entities Effect}) * \text{Vertex Effect} . \quad (3)$$

Using a population control parameter with an $\alpha = 0.6$, maximum vertices of 10,000, and minimum vertices of 6000, the dark blue line (population control parameter) in Fig. 6 shows the number of entities added over time. In order to show that the population control parameter also has the effect of improving the quality of the search, Fig. 6 also shows the length of the longest path, in terms of number of vertices times 100, for both the use of just a simple maximum (in light red) and when using the population control parameter (in light blue).

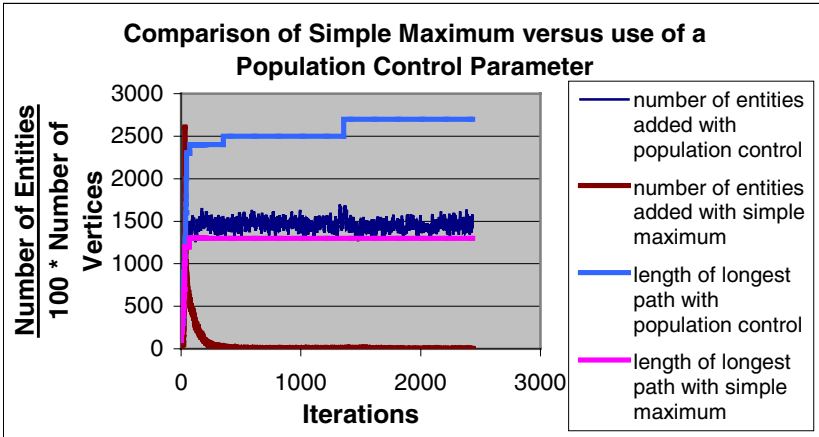


Fig. 6. Comparison of use of a simple maximum versus a population control parameter in terms of both the number of entities added over time and the length of the longest path

Under these conditions, random graphs under 10 vertices can be run with high reliability on a single processor in a matter of hours. The nature of the approach in this paper is instantly scalable to a cluster of processors. Experiments under way may test whether running on a cluster of p processors, *Edna* is really able to handle random graphs of about $10 * p$ vertices, the theoretical maximum.

4 Summary and Conclusions

The results of this paper provide a preliminary estimation of the improved effectiveness and reliability of evolutionary computations *in vitro* that DNA-like genomic representations and environmentally dependent online fitness functions may bring to evolutionary computation. DNA-like computation brings in advantages that biological molecules (DNA, RNA and the like) have gained in the course of millions of years of evolution [11], [7]. First, their operation is inherently parallel and distributable to any number of processors, with the consequent computational advantages. Further, their computational mode is asynchronous and includes massive communications over

noisy media, load balancing, and decentralized control. Second, it is equally clear that the savings in cost and perhaps even time, at least in the range of feasibility of small clusters of conventional sequential computers, is enormous. The equivalent biochemical protocols *in silico* can solve the same problems with a few hundred virtual molecules while requiring trillions of molecules in wet test tubes. Virtual DNA thus inherits the customary efficiency, reliability, and control now standard in electronic computing, hitherto only dreamed of in wet tube computations.

On the other hand, it is also interesting to contemplate the potential to scale these algorithms up to very large graphs when conducting these experiments, either in a real or in virtual test tubes. Biomolecules seem unbeatable by electronics in their ability to pack enormous amounts of information in tiny regions of space and to perform their computations with very high thermodynamical efficiency [13]. This paper also suggests that this efficiency can be brought to evolutionary algorithms *in silico* as well using the DNA-inspired architecture *Edna* used herein.

References

1. Adleman, L.M.: Molecular Computation of Solutions to Combinatorial Problems. In: Science, Vol. 266. (1994) 1021-1024. <http://citeseer.nj.nec.com/adleman94molecular.html>
2. Arita, M., Suyama, A., Hagiya, M.: A heuristic approach for Hamiltonian Path Problem with molecules. In: Proceedings of the Second Annual Genetic Programming Conference (GP-97), Morgan Kaufmann Publishers (1997) 457-461
3. Condon, A., Rozenburg, G. (eds.): DNA Computing (Revised Papers). In: Proc. of the 6th International Workshop on DNA-based Computers. Leiden University, The Netherlands (2000). Springer-Verlag Lecture Notes in Computer Science **2054**
4. Deaton, R., Murphy, R., Rose, J., Garzon, M., Franceschetti, D., Stevens Jr., S.E.: Good Encodings for DNA Solution to Combinatorial Problems. In Proc. IEEE Conference on Evolutionary Computation, IEEE/Computer Society Press. (1997) 267-271
5. Garzon, M., Blain, D., Bobba, K., Neel, A., West, M.: Self-Assembly of DNA-like structures *In Silico*. In Journal of Genetic Programming and Evolvable Machines **4:2** (2003), in press
6. M. Garzon, P. Neathery, R. Deaton, R.C. Murphy, D.R. Franceschetti, S.E. Stevens, Jr.. A New Metric for DNA Computing. In: J.R. Koza, K. Deb, M. Dorigo, D.B. Fogel, M. Garzon, H. Iba, R.L. Riolo (eds.): Proc. 2nd Annual Genetic Programming Conference, San Mateo, CA: Morgan Kaufmann (1997) 472-478
7. Garzon, M., Oehmen, C.: Biomolecular Computation on Virtual Test Tubes, In: Proc. 7th Int. Meeting on DNA Based Computers, Springer-Verlag Lecture Notes in Computer Science **2340** (2001) 117-128
8. Lee, J., Shin, S., Augh, S.J., Park, T.H., Zhang, B.: Temperature Gradient-Based DNA Computing for Graph Problems with Weighted Edges. In: Hagiya, M. and Ohuchi, A. (eds): Proceedings of the 8th Int. Meeting on DNA Based Computers (DNA8), Hokkaido University, Springer-Verlag Lecture Notes in Computer Science **2568** (2002) 73-84
9. Lipton, R.: DNA Solutions of Hard Computational Problems. *Science* 268 (1995) 542-544.

10. Morimoto, N., Masanori, A., Suyama, A.: Solid Phase Solution to the Hamiltonian Path Problem. In: DNA Based Computers III, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, Vol. **48** (1999) 193–206
11. Sigmund, K: Games of Life. Oxford University Press (1993) 145
12. Spencer, J.: Ten Lectures on the Probabilistic Method. In: CMBS **52**, Society for Industrial and Applied Mathematics, Philadelphia (1987) 17–28
13. Wetmur, J.G.: Physical Chemistry of Nucleic Acid Hybridization. In: Rubin, H. and Wood, D.H. (eds.): Proc. DNA-Based Computers III, University of Pennsylvania, June 1997. DIMACS series in Discrete Mathematics and Theoretical Computer Science **48** (1999) 1–23
14. Wood, D.H., Chen, J., Lemieux, B., Cedeno, W.: A design for DNA computation of the OneMax problem. In: Garzon, M., Conrad, M. (eds.): Soft Computing in Biomolecules. Vol. 5:1. Springer-Verlag, Berlin Heidelberg New York (2001) 19–24

String Binding-Blocking Automata^{*}

M. Sakthi Balan

Department of Computer Science and Engineering,
Indian Institute of Technology, Madras
Chennai – 600036, India
`sakthi@cs.iitm.ernet.in`

In a similar way to DNA hybridization, antibodies which specifically recognize peptide sequences can be used for calculation [3,4]. In [4] the concept of peptide computing via peptide-antibody interaction is introduced and an algorithm to solve the satisfiability problem is given. In [3], (1) it is proved that peptide computing is computationally complete and (2) a method to solve two well-known NP-complete problems namely Hamiltonian path problem and exact cover by 3-set problem (a variation of set cover problem) using the interactions between peptides and antibodies is given.

In our earlier paper [1], we proposed a theoretical model called as binding-blocking automata (BBA) for computing with peptide-antibody interactions. In [1] we define two types of transitions - leftmost(l) and locally leftmost(ll) of BBA and prove that the acceptance power of multihead finite automata is sandwiched between the acceptance power of BBA in l and ll transitions. In this work we define a variant of binding-blocking automata called as string binding-blocking automata and analyze the acceptance power of the new model.

The model of binding-blocking automaton can be informally said as a finite state automaton (reading a string of symbols at a time) with (1) blocking and unblocking functions and (2) priority relation in reading of symbols. Blocking and unblocking facilitates skipping¹ some symbols at some instant and reading it when it is necessary.

In the sequel we state some results from [1,2] - (1) for every *BBA* there exists an equivalent *BBA* without priority, (2) for every language accepted by *BBA* with l transition, there exists *BBA* with ll transitions accepting the same language, (3) for every language accepted by *BBA* with l transition there is an equivalent multi-head finite automata which accepts the same language and (4) for every language L accepted by a multi-head finite automaton there is a language L' accepted by *BBA* such that L can be written in the form $h^{-1}(L')$ where h is a homomorphism from L to L' .

The basic model of the string binding-blocking automaton is very similar to a *BBA* but for the blocking and unblocking. Some string of symbols (starting from the head's position) can be blocked from being read by the head. So only those symbols which are not already read and not blocked can be read by the head. The finite control of the automaton is divided into three sets of states namely blocking states, unblocking states and general reading states. A read symbol can not be read again, but a blocked symbol can be unblocked and read.

^{*} Financial support from Infosys Technologies Limited, India is acknowledged

¹ running through the symbols without reading

Let us suppose the input string is y . At any time the system can be in any one of the three states - reading state, blocking state or unblocking state. In reading state the system can read a string of symbols (say l symbols) at a time and move its head l positions to the right. In the blocking state q , the system blocks a string of symbols as specified by the blocking function (say $x \in L$ where $L \in \beta_b(q)$, $x \in \text{Sub}(y)$ ²) starting from the position of the head. The string x satisfies the maximal property i.e., there exists no $z \in L$ such that $x \in \text{Pre}(z)$ ³ and $z \in \text{Sub}(y)$. When the system is in the unblocking state q the recently blocked string $x \in \text{Sub}(y)$ and $x \in L$ where $L \in \beta_{ub}(q)$ is unblocked. We note that the head can only read symbols which are neither read nor blocked. The symbols which are read by the head are called *marked* symbols, which are blocked are called as *blocked* symbols.

A string binding-blocking automaton with D -transition is denoted by $strbba_D$ and the language accepted by the above automaton is denoted by $StrBBA_D$. If the blocking languages are finite languages then the above system is represented by $strbba(\text{Fin})$.

We show that $strbba_l$ system is more powerful than bba system working in l transition by showing that $L = \{a^n ba^n \mid n \geq 1\}$ is accepted by $strbba_l$ but not by any bba working in l transition.

The above language is accepted by $strbba_{ll}$.

The language $L = \{a^{2n+1}(aca)^{2n+1} \mid n \geq 1\}$ shows that $strbba_l$ system is more powerful than bba system working in ll transition.

We also prove the following results,

1. For any bba_{ll} we can construct an equivalent $strbba_{ll}$.
2. For every $L \in StrBBA_l$ there exists a random-context grammar RC with Context-free rules such that $L(RC) = L$.
3. For every $strbba_D$, $\mathcal{P}$ there is an equivalent $strbba_D$, $\mathcal{Q}$ such that there is only one accepting state and there is no transition from the accepting state.

Hence by above examples and results we have $\mathcal{L}(bba_{ll}) \subset \mathcal{L}(strbba_{ll})$ and $\mathcal{L}(bba_l) \neq \mathcal{L}(strbba_l)$

References

1. M.Sakthi Balan and Kamala Krithivasan. Blocking-binding automata. poster presentation in Eighth International Conference on DNA based Computers, 2002.
2. M.Sakthi Balan and Kamala Krithivasan. Normal-forms of binding-blocking automata. poster presentation in Unconventional Models of Computing, 2002.
3. M.Sakthi Balan, Kamala Krithivasan, and Y.Sivasubramanyam. Peptide computing – universality and complexity. In Natasha Jonoska and Nadrian Seeman, editors, *Proceedings of Seventh International Conference on DNA Based Computers – DNA7, LNCS*, volume 2340, pages 290–299, 2002.
4. Hubert Hug and Rainer Schuler. Strategies for the development of a peptide computer. *Bioinformatics*, 17:364–368, 2001.

² $\text{Sub}(y)$ is the set of all sub-strings of y

³ $\text{Pre}(z)$ is the set of all prefixes of z

On Setting the Parameters of QEA for Practical Applications: Some Guidelines Based on Empirical Evidence

Kuk-Hyun Han and Jong-Hwan Kim

Department of Electrical Engineering and Computer Science,
Korea Advanced Institute of Science and Technology (KAIST),
Guseong-dong, Yuseong-gu, Daejeon, 305-701, Republic of Korea
{khhan, johkim}@rit.kaist.ac.kr

Abstract. In this paper, some guidelines for setting the parameters of quantum-inspired evolutionary algorithm (QEA) are presented. Although the performance of QEA is excellent, there is relatively little or no research on the effects of different settings for its parameters. The guidelines are drawn up based on extensive experiments.

1 Introduction

Quantum-inspired evolutionary algorithm (QEA) recently proposed in [1] can treat the balance between exploration and exploitation more easily when compared to conventional GAs (CGAs). Also, QEA can explore the search space with a small number of individuals and exploit the global solution in the search space within a short span of time. QEA is based on the concept and principles of quantum computing, such as a quantum bit and superposition of states. However, QEA is not a quantum algorithm, but a novel evolutionary algorithm.

In [1], the structure of QEA and its characteristics were formulated and analyzed, respectively. According to [1], the results (on the knapsack problem) of QEA with population size of 1 were better than those of CGA with population size of 50. In [2], a QEA-based disk allocation method (QDM) was proposed. According to [2], the average query response times of QDM are equal to or less than those of DAGA (disk allocation methods using GA), and the convergence of QDM is 3.2-11.3 times faster than that of DAGA. In [3], a QEA-based face verification was proposed. In this paper, some guidelines for setting the related parameters are presented to maximize the performance of QEA.

2 Some Guidelines for Setting the Parameters of QEA

In this section, some guidelines for setting the parameters of QEA are investigated. These guidelines are drawn up based on empirical results.

The initial values of Q-bit are set to $\left(\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}\right)$ for the uniform distribution of 0 or 1. To improve the performance, we can think of the two-phase mechanism

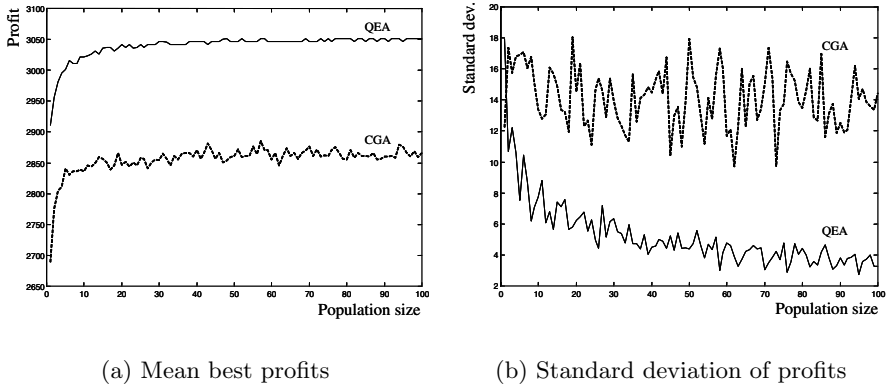


Fig. 1. Effects of changing the population sizes of QEA and CGA for the knapsack problem with 500 items. The global migration period and the local migration period were 100 and 1, respectively. The results were averaged from 30 runs.

for initial conditions. In the first phase, some promising initial values can be searched. If they are used in the second phase, the performance of QEA will increase.

From the empirical results, Table I in [1] for the rotation gate can be simplified as $[0 * p * n * 0]^T$, where p is a positive number and n is a negative number, for various optimization problems. The magnitude of p or n has an effect on the speed of convergence, but if it is too big, the solutions may diverge or converge prematurely to a local optimum. The values from 0.001π to 0.05π are recommended for the magnitude, although they depend on the problems. The sign determines the direction of convergence.

From the results of Figure 1, the values ranging from 10 to 30 are recommended to be used as the population size. However, if more robustness is needed, the population size should be increased (see Figure 1-(b)). The global migration period is recommended to be set to the values ranging from 100 to 150, and the local migration period to 1.

These guidelines can help researchers and engineers who want to use QEA for their application problems.

References

1. Han, K.-H., Kim, J.-H.: Quantum-inspired Evolutionary Algorithm for a Class of Combinatorial Optimization. *IEEE Trans. Evol. Comput.* **6** (2002) 580–593
2. Kim, K.-H., Hwang, J.-Y., Han, K.-H., Kim, J.-H., Park, K.-H.: A Quantum-inspired Evolutionary Computing Algorithm for Disk Allocation Method. *IEICE Trans. Inf. & Syst.*, **E86-D** (2003) 645–649
3. Jang, J.-S., Han, K.-H., Kim, J.-H.: Quantum-inspired Evolutionary Algorithm-based Face Verification. *Proc. Genet. & Evol. Comput. Conf.* (2003)

Evolutionary Two-Dimensional DNA Sequence Alignment

Edgar E. Vallejo¹ and Fernando Ramos²

¹ Computer Science Dept., Tecnológico de Monterrey, Campus Estado de México
Carretera Lago de Guadalupe Km 3.5 Col. Margarita Maza de Juárez, 52926
Atizapán de Zaragoza, Eestado de México, México

`vallejo@itesm.mx`

² Computer Science Dept., Tecnológico de Monterrey, Campus Cuernavaca
Ave. Paseo de la Reforma 182 Col. Lomas de Cuernavaca, 62589

Cuernavaca, Morelos, México

`fernando.ramos@itesm.mx`

Abstract. This article presents a model for DNA sequence alignment. In our model, a finite state automaton writes two-dimensional maps of nucleotide sequences. An evolutionary method for sequence alignment from this representation is proposed. We use HIV as the working example. Experimental results indicate that structural similarities produced by two-dimensional representation of sequences allow us to perform pairwise and multiple sequence alignment efficiently using genetic algorithms.

1 Introduction

The area of bioinformatics is concerned with the analysis of molecular sequences to determine the structure and function of biological molecules [2]. Fundamental questions about functional, structural and evolutionary properties of molecular sequence can be answered using sequence alignment.

Research in sequence alignment has focused for many years on the design and analysis of efficient algorithms that operate on linear character representation of nucleotide and protein sequences. The intractability of multiple sequence alignment algorithms evidences limitations for the analysis of molecular sequences from this representation. Similarly, due to the extension of typical genomes, this representation is also inconvenient from the human perception perspective.

2 The Model

In our model, a finite state automaton writes a two-dimensional map of DNA sequences [3]. The proposed alignment method is based on the overlapping of a collection of these maps. We overlap a two-dimensional map over another to discover coincidences in character patterns. Sequence alignment consists of the sliding of maps over a reference plane in order to search for the optimum overlapping. We use genetic algorithms to evolve the cartesian positions of a collection of maps that maximize coincidences in character patterns.

3 Experiments and Results

We performed several runs using HIV nucleotide sequences. Figure 1 shows the results of a typical run. We performed comparisons using conventional sequence alignment methods that operate on linear sequences. We found that our method yields similar results to those produced by the SIM local alignment algorithm.

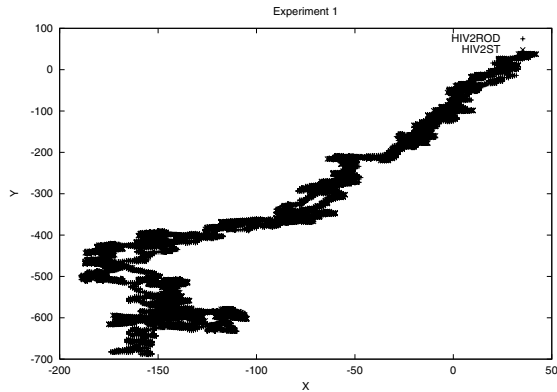


Fig. 1. Results. Pairwise DNA sequence alignment

4 Conclusions and Future Work

We present a sequence alignment method based on two-dimensional representation of DNA sequences and genetic algorithms. An immediate extension of this work is the consideration of protein sequences and the construction of phylogenies from two-dimensional alignment scores. Finally, a more detailed comparative analysis using evolutionary [1] and conventional [2] alignment methods could elucidate the significance of evolutionary two-dimensional sequence alignment.

References

1. Fogel, G. E., Corne, D. W. (eds.) 2003. *Evolutionary Computation in Bioinformatics*. Morgan Kaufmann Publishers.
2. Mount, D. 2000. *Bioinformatics. Sequence and Genome Analysis*. Cold Spring Harbor, NY: Cold Spring Harbor Laboratory Press.
3. Vallejo, E. E., Ramos, F. 2002. Evolving Finite Automata with Two-dimensional Output for Biosequence Recognition and Visualization In W. B. Langton, E. Cantú-Paz, K. Mathias, R. Roy, R. Poli, K. Balakrishnan, V. Honovar, G. Rudolph, J. Wegener, L. Bull, M. A. Potter, A. C. Schultz, J. E. Miller, E. Burke, N. Jonoska (eds.) *Proceedings of the Genetic and Evolutionary Computation Conference GECCO 2002*. Morgan Kaufmann Publishers.

Active Control of Thermoacoustic Instability in a Model Combustor with Neuromorphic Evolvable Hardware

John C. Gallagher and Saranyan Vighram

Department of Computer Science and Engineering
Wright State University, Dayton, OH, 45435-0001
{jgallagh,svigraha}@cs.wright.edu

Abstract. Continuous Time Recurrent Neural Networks (CTRNNs) have previously been proposed as an enabling paradigm for evolving analog electrical circuits to serve as controllers for physical devices [6]. Currently underway is the design of a CTRNN-EH VLSI chips that combines an evolutionary algorithm and a reconfigurable analog CTRNN into a single hardware device capable of learning control laws of physical devices. One potential application of this proposed device is the control and suppression of potentially damaging thermoacoustic instability in gas turbine engines. In this paper, we will present experimental evidence demonstrating the feasibility of CTRNN-EH chips for this application. We will compare our controller efficacy with that of a more traditional Linear Quadratic Regulator (LQR), showing that our evolved controllers consistently perform better and possess better generalization abilities. We will conclude with a discussion of the implications of our findings and plans for future work.

1 Introduction

An area of particular interest in modern combustion research is the study of lean premixed (LP) fuel combustors that operate at low fuel-to-air ratios. LP fuels have the advantage of allowing for more complete combustion of fuel products, which decreases harmful combustor emissions that contribute to the formation of acid rain and smog. Use of LP fuels however, contributes to flame instability, which causes potentially damaging acoustic oscillations that can shorten the operational life of the engine. In severe cases, flame-outs or major engine component failure are also possible. One potential solution to the thermoacoustic instability problem is to introduce active control devices capable of sensing and suppressing dangerous oscillations by introducing appropriate control efforts.

Because combustion systems can be so difficult to model and analyze, self-configuring evolvable hardware (EH) control devices are likely to be of enormous value in controlling real engines that might defy more traditional techniques. Further, an EH controller would be able to adapt and change online, continuously optimizing its control over the service life of a particular combustor. This paper

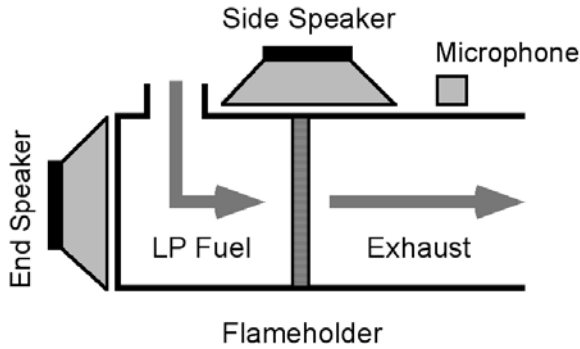


Fig. 1. Schematic of a Test Combustor

will discuss our efforts to control the model combustor presented in [10] [11] with a simulated evolvable hardware device. We will begin with brief summaries of the simulated combustor and our CTRNN-EH device. Following, we will discuss our evolved CTRNN-EH control devices and how their performance compares to a traditional LQR controller. Finally, we will discuss the implications of our results and discuss future work in which we will apply CTRNN-EH to the control of real engines.

2 The Model Combustor

Figure 1 shows a schematic of a simple combustor. Premixed fuel and air is introduced at the closed end and the flame is anchored on a perforated disk mounted inside the chamber a short distance from the closed end (the flameholder). Combustion products are forced out the open end. Thermoacoustic instability can occur due to positive feedback between combustion dynamics of the flame and acoustic properties of the combustion chamber. Qualitatively speaking, flame dynamics are affected by mechanical vibration of the combustion chamber and mechanical vibration of the combustion chamber is affected by heat release/flame dynamics. When these two phenomena reinforce one another, it is possible for the vibrations of the combustion chamber to grow to unsafe levels. Figure 2 shows the engine pressure with respect to time for the first 0.04 seconds of uncontrolled operation of an unstable engine. Note that maximum pressure amplitude is growing exponentially and would quickly grow to unsafe levels. In the model engine, a microphone is mounted on the chamber to monitor the frequency and magnitude of pressure oscillations. A loudspeaker effector used to introduce additional vibrations is mounted either at the closed end of the chamber or along its side. Figure 1 shows both speaker mounting options, though for any experiment we discuss here, only one would be used at a time.

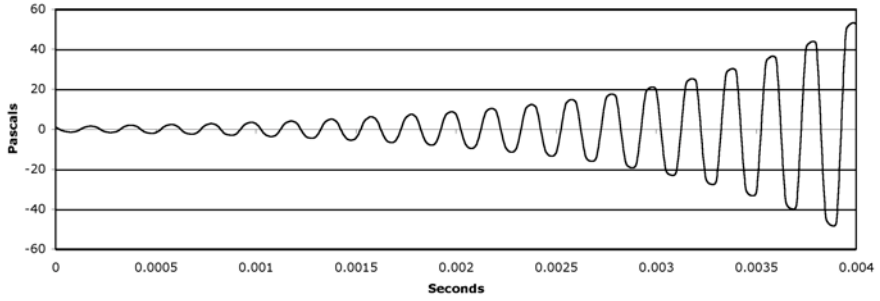


Fig. 2. Time Series Response of the Uncontrolled EM1 Combustor

A full development of the simulation state equations, which have been verified against a real propane burning combustor, is given in [10]. Using these state equations, we implemented C language simulations of four combustor configurations. All four simulations assumed a specific heat ratio of 1.4, an atmospheric pressure of 1 atmosphere, an ambient temperature of 350K, a fuel/air mixture of 0.8, a speed of sound of 350 m/s, and a burn rate of 0.4 m/s. The four engine configurations, designated SM1, SM2, EM1, and EM2, were drawn from [10] and represent speaker side-mount configurations resonant at 542 Hz and 708 Hz and end-mount configurations resonant at 357 Hz and 714 Hz respectively.

3 CTRNN-EH

CTRNN-EH devices combine a reconfigurable analog continuous time recurrent neural network (CTRNN) and Star Compact Genetic Algorithm (*CGA) into a single hardware device.

CTRNNs are networks of Hopfield continuous model neurons [2][5][12] with unconstrained connection weight matrices. Each neuron's activity can be expressed by an equation of the following form:

$$\tau_i \frac{dy_i}{dt} = -y_i + \sum_{j=1}^N w_{ji} \sigma(y_j + \theta_j) + s_i I_i(t) \quad (1)$$

where y_i is the state of neuron i , τ_i is the time constant of neuron i , w_{ji} is the connection weight from neuron j to neuron i , $\sigma(x)$ is the standard logistic function, θ_j is the bias of neuron j , s_i is the sensor input weight of neuron i , and $I_i(t)$ is the sensory input to neuron i at time t .

CTRNNs differ from Hopfield networks in that they have no restrictions on their interneuron weights and are universal dynamics approximators [5]. Due to their status as universal dynamics approximators, we can be reasonably assured

that any control law of interest is achievable using collections of CTRNN neurons. Further, a number of analog and mixed analog-digital implementations are known [13] [14] [15] and available for use.

*CGAs are any of a family of tournament-based modified Compact Genetic Algorithms [9] [7] selected for this application because of the ease in which they may be implemented using common VLSI techniques [1] [8]. The *CGAs require far less memory than other EAs because they represent populations as compact probability vectors rather than as sets of actual bit strings. In this work, we employed the mCGA variation similar to that documented in [9]. The algorithm can be stated as shown in figure 3.

Figure 4 shows a schematic representation of our CTRNN-EH device used in intrinsic mode to learn the control law of an attached device. In this case, the user would provide a hardware or software system that produces a scalar measure (performance score) of the controlled devices effectiveness based upon inputs from some associated instrumentation. This is represented in the rightmost block of Figure 4. The CTRNN-EH device, represented by the leftmost block in the figure, would receive fitness scores from the evaluator and sensory inputs from the controlled device. The CGA engine would evolve CTRNN configurations that monitor device sensors and supply effector efforts that maximized the controlled devices performance.

4 CTRNN-EH Control Experiments

In the experiments reported in this paper, we employed a simulated CTRNN-EH device that contained a five neuron, fully-connected CTRNN as the analog neuromorphic component and a mCGA [8] as the EA component. The CTRNN was interfaced to the combustor as shown in Figure 5. Each neuron received the raw microphone value as input. The outputs of two CTRNN neurons controlled the amplitude and frequency of a voltage controlled oscillator that itself drove the loudspeaker (I.E. The CTRNN had control over the amplitude and frequency of the loudspeaker effector). Speaker excitations could range from 0 to 10 mA in amplitude and 0 to 150 Hz in frequency. The error function (performance evaluator) was the sum of amplitudes of all pressure peaks observed in a period of one second. This error function roughly approximates and produces the same relative rankings that would be produced by using simple hardware to integrate the area under the microphone signal in the time domain. mCGA parameters were chosen as follows: simulated population size of 1023, a maximum tournament count of 100,000, and a bitwise mutation rate of 0.05. Forty CTRNN parameters (five time constants, five biases, five sensor weights, and twenty-five intra-network weights) were encoded as eight bit values resulting in a 320 bit genome. All experiments were run on a 16 node SGI Beowulf cluster.

We ran 100 evolutionary trials for each of the four engine configurations. On average, 589, 564, 529, and 501 tournaments were required to evolve effective oscillation suppression for SM1, SM2, EM1, and EM2 respectively. Each of the the resulting four hundred evolved champions was tested for control efficacy across all

1. Initialize probability vector
for i := 1 to L do p[i] := 0.5
2. Generate two individuals from the vector
a := generate(p);
b := generate(p);
3. Let them compete
winner, loser := evaluate(a, b)
4. Update the probability vector toward the winner
for i := 1 to L do
if winner[i] <> loser[i] then
if winner[i] = 1 then
p[i] := p[i] + (1 / N)
else
p[i] := p[i] - (1 / N)
5. Mutate champ and evaluate
if winner = a then
c := mutate(a);
evaluate(c);
if fitness(c) > fitness(a) then
a := c;
else
c := mutate(b);
evaluate(c);
if fitness(c) > fitness(b) then
b := c;
6. Generate one individual from the vector
if winner = a then
b := generate(p);
else
a := generate(p);
7. Check if probability vector has converged
for i := 1 to L do
if p[i] > 0 and p[i] < 1 then goto step 3
8. P represents the final solution

Fig. 3. Pseudo-code for mCGA

four modeled engine configurations (SM1, SM2, EM1, and EM2). All were effective in suppressing vibrations under the conditions for which they were evolved. In addition, all were capable of effectively suppressing vibrations in the engine configurations for which they were not evolved. Typical engine noise suppression

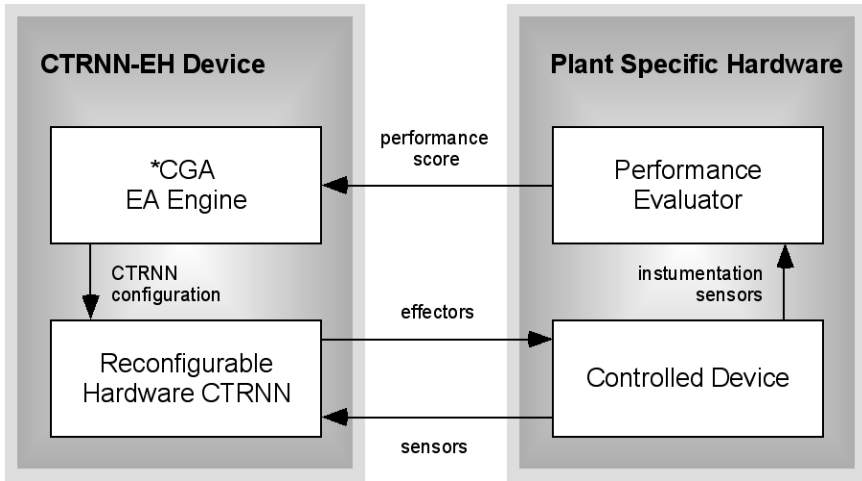


Fig. 4. Schematic of CTRNN-EH Controller

results for both a side mounted CTRNN-EH controller and a Linear Quadratic Regulator (LQR) are shown in Figure 6. Tables 1, 2 3, and 4 summarize the average settling times (the time the controller requires to stabilize the engine) across all experiments. Note that in Figure 6, our evolved controller settles to stability significantly faster than the LQR. The LQR controllers presented in [10] and [11] had settling times of about 40 mS and 20 mS for the end-mounted and side-mounted configurations respectively. Note that our evolved CTRNNs compare very well to LQR devices. On average, they evolved to produce settling times of better than 20 ms. The very best CTRNN controllers settle in as few as 8 ms. Further, the presented LQR controllers failed to function properly when used in a mounting configuration for which they were not designed, while all of our evolved controllers appear capable of controlling oscillations irregardless of where the effector is mounted. Both of these results suggest that our evolved controllers may be both faster (in terms of settling time) and more flexible (in terms of effector placement) than the given LQR devices. Presuming that we implemented only the analog CTRNN portion of the CTRNN-EH device, this improved capability would be achieved without a significant increase in the amount of analog hardware required.

In other, related work, we have observed that mCGA seems better able to evolve CTRNN controllers than the population based Simple Genetic Algorithm (sGA) that it emulates [7]. This effect was observed in experiments reported here as well. We evolved 100 CTRNN controllers for the each engine configuration using a tournament based simple GA with uniform crossover, a bitwise mutation rate of 0.05, and a population size of 1023. On average, the sGA required 5000 tournaments to evolve effective control. The difference between the number of generations required for sGA and mCGA is statistically significant. Table 5 shows

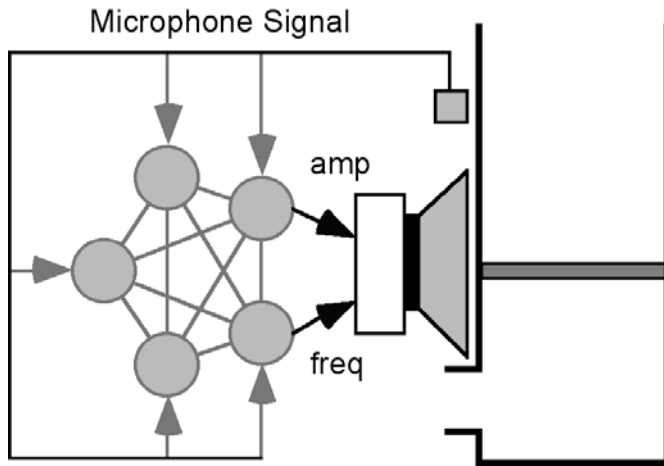


Fig. 5. CTRNN to Combustor Interface

the average settling times of sGA and mCGA controllers evolved in the SM1 configuration. These results are representative of those observed under other evolutionary conditions.

5 Conclusions and Discussion

In this paper, we demonstrated that, against an experimentally verified combustor model, CTRNN-EH evolvable hardware controllers are consistently capable of evolving highly effective active oscillation suppression abilities that generalized to control different engine configurations as well. Further, we demonstrated that we could surpass the performance of a benchmark LQR device reported in the literature as a means of solving the same problem. These results are in themselves significant. More significant, however, are the implications of those results.

First, the LQR devices referenced were developed based upon detailed knowledge of the system to be controlled. A model needed to be constructed and validated before controllers could be constructed. Even in the case of the relatively simple combustion device that was modeled and simulated, this was a significant effort. Though it may be the case that improved control can be had by using other model-based methods, any such improvements would be purchased at the cost of significant additional work. Further, it is not clear that one would be able to construct appropriately detailed mathematical models of more realistic combustor systems with more realistic engine actuation methods. Thus, it is not clear if model-based control methods could be applied to more realistic engines. Our CTRNN-EH controllers were developed without specific knowledge of the plant to be controlled. A *CGA evolved a very general dynamics approximator

Table 1. Controllers Evolved in SM1 Configuration

Statistic	Tested in EM1	Tested in EM2	Tested in SM1	Tested in SM2
Average	12.51 ms	11.80 ms	11.141 ms	11.78 ms
Stdev	5.38 ms	5.22 ms	5.21 ms	1.08 ms

Table 2. Controllers Evolved in EM1 Configuration

Statistic	Tested in EM1	Tested in EM2	Tested in SM1	Tested in SM2
Average	14.68 ms	13.84 ms	13.05 ms	12.20 ms
Stdev	6.37 ms	6.23 ms	5.97 ms	1.14 ms

Table 3. Controllers Evolved in SM2 Configuration

Statistic	Tested in EM1	Tested in EM2	Tested in SM1	Tested in SM2
Average	21.93 ms	21.41 ms	20.06 ms	13.03 ms
Stdev	3.74 ms	3.80 ms	3.92 ms	0.67 ms

Table 4. Controllers Evolved in EM2 Configuration

Statistic	Tested in EM1	Tested in EM2	Tested in SM1	Tested in SM2
Average	13.22 ms	12.53 ms	11.85 ms	11.91 ms
Stdev	5.79 ms	5.58 ms	5.58 ms	1.07 ms

Table 5. Controllers Evolved with sGA in SM1 Configuration

Statistic	Tested in EM1	Tested in EM2	Tested in SM1	Tested in SM2
Average	14.72 ms	17.31 ms	13.65 ms	14.03 ms
Stdev	4.92 ms	5.61 ms	5.16 ms	3.23 ms

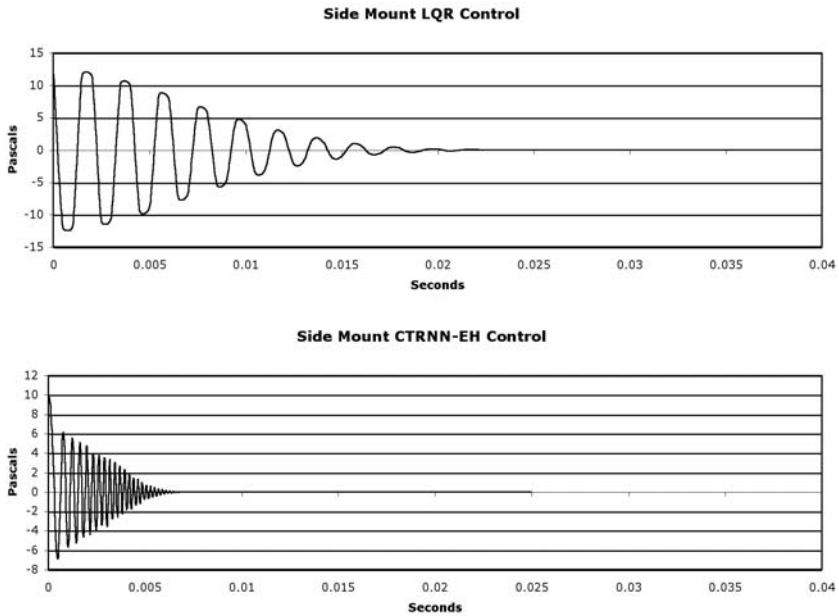


Fig. 6. Typical LQR Response vs. CTRNN-EH Response

to stabilize the engine. Such a technique could be applied without modification to any engine and/or combustor system – with any sort of engine effectors. Naturally, one might argue that the evolved control devices would be too difficult to understand and verify, rendering them less attractive for use in important control applications. However, especially in cases where there are few sensor inputs, we have already developed analysis techniques that should be able to construct detailed explanations of CTRNN operation with respect to specific control problems [3] [4]. The engine controllers we presented in this paper are currently undergoing analysis using these dynamical systems methods and we expect to construct explanations of their operation in the near future.

Second, although our initial studies have been of necessity in simulation, we have made large strides in constructing hardware prototypes on our way to a complete, self-contained VLSI implementation. We have already constructed and verified a reconfigurable analog CTRNN engine using off-the-shelf components [6] and have implemented the mCGA completely in hardware with FPGAs [7]. Our early experiments suggest that our hardware behaves as predicted in simulation. We are currently integrating these prototypes to create the first, fully hardware CTRNN-EH device. This first integrated prototype will be used to evolve oscillation suppression on a physical test combustor patterned after that

modeled in [10]. Our positive results in simulation make moving to this next phase possible.

Third, earlier in this paper, we reported that mCGA evolves better solutions than does a similar simple GA. This phenomenon is not unique to the engine control problem, in fact, we have observed it in evolving CTRNN based controllers for other physical processes [7]. Understanding why this is the case will likely lead to important information about the nature of CTRNN search spaces, the mechanics of the *CGAs, or both. This study is also currently underway.

Evolvable hardware has the potential to produce computational and control devices with unprecedented abilities to automatically configure to specific requirements, to automatically heal in the face of damage, and even to exploit methods beyond what is currently considered state of the art. The results in this paper argue strongly for the feasibility of EH methods to address a difficult problem of practical import. They also point the way toward further study and development of general techniques of potential use to the EH community.

Acknowledgements. This work was supported by Wright State University and The Ohio Board of Regents through the Research Challenge Grant Program.

References

1. Aporntewan, C. and Chongstitvatana. (2001). A hardware implementation of the compact genetic algorithm. in *The Proceedings of the 2001 IEEE Congress on Evolutionary Computation*
2. Beer, R.D. (1995). On the dynamics of small continuous-time recurrent neural networks. in *Adaptive Behavior* 3(4):469–509.
3. Beer, R.D., Chiel, H.J. and Gallagher, J.C. (1999). Evolution and analysis of model CPGs for walking II. general principles and individual variability. in *J. Computational Neuroscience* 7(2):119–147.
4. Chiel, H.J., Beer, R.D. and Gallagher, J.C. (1999). Evolution and analysis of model CPGs for walking I. dynamical modules. in *J. Computational Neuroscience* 7(2):99–118.
5. Funahashi, K & Nakamura, Y. (1993), Approximation of dynamical systems by continuous time recurrent neural networks, in *Neural Networks* 6:801–806
6. Gallagher, J.C. & Fiore, J.M., (2000). Continuous time recurrent neural networks: a paradigm for evolvable analog controller circuits, in *The Proceedings of the 51st National Aerospace and Electronics Conference*
7. Gallagher, J.C., Vighram, S., Kramer, G. (2002). A family of compact genetic algorithms for intrinsic evolvable hardware. Submitted to *IEEE Transactions on Evolutionary Computation*
8. Gallagher, J.C. & Vighram, S. (2002). A modified compact genetic algorithm for the intrinsic evolution of continuous time recurrent neural networks. in *The Proceedings of the 2002 Genetic and Evolutionary Computation Conference*. Morgan-Kaufmann.
9. Harik, G., Lobo, F., & Goldberg, D.E. (1999). The compact genetic algorithm. In *IEEE Transactions on Evolutionary Computation*. Vol 3, No. 4. pp. 287–297

10. Hathout, J.P., Annaswamy, A.M., Fleifil, M. and Ghoniem, A.F. (1998). Model-based active control design for thermoacoustic instability. in *Combustion Science and Technology*, 132: 99–138
11. Hathout, J.P., Fleifil, M., Rumsey, J.W., Annaswamy, A.M., and Ghoniem, A.F. (1997). Model-based analysis and design of active control of thermoacoustic instability. in *IEEE Conference on Control Applications*, Hartford, CT, October 1997.
12. Hopfield, J.J. (1984). Neurons with graded response properties have collective computational properties like those of two-state neurons, in *Proceedings of the National Academy of Sciences* 81:3088–3092
13. Maass, W. and Bishop, C. (1999). *Pulsed Neural Networks*. MIT Press.
14. Mead, C.A., (1989). *Analog VLSI and Neural Systems*, Addison-Wesley, New York
15. Murray, A. and Tarassenko, L. (1994). *Analogue Neural VLSI : A Pulse Stream Approach*. Chapman and Hall, London.

Hardware Evolution of Analog Speed Controllers for a DC Motor

David A. Gwaltney¹ and Michael I. Ferguson²

¹ NASA Marshall Space Flight Center, Huntsville, AL 35812, USA

David.A.Gwaltney@nasa.gov

² Jet Propulsion Laboratory, California Institute of Technology
Pasadena, CA 91109, USA

Michael.I.Ferguson@jpl.nasa.gov

Abstract. Evolvable hardware provides the capability to evolve analog circuits to produce amplifier and filter functions. Conventional analog controller designs employ these same functions. Analog controllers for the control of the shaft speed of a DC motor are evolved on an evolvable hardware platform utilizing a Field Programmable Transistor Array (FPTA). The performance of these evolved controllers is compared to that of a conventional proportional-integral (PI) controller. It is shown that hardware evolution is able to create a compact design that provides good performance, while using considerably less functional electronic components than the conventional design.

1 Introduction

Research on the application of hardware evolution to the design of analog circuits has been conducted extensively by many researchers. Many of these efforts utilize a SPICE simulation of the circuitry, which is acted on by the evolutionary algorithm chosen to evolve the desired functionality. An example of this is the work done by Lohn and Columbano at NASA Ames Research Center to develop a circuit representation technique that can be used to evolve analog circuitry in software simulation[1]. This was used to conduct experiments in evolving filter circuits and amplifiers. A smaller, but rapidly increasing number of researchers have pursued the use of physical circuitry to study evolution of analog circuit designs. The availability of reconfigurable analog devices via commercial or research-oriented sources is enabling this approach to be more widely studied. Custom Field Programmable Transistor Array (FPTA) chips have been used for the evolution of logic and analog circuits. Efforts at the Jet Propulsion Laboratory (JPL) using their FPTA2 chip are documented in [2,3,4]. Another FPTA development effort at Heidelberg University is described in [5]. Some researchers have conducted experiments using commercially available analog programmable devices to evolve amplifier designs, among other functions[6,7].

At the same time, efforts to use evolutionary algorithms to design controllers have also been widely reported. Most of the work is on the evolution of controller designs suitable only for implementation in software. Koza, et al., presented

automatic synthesis of control laws and tuning for a plant with time delay using Genetic programming. This was done in simulation [8]. However, Zebulum, et. al., have evolved analog controllers for a variety of industrially representative dynamic system models[10]. In this work, the evolution was also conducted in a simulated environment.

Hardware evolution can enable the deployment of a self-configurable controller in hardware. Such a controller will be able to adapt to environmental conditions that would otherwise degrade performance, such as temperature varying to extremes or ionizing radiation. Hardware evolution can provide fault-tolerance capability by re-routing internal connections around damaged components or by reuse of degraded components in novel designs. These features, along with the capability to accommodate unanticipated or changing mission requirements, make an evolvable controller attractive for use in a remotely located platform, such as a spacecraft. Hence, this effort focuses on the application of hardware evolution to the *in situ* design of a shaft speed controller for a DC motor. To this end, the Stand-Alone Board-Level Evolvable (SABLE) System[3], developed by researchers at the Jet Propulsion Laboratory, is used as the platform to evolve analog speed controllers for a DC motor.

Motor driven actuators are ubiquitous in the commercial, industrial, military and aerospace environments. A recent trend in aviation and aerospace is the use of power-by-wire technologies. This refers to the use of motor driven actuators, rather than hydraulic actuators for aero-control surfaces[11][12]. Motor driven actuators have been considered for upgrading the thrust vector control of the Space Shuttle main engines [13]. In spacecraft applications, servo-motors can be used for positioning sun-sensors, Attitude and Orbit Control Subsystems (AOCSs), antennas, as well as valves, linear actuators and other closed-loop controllers.

In this age of digital processor-based control, analog controllers are still frequently used at the actuator level in a variety of systems. In the harsh environment of space, electronic components must be rated to survive temperature extremes and exposure to radiation. Very few microcontrollers and digital signal processors are available that are rated for operation in a radiation environment. However, operational amplifiers and discrete components are readily available and are frequently applied. Reconfigurable analog devices provide a small form factor platform on which multiple analog controllers can be implemented. The FPTA2, as part of the SABLE System, is a perfect platform for implementation of multiple controllers, because its sixty-four cells can theoretically provide sixty-four operational amplifiers, or evolved variations of amplifier topologies. Further, its relatively small size and low power requirements provide savings in space and power consumption over the uses of individual operational amplifiers and discrete components[2].

The round-trip communication time between the Earth and a spacecraft at Mars ranges from 10 to 40 minutes. For spacecraft exploring the outer planets the time increases significantly. A spacecraft with self-configuring controllers could work out interim solutions to control system failures in the time it takes

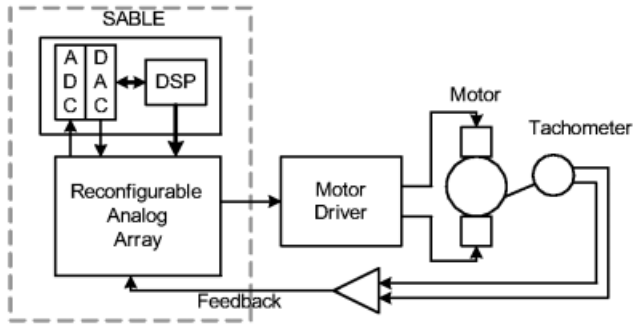


Fig. 1. Configuration of the SABLE System and motor to be controlled

for the spacecraft to alert its handlers on the Earth of a problem. The evolvable nature of the hardware allows a new controller to be created from compromised electronics, or the use of remaining undamaged resources to achieve required system performance. Because the capabilities of a self-configuring controller could greatly increase the probability of mission success in a remote spacecraft, and motor driven actuators are frequently used, the application of hardware evolution to motor controller design is considered a good starting point for the development of a general self-configuring controller architecture.

2 Approach

The JPL developed Stand-Alone Board Level Evolvable (SABLE) System[3] is used for evolving the analog control electronics. This system employs the JPL designed Second Generation, Field Programmable Transistor Array (FPTA2). The FPTA2 contains 64 programmable cells on which an electronic design can be implemented by closing internal switches. The schematic diagram of one cell is given in the Appendix. Each cell has inputs and outputs connected to external pins or the outputs of neighboring cells. More detail on the FPTA2 architecture is found in [2]. A diagram of the experimental setup is shown in Figure 1. The main components of the system are a TI-6701 Digital Signal Processor (DSP), a 100kSa/sec 16-channel DAC and ADC and the FPTA2. There is a 32-bit digital I/O interface connecting the DSP to the FPTA2. The genetic algorithm running on the DSP follows a simple algorithm of download, stimulate the circuit with a control signal, record the response, evaluate the response against the expected. This is repeated for each individual in the population and then crossover, and mutation operators are performed on all but the elite percentage of individuals.

The motor used is a DC servo-motor with a tachometer mounted to the shaft of the motor. The motor driver is configured to accept motor current commands and requires a 17.5 volt power supply with the capability to produce 6 amps of current. A negative 17.5 volt supply with considerably lower current requirements is needed for the circuitry that translates FPTA2 output signals

to the proper range for input to the driver. The tachometer feedback range is roughly $[-4, +4]$ volts which corresponds to a motor shaft speed range of $[-1300, +1300]$ RPM. Therefore, the tachometer feedback is biased to create a unipolar signal, then reduced in magnitude to the $[0, 1.8]$ volt range the FPTA2 can accept.

3 Conventional Analog Controller

3.1 Design

All closed-loop control systems require the calculation of an error measure, which is manipulated by the controller to produce a control input to the dynamic system being controlled, commonly referred to as the plant. The most widely used form of analog controller is a proportional-integral (PI) controller. This controller is frequently used to provide current control and speed control for a motor. The PI control law is given in Equation 1,

$$u(t) = K_P e(t) + \int \frac{1}{K_I} e(t) dt . \quad (1)$$

where $e(t)$ is the difference between the desired plant response and the actual plant response, K_P is called the proportional gain, and K_I is called the integral gain. In this control law, the proportional and integral terms are separate and added together to form the control input to the plant. The proportional gain is set to provide quick response to changes in the error, and the integral term is set to null out steady state error.

The FPTA2 is a unipolar device using voltages in the range of 0 to 1.8 volts. In order to directly compare a conventional analog controller design with evolved designs, the PI controller must be implemented as shown in Figure 2. This figure includes the circuitry needed to produce the error signal. Equation 2 gives the error voltage, V_e , given the desired response V_{SP} , or setpoint, and the measured motor speed V_{TACH} . The frequency domain transfer function for the voltage output, V_u , of the controller, given V_e , is shown in Equation 3,

$$V_e = \frac{V_{SP}}{2} - \frac{V_{TACH}}{2} + 0.9V . \quad (2)$$

$$V_u = (V_e - V_{bias2}) \left(\frac{R_2}{R_1} + \frac{1}{sR_1C} \right) + V_e . \quad (3)$$

where s is complex frequency in rad/sec, $\frac{R_2}{R_1}$ corresponds to the proportional gain and $\frac{1}{R_1C}$ corresponds to the integral gain. This conventional design requires four op-amps. Two are used to isolate voltage references V_{bias1} and V_{bias2} from the rest of the circuitry, thereby maintaining a steady bias voltage in each case. V_{bias2} must be adjusted to provide a plant response without a constant error bias. The values for R_1 , R_2 , and C are chosen to obtain the desired motor speed response.

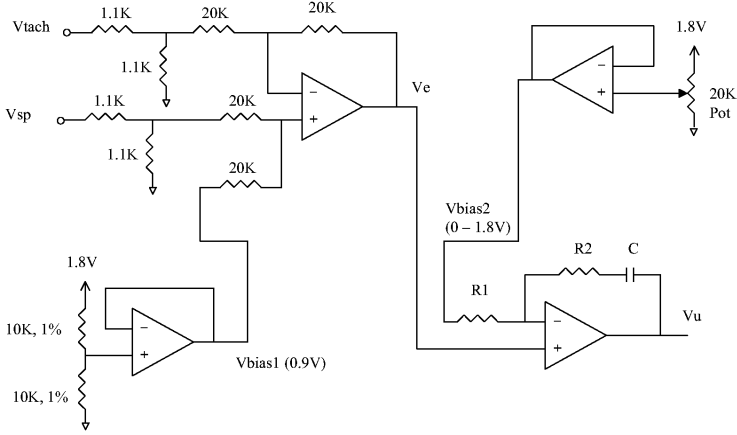


Fig. 2. Unipolar analog PI controller with associated error signal calculation and voltage biasing

3.2 Performance

The controller circuitry in Figure 2 is used to provide a baseline control response to compare with the responses obtained via evolution. The motor is run with no external torque load on the shaft. The controller is configured with $R_1 = 10K$ ohms, $R_2 = 200K$ ohms, and $C = 0.47\mu F$. V_{bias2} is set to 0.854 volts. Figure 3 illustrates the response obtained for V_{SP} consisting of a 2 Hz sinusoid with amplitude in the range of approximately 500 millivolts to 1.5 Volts, as well as for V_{SP} consisting of a 2 Hz square wave with the same magnitude. Statistical analysis of the error for sinusoidal V_{SP} is presented in Table 1 for comparison with the evolved controller responses. Table 2 gives the rise time and error statistics at steady state for the first full positive going transition in the square wave response. This is the equivalent of analyzing a step response. Note that in both cases V_{TACH} tracks V_{SP} very well. In the sinusoid case, there is no visible error between the two. For the square wave case, the only visible error is at the instant V_{SP} changes value. This is expected, because no practical servo-motor can follow instantaneous changes in speed. There is always some lag between the setpoint and response. After the transition, the PI controller does not overshoot the steady state setpoint value, and provides good regulation of motor shaft speed at the steady state values.

4 Evolved Controllers

Two cells within the FPTA2 are used in the evolution of the motor speed controllers. The first cell is provided with the motor speed setpoint, V_{SP} , and the motor shaft feedback, V_{TACH} , as inputs, and it produces the controller output, V_u . An adjacent cell is used to provide support electronics for the first cell. The evolution uses a fitness function based on the error between V_{SP} and V_{TACH} .

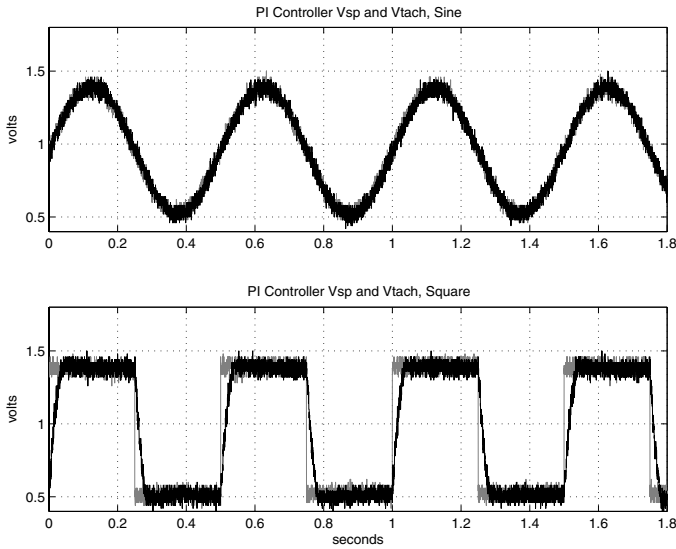


Fig. 3. Response obtained using PI controller. V_{sp} is gray, V_{tach} is black

Lower fitness is better, because the goal is to minimize the error. The population is randomly generated, and then modified to ensure that, initially, the switches are closed that connect V_{SP} and V_{TACH} to the internal reconfigurable circuitry. This is done because the evolution will, in many cases, attempt to control the motor speed by using the setpoint signal only, resulting in an undesirable "controller" with poor response characteristics. Many evolutions were run, and the frequency of the sinusoidal signal was varied, along with the population size and the fitness function. There were some experiments that failed to produce a desirable controller and some that produced very desirable responses, with the expected distribution of mediocre controllers in between. Two of the evolved controllers are presented along with the response data for comparison to the PI controller. The first is the best evolved controller obtained, so far, and the second provides a reasonable control response with an interesting circuit design. In each case, the data presented in the plots was obtained by loading the previously evolved design on the FPTA2, and then providing V_{SP} via a function generator. The system response was recorded using a digital storage oscilloscope.

4.1 Case 1

For this case the population size is 100 and a roughly 2 Hz sinusoidal signal was used for the setpoint. For a population of 100, the evaluation of each generation takes 45 seconds. The target fitness is 400,000 and the fitness function used is,

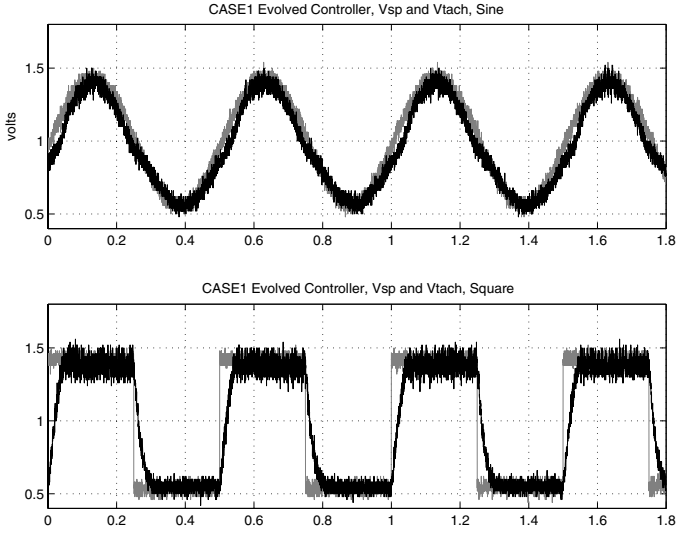


Fig. 4. Response obtained using CASE1 evolved controller. Vsp is gray, Vtach is black

$$F = 0.04 * \sum_{i=1}^n e_i^2 + \frac{100}{n} \sum_{i=1}^n |e_i| + 100000 * \text{not}(S_{57} \vee S_{53}) . \quad (4)$$

where e_i is the error between V_{SP} and V_{TACH} at each voltage signal sample, n is the number of samples over one complete cycle of the sinusoidal input, and S_{57} , S_{53} represent the state of the switches connecting V_{SP} and V_{TACH} to the reconfigurable circuitry. This fitness function punishes individuals that do not have switches S_{57} and S_{53} closed. The location of these switches can be seen in the cell diagram in the Appendix. V_{SP} is connected to Cell_in6 and V_{TACH} is connected to Cell_in2. The evolution converged to a fitness of 356,518 at generation 97. The fitness values are large due to the small values of error that are always present in a physical system. Figure 4 illustrates the response obtained for V_{SP} consisting of a 2 Hz sinusoid with amplitude in the range of approximately 500 millivolts to 1.5 Volts, as well as for V_{SP} consisting of a 2 Hz square wave with the same magnitude. This is the same input used to obtain controlled motor speed responses for the PI controller.

In the sinusoidal case, the evolved controller is able to provide good peak to peak magnitude response, but is not able to track V_{SP} as it passes through 0.9. The evolved controller provides a response to the square wave V_{SP} , which has a slightly longer rise time but provides similar regulation of the speed at steady state. The statistical analysis of the CASE 1 evolved controller response to the sinusoidal V_{SP} is presented in Table 1. Note the increase in all the measures, with the mean error indicating a larger constant offset in the error response. Despite these increases, the controller response is reasonable and could

Table 1. Error metrics for sinusoidal response

Controller	Max Error	Mean Error	Std Dev Error	RMS Error
PI	0.16 V	0.0028 V	0.0430 V	0.0431 V
CASE1	0.28 V	0.0469 V	0.0661 V	0.0810 V

Table 2. Response and error metrics for square wave. First full positive transition only

Controller	Rise Time	Mean Error	Std Dev Error	RMS Error
PI	0.0358 sec	0.0626 V	0.1816 V	0.1920 V
CASE1	0.0394 sec	0.1217 V	0.2026 V	0.2362 V

be considered good enough. The rise time and steady state error analysis for the first full positive going transition in the square wave response is given in Table 2. While there is an increase in rise time and in the error measures at steady state, when compared to those of the PI controller, the evolved controller can be considered to perform very well. Note again that the increase in the mean error indicates a larger constant offset in the error response. In the PI controller, this error can be manually trimmed out via adjustment of V_{bias2} . The evolved controller has been given no such bias input, so some increase in steady state error should be expected. However, the evolved controller is trimming this error, because other designs have a more significant error offset. Experiments with the evolved controller show that the "support" cell is providing the error trimming circuitry.

It is notable that the evolved controller is providing a good response using a considerably different set of components than the PI controller. The evolved controller is using two adjacent cells in the FPTA to perform a similar function to four op-amps, a collection of 12 resistors and one capacitor. The FPTA switches have inherent resistance on the order of kilo-ohms, which can be exploited by evolution during the design. But the two cells can only be used to implement op-amp circuits similar to those in Figure 2 with the use of external resistors, capacitors and bias voltages. These external components are not provided. The analysis of the evolved circuit is complicated and will not be covered in more detail here.

4.2 Case 2

This evolved controller is included, not because it represents a better controller, but because it has an interesting characteristic. In this case, the population size is 200 and a roughly 3 Hz sinusoidal signal was used for the setpoint during evolution. For a population of 200, the evaluation of each generation takes 90 seconds. The fitness function is the same as used for Case 1, with one exception, as shown in Equation 5.

$$F = 0.04 * \sum_{i=1}^n e_i^2 + \frac{100}{n} \sum_{i=1}^n |e_i| . \quad (5)$$

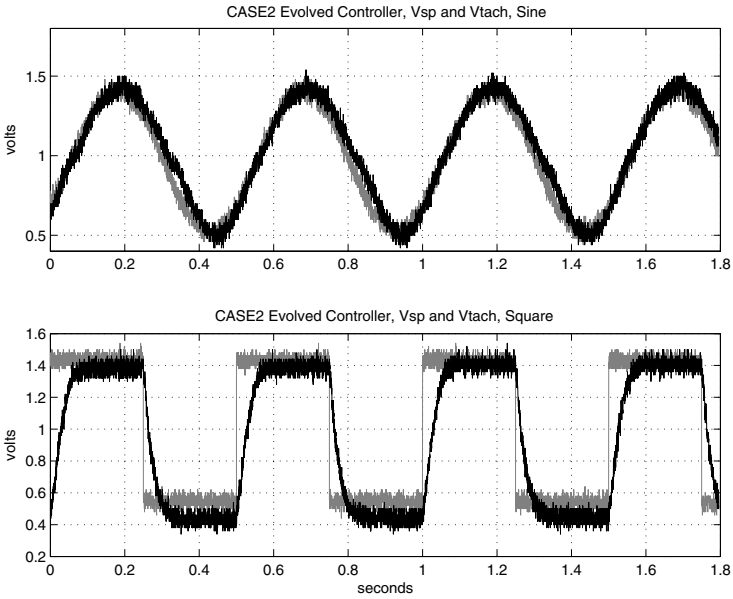


Fig. 5. Response obtained using CASE2 evolved controller. Vsp is gray, Vtach is black

In this case, the switches S_{57} , S_{53} are forced to be closed (refer to the cell diagram in the appendix), and so no penalty based on the state of these switches is included in the fitness function. The evolution converged to a fitness of approximately 1,000,000, and was stopped at generation 320. The interesting feature of this design is that switches S_{54} , S_{61} , S_{62} , S_{63} are all open. This indicates that the V_{TACH} signal is not directly connected to the internal circuitry of the cell. However, the controller is using the feedback, because opening S_{53} caused the controller to no longer work. The motor speed response obtained using this controller can be seen in Figure 5. The response to sinusoidal V_{SP} is good, but exhibits noticeable transport delay on the negative slope. The response to the square wave V_{SP} exhibits offset for the voltage that represents a "negative" speed. Overall the response is reasonably good. The analysis of this evolved controller is continuing in an effort to understand precisely how the controller is using the V_{TACH} signal internally.

5 Summary

The results presented show the FPTA2 can be used to evolve simple analog closed-loop controllers. The use of two cells to produce a controller that provides good response in comparison with a conventional controller shows that hardware evolution is able to create a compact design that still performs as re-

quired, while using less transistors than the conventional design, and no external components. Recall that one cell is can be used to implement an op-amp design on the FPTA2. While a programmable device has programming overhead that fixed discrete electronic and integrated circuit components do not, this overhead is typically neglected when comparing the design on the programmable device to a design using fixed components. The programming overhead is indirect, and is not a functional component of the design. As such, the cell diagram in the Appendix shows that each cell contains 15 transistors available for use as functional components in the design. Switches have a finite resistance, and therefore functionally appear as passive components in a cell. The *simplified* diagram in the data sheets for many op-amps indicate that 30, or more, transistors are utilized in their design, and op-amp circuit designs require multiple external passive components.

In order to produce self-configuring controllers that can rapidly converge to provide desired performance, more work is needed to speed up the evolution and guide it to the best response. The per generation evaluation time of 45 or more seconds is a bottleneck to achieving this goal. Further, the time constants of a real servo-motor may make it impossible to achieve more rapid evaluation times. Most servo-motor driven actuators cannot respond to inputs with frequency content of more than a few tens of Hertz, without attenuation in the response. Alternative methods of guiding the evolution or novel controller structures are required.

A key to improving upon this work and evolving more complex controllers is a good understanding of the circuits that have been evolved. Evolution has been shown to make use of parasitic effects and to use standard components in novel, and often difficult to understand, ways. Case 2 illustrates this notion. Gaining this understanding may prove to be useful in developing techniques for guiding the evolution towards rapid convergence.

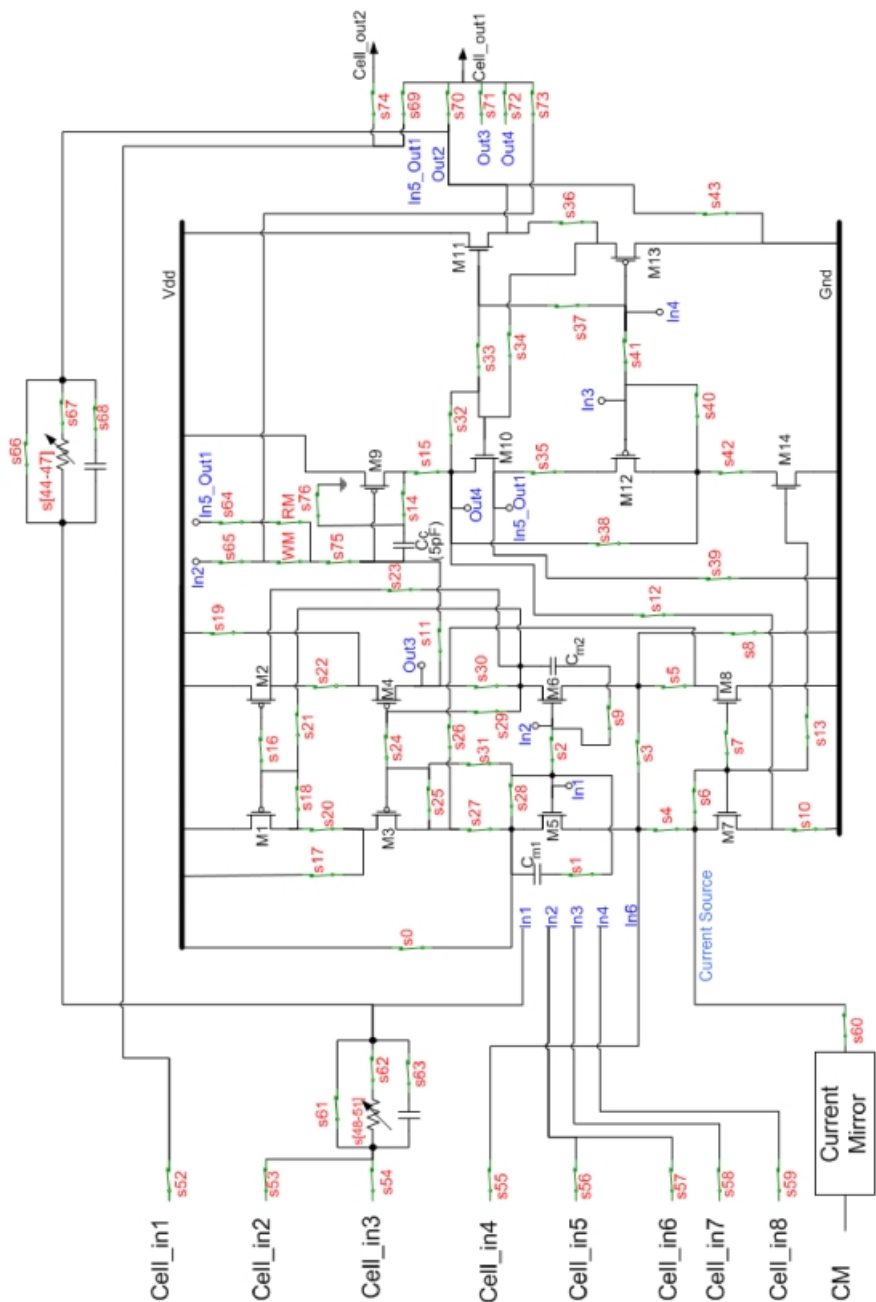
Acknowledgements. The authors would like to thank Jim Steincamp and Adrian Stoica for establishing the initial contact between Marshall Space Flight Center and the Jet Propulsion Laboratory leading to the collaboration for this work. The Marshall team appreciates JPL making available their FPTA2 chips and SABLE system design for conducting the experiments. Jim Steincamps continued support and helpful insights into the application of genetic algorithms have been a significant contribution to this effort.

References

- [1] Lohn, J. D. and Columbano, S. P., *A Circuit Representation Technique for Automated Circuit Design*, IEEE Transactions on Evolutionary Computation, Vol. 3, No. 3, September 1999.
- [2] Stoica, A., Zebulum, R., Keymeulen, D., *Progress and Challenges in Building Evolvable Devices*, Evolvable Hardware, Proceedings of the third NASA/DoD Workshop on, July 2001, pp 33–35.

- [3] Ferguson, M. I., Zebulum, R., Keymeulen, D. and Stoica, A., *An Evolvable Hardware Platform Based on DSP and FPTA*, Late Breaking Papers at the Genetic and Evolutionary Computation Conference (GECCO-2002), July 2002, pp. 145–152.
- [4] Stoica, A., Zebulum, R., Ferguson, M. I., Keymeulen, D. and Duong V., *Evolving Circuits in Seconds: Experiments with a Stand-Alone Board Level Evolvable System*, 2002 NASA/DoD Conference on Evolvable Hardware, July 2002, pp. 67–74.
- [5] Langeheine, J., Meier, K., Schemmel, J., *Intrinsic Evolution of Quasi DC solutions for Transistor Level Analog Electronic Circuits Using a CMOS FTPA Chip*, 2002 NASA/DoD Conference on Evolvable Hardware, July 2002, pp. 75–84.
- [6] Flockton, S. J. and Sheehan, K., “Evolvable Hardware Systems Using Programmable Analogue Devices”, *Evolvable Hardware Systems* (Digest No. 1998/233), IEE Half-day Colloquium on , 1998 ,Page(s): 5/1–5/6.
- [7] Ozsvald, Ian, “Short-Circuit the Design Process: Evolutionary Algorithms for Circuit Design using Reconfigurable Analogue Hardware”, Master’s Thesis, University of Sussex, September, 1998.
- [8] Koza, J. R., Keane, M. A., Yu, J., Mydlowec, W. and Bennet, F., *Automatic Synthesis of Both the Control Law and Parameters for a Controller for a Three-lag plant with Five-Second delay using Genetic Programming and Simulation Techniques*, American Control Conference, June 2000.
- [9] Keane, M. A., Koza, J. R., and Streeter, M.J., *Automatic Synthesis Using Genetic Programming of an Improved General-Purpose Controller for Industrially Representative Plants*, 2002 NASA/DoD Conference on Evolvable Hardware, July 2002, pp. 67–74.
- [10] Zebulum, R. S., Pacheco, M. A., Vellasco, M., Sinohara, H. T., *Evolvable Hardware: On the Automatic Synthesis of Analog Control Systems*, 2000 IEEE Aerospace Conference Proceedings, March 2000, pp 451–463.
- [11] Raimondi, G. M., et. al., *Large Electromechanical Actuation Systems for Flight Control Surfaces*, IEE Colloquium on All Electronic Aircraft, 1998.
- [12] Jensen, S.C., Jenney, G. D., Raymond, B., Dawson, D., *Flight Test Experience with an Electromechanical Actuator on the F-18 Systems Research Aircraft*, Proceedings of the 19th Digital Avionics System Conference, Volume 1, 2000.
- [13] Byrd, V. T., Parker, J. K, *Further Consideration of an Electromechanical Thrust Vector Control Actuator Experiencing Large Magnitude Collinear Transient Forces*, Proceedings of the 29th Southeastern Symposium on System Theory, March 1997, pp 338–342.

Appendix: FPTA2 Cell Diagram



An Examination of Hypermutation and Random Immigrant Variants of mrCGA for Dynamic Environments

Gregory R. Kramer and John C. Gallagher

Department of Computer Science and Engineering
Wright State University, Dayton, OH, 45435-0001
{gkramer, johng}@cs.wright.edu

1 Introduction

The mrCGA is a GA that represents its population as a vector of probabilities, where each vector component contains the probability that the corresponding bit in an individual's bitstring is a one [2]. This approach offers significant advantages during hardware implementation for problems where power and space are severely constrained. However, the mrCGA does not currently address the problem of continuous optimization in a dynamic environment. While, many dynamic optimization techniques for population-based GAs exist in the literature, we are unaware of any attempt to examine the effects of these techniques on probability-based GAs. In this paper we examine the effects of two such techniques, hypermutation and random immigrants, which can be easily added to the existing mrCGA without significantly increasing the complexity of its hardware implementation. The hypermutation and random immigrant variants will be compared to the performance of the original mrCGA on a dynamic version of the single-leg locomotion benchmark.

2 Dynamic Optimization Variants of mrCGA

The hypermutation strategy, proposed in [1], increases the mutation rate following an environmental change and then slowly decreases it back to its original level. For this problem the hypermutation variant was set to increase the mutation rate from 0.05 to 0.1. Random immigrants is another strategy that diversifies the population by inserting random individuals [4]. Simulating the insertion of random individuals is accomplished in the probability vector by shifting each bit probability toward its original value of 50%. For this problem the random immigrants variant was set to shift each bit probability by 0.12. To ensure fair comparisons between the two variants, the hypermutation rate, and the bit probability shift were empirically determined to produce roughly the same divergence in the GA's population.

3 Testing and Results

The mrCGA and its variants were tested on the single-leg robot locomotion problem. The goal for this problem is to evolve a five neuron CTRNN (Continuous Time Recurrent Neural Network) controller that allows the robot to walk forward at optimal speed. Each benchmark run consisted of 50,000 evaluation cycles with the leg's length and angular inertia changed every 5,000 evaluation cycles. The algorithms were each run 100 times on this problem. Performance was evaluated by examining the quality of the final solution achieved prior to each leg model change. A more formal examination of the single-leg locomotion problem can be found in [3].

Comparisons between the mrCGA, hypermutation, and random immigrant results show that the best solutions are achieved by the hypermutation variant. The average pre-shift error for the mrCGA is 18.12%, whereas the average pre-shift error for the hypermutation variant shows a 2.27% decrease to 15.85%. In contrast, the random immigrant variant performed worse than mrCGA, with a 4.18% increase in error to 22.30%.

4 Conclusions

Our results show that for the single-leg locomotion problem, hypermutation increases the quality of the mrCGA's solution in a dynamic environment, whereas the random immigrant variant produces slightly lower scores. Both of these variants can be easily added to the existing mrCGA hardware implementation without significantly increasing its complexity. In the future we plan to categorize the effects of the hypermutation and random immigrant strategies on the mrCGA for a variety of generalized benchmarks. This categorization will be useful to help determine which dynamic optimization strategy should be employed for a given problem.

References

1. Cobb, H.G. (1990) *An investigation into the use of hypermutation as an adaptive operator in genetic algorithms having continuous, time-dependent nonstationary environments*. Technical Report AIC-90-001, Naval Research Laboratory, Washington, USA.
2. Gallagher, J.C. & Vigraham, S. (2002) *A Modified Compact Genetic Algorithm for the Intrinsic Evolution of Continuous Time Recurrent Neural Networks*. The Proceedings of the 2002 Genetic and Evolutionary Computation Conference. Morgan-Kaufmann.
3. Gallagher, J.C., Vigraham, S., & Kramer, G.R. (2002) *A Family of Compact Genetic Algorithms for Intrinsic Evolvable Hardware*.
4. Grefenstette, J.J. (1992) *Genetic algorithms for changing environments*. In R. Maenner and B. Manderick, editors, *Parallel Problem Solving from Nature 2*, pages 137–144. North Holland.

Inherent Fault Tolerance in Evolved Sorting Networks

Rob Shepherd and James Foster*

Department of Computer Science, University of Idaho, Moscow, ID 83844

rob_shepherd@bigfoot.com

foster@cs.uidaho.edu

Abstract. This poster paper summarizes our research on fault tolerance arising as a by-product of the evolutionary computation process. Past research has shown evidence of robustness emerging directly from the evolutionary process, but none has examined the large number of diverse networks we used. Despite a thorough study, the linkage between evolution and increased robustness is unclear.

Discussion

Previous research has suggested that evolutionary search techniques may produce some fault tolerance characteristics as a by-product of the process. Masner et al. [1, 2] found evidence of this while evolving sorting networks, as their evolved circuits were more tolerant of low-level logic faults than hand-designed networks. They also introduced a new metric, bitwise stability (*BS*), to measure the degree of robustness in sorting networks.

We evaluated the hypothesis that evolved sorting networks were more robust than those designed by hand, as measured by *BS*. We looked at sorting networks with larger numbers of inputs to see if the results reported by Masner et al. would still be apparent. We selected our subject circuits from three primary sources: hand-designed, evolved and “reduced” networks. The last category included circuits manipulated using Knuth’s technique in which we created a sorter for a certain number of inputs by eliminating inputs and comparators from an existing network [3].

Masner et al. found that evolution produced more robust 6-bit sorting networks than hand-designed ones reported in the literature. We expanded our set of comparative networks, comprising 157 circuits sorting between 4 and 16 inputs. Our 16 bit networks were only used as the basis for other reduced circuits.

Table 1 shows the results for our entire set of circuits. We listed the 3 best networks for each width to give some sense of the inconsistency between design methods. As with the 4-bit sorters, evolution produced the best 5-, 7- and 10-bit circuits, but reduction was more effective for 6, 9, 12 and 13 inputs. Juillé’s evolved 13-bit

* Foster was partially funded for this research by NIH NCRR 1P20 RR16448.

network (J13b_E) was inferior to the reduced circuits and Knuth's 12-bit sorter (Kn12b_H) was the only hand-designed network to make this list.

Table 1. Top 3 results for all sorting networks in Shepherd [4]. K represents the number of inputs to the network and BS indicates the bitwise stability, as defined in [1]. The last character of the index indicates the design method: E for evolved, H for hand-designed, R for reduced

K	Best circuit		2nd best circuit		3rd best circuit	
	Index	BS	Index	BS	Index	BS
4	M4A_E	0.943359	M4Rc_E	0.942057	Kn4Rd_R	0.941840
5	M5A_E	0.954282	M5Rd_R	0.954028	M5Rc_R	0.953935
6	M6Ra_R	0.962836	Kn6Ra_R	0.962565	M6A_E	0.962544
7	M7_E	0.968276	M7Rc_R	0.968206	M7Ra_R	0.967892
9	M9R_R	0.976066	G9R_R	0.975509	Kn9Rb_R	0.975450
10	M10A_E	0.978257	H10R_R	0.978201	G10R_R	0.978189
12	H12R_R	0.981970	G12R_R	0.981932	Kn12b_H	0.981832
13	H13R_R	0.983494	G13R_R	0.983461	J13b_E	0.983305

Our data do not support our hypothesis that evolved sorting networks are more robust, in terms of bitwise stability, than those designed by hand. Masner's early work showed evolution's strength in generating robust networks, but support for the hypothesis evaporated as we added more circuits to our comparison set, to the point that there is no clear evidence that one design method inherently produces more robust sorting networks. Our data do not necessarily disconfirm our hypothesis, but leave it open for further examination. One area for future study is the linkage between faults and the evolutionary operators. Thompson [5] used a representation method in which faults and genetic mutation had the same effect, but these operators affected different levels of abstraction in our model.

References

1. Masner, J., Cavalieri, J., Frenzel, J., & Foster, J. (1999). Representation and Robustness for Evolved Sorting Networks. In Stoica, A., Keymeulen, D., & Lohn, J., (Eds.), *The First NASA/DoD Workshop on Evolvable Hardware*, California: IEEE Computer Society, 255–261.
2. Masner, J. (2000). *Impact of Size, Representation and Robustness in Evolved Sorting Networks*. M.S. thesis, University of Idaho.
3. Knuth, D. (1998). *The Art of Computer Programming, Volume 3: Sorting and Searching*, Second Edition, Massachusetts: Addison-Wesley, 219–229.
4. Shepherd, R. (2002). *Fault Tolerance in Evolved Sorting Networks: The Search for Inherent Robustness*. M.S. thesis, University of Idaho.
5. Thompson, A. (1995). Evolving fault tolerant systems. In *Proceedings of the 1st IEE/IEEE International Conference on Genetic Algorithms in Systems: Innovations and Applications (GALESIA '95)*. IEE Conference Publication No. 414, 524–529.

Co-evolving Task-Dependent Visual Morphologies in Predator-Prey Experiments

Gunnar Buason and Tom Ziemke

Department of Computer Science, University of Skövde
Box 408, 541 28 Skövde, Sweden
{gunnar.buason,tom}@ida.his.se

Abstract. This article presents experiments that integrate competitive co-evolution of neural robot controllers with ‘co-evolution’ of robot morphologies and control systems. More specifically, the experiments investigate the influence of constraints on the evolved behavior of predator-prey robots, especially how task-dependent morphologies emerge as a result of competitive co-evolution. This is achieved by allowing the evolutionary process to evolve, in addition to the neural controllers, the view angle and range of the robot’s camera, and introducing dependencies between different parameters.

1 Introduction

The possibilities of evolving both behavior and structure of autonomous robots has been explored by a number of researchers [5, 7, 10, 15]. The artificial evolutionary approach is based upon the principles of natural evolution and the survival of the fittest. That means, robots are not pre-programmed to perform certain tasks, but instead they are able to ‘evolve’ their behavior. This, to a certain level, decreases human involvement in the design process as the task of designing the behavior of the robot is moved from the distal level of the human designer down to the more proximal level of the robot itself [13, 16]. As a result, the evolved robots are, at least in some cases, able to discover solutions that might not be obvious beforehand to human designers.

A further step in minimizing human involvement is adopting the principles of competitive co-evolution (CCE) from nature, where in many cases two or more species live, adapt and co-evolve together in a delicate balance. The adaptation of this approach in Evolutionary Robotics allows for simpler fitness function and that the evolved behavior of both robot species emerges in incremental stages [13]. The use of this approach has been extended, not only co-evolving the neural control system of two competing robotic species, but also ‘co-evolving’ the neural control system of a robot together with its morphology.

The experiments performed by Cliff and Miller [5, 6] can be mentioned as examples of demonstrations of CCE in evolutionary robotics, both concerning evolution of morphological parameters (such as ‘eye’ positions) and behavioral strategies between two robotic species. More recent experiments are the ones performed by Nolfi and Floreano [7, 8, 9, 12]. In a series of experiments they studied different aspects of CCE of neural robot controllers in a predator-prey scenario. In

one of their experiments [12] Nolfi and Floreano demonstrated that the robots' sensory-motor structure had a large impact on the evolution of behavioral (and learning) strategies, resulting in a more natural 'arms race' between the robotic species. Different authors have further pointed out in [14, 15] that an evolutionary process that allows the integrated evolution of morphology and control might lead to completely different solutions that are to a certain extent less biased by the human designer.

The aim of our overall work has been to further systematically investigate the tradeoffs and interdependencies between morphological parameters and behavioral strategies through a series of predator-prey experiments in which increasingly many aspects are subject to self-organization through CCE [1, 3]. In this article we only present experiments that extend the experiments of Nolfi and Floreano [12] considering two robots, both equipped with cameras, taking inspiration mostly from Cliff and Miller's [6] work on the evolution of "eye" positions. However, the focus will not be on evolving the positions of the sensors on the robot alone but instead on investigating the trade-offs the evolutionary process makes in the robot morphology as a result of different constraints and dependencies, both implicit and explicit. The latter is in line with the research of Lee et al. [10] and Lund et al. [11].

2 Experiments

The experiments described in this paper focus on evolving the weights of the neural network, i.e. the control system, and the view angle of the camera (0 to 360 degrees) as well as its range (5 to 500 mm) of two predator-prey robots. That means, only a limited number of morphological parameters were evolved. The size of the robot was kept constant, assuming a Khepera-like robot, using all the infrared sensors, for the sake of simplicity. In addition constraints and dependencies were introduced, e.g. by letting the view angle constrain the maximum speed, i.e. the larger the view angle, the lower the maximum speed the robot was allowed to accelerate to. This is in contrast to the experiments in [7, 8, 9, 12], where the predator's maximum speed was always set to half the prey's. All experiments were replicated three times.

2.1 Experimental Setup

For finding and testing the appropriate experimental settings a number of pilot experiments were performed [1]. The simulator used in this work is called YAKS [4], which is similar to the one used in [7, 8, 9, 12]. YAKS simulates the popular Khepera robot in a virtual environment defined by the experimenter (cf. Fig. 1). The simulation of the sensors is based on pre-recorded measurements of a real Khepera robot's infrared sensors and motor commands at different angles and distances [1].

The experimental framework that was implemented in the YAKS simulator was in many ways similar to the framework used in [7, 8, 9, 12]. What differed was that in our work we used a real-valued encoding to represent the genotype instead of direct

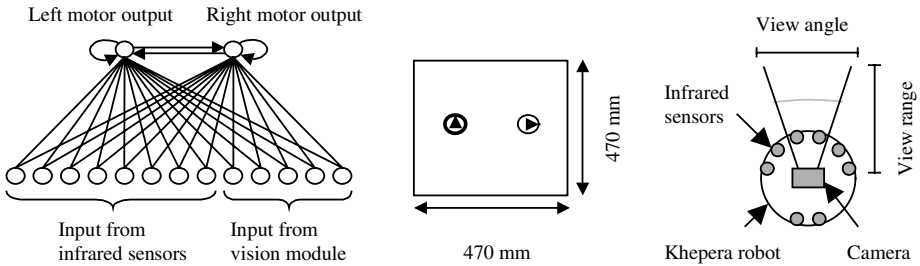


Fig. 1. Left: Neural network control architecture (adapted from [7]). Center: Environment and starting positions. The thicker circle represents the starting position of the predator while the thinner circle represents the starting position of the prey. The triangles indicate the starting orientation of the robots, which is random for each generation. Right: Khepera robot equipped with eight short-range infrared sensors and a vision module (a camera).

encoding, and the number of generations was extended from 100 to 250 generations to allow us to observe the morphological parameters over longer period of time. Beside that, most of the evolutionary parameters were ‘inherited’ such as the use of *elitism* as a selection method, choosing the 20 best individuals from a population of 100 for reproduction. In addition, a similar fitness function was used. Maximum fitness was one point while minimum fitness was zero points. The fitness was a simple time-to-contact measurement, giving the selection process finer granularity, where the prey achieved the highest fitness by avoiding the predator for as long as possible while the predator received the highest fitness by capturing the prey as soon as possible. The competition ended if the prey survived for 500 time steps or when the predator made contact with the prey before that.

For each generation the individuals were tested for ten epochs. During each epoch, the current individual was tested against one of the best competitors of the ten previous generations. At generation zero, competitors were randomly chosen within the same generation, whereas in the other nine initial generations they were randomly chosen from the pool of available best individuals of previous generations. This is in line with the work of [7, 8, 9, 12]. In addition, the same environment as in [7, 8, 9, 12] was used (cf. Fig. 1).

A simple recurrent neural network architecture was used, similar to the one used in [7, 8, 9, 12] (cf. Fig. 1). The experiments involved both robots using the camera so each control network had eight input neurons for receiving input from the infrared sensors and five input neurons for the camera. The neural network had one sigmoid output neuron for each motor of the robot. The vision module, which was only one-dimensional, was implemented with flexible view range and angle while the number of corresponding input neurons was kept constant.

For each experiment, the weights of the neural network were initially randomized and evolved using a Gaussian distribution with a standard deviation of 2.0. The starting values of angle and range were randomized using a uniform distribution function, and during evolution the values were mutated using Gaussian distribution with a standard deviation of 5.0. The view angle could evolve up to 360 degrees; if the random function generated a value of over 360 degrees then the view angle was

set to 360 degrees. The same was valid for the lower bounds of the view angle and also for the lower and upper bounds of the view range.

Constraints, such as those used in [7, 8, 9, 12], where the maximum speed of the predator was only half the prey's, were adapted here where speed was dependent on the view angle. For this, the view angle was divided into ten intervals covering 36 degrees each¹. The maximum speed of the robot was then reduced by 10% for each interval, e.g. if the view angle was between 0 and 36 degrees there were no constraints on the speed, and if it was a value between 36 and 72 degrees, the maximum speed of the robot was limited to 90% of its original maximum speed.

2.2 Results

The experiments were analyzed using fitness measurements, Master Tournament [7] and collection of CIAO data [5]. A Master Tournament shows the performance of the best individuals of each generation tested against all best competitors from that replication. CIAO data are fitness measurements collected by arranging a tournament where the current individual of each generation competes against all the best competing ancestors [5]. In addition some statistical calculations and behavioral observations were performed. Concerning analysis of the robots' behavior, trajectories from different tournaments will be presented together with qualitative descriptions. Here a summary of the most interesting results will be given (for further details see [1]).

Experiment A: Evolving the Vision Module

This experiment (cf. experiment 9 in [1]) extends Nolfi and Floreano's experiment in [12]. What differs is that here the view angle and range are evolved instead of being constant. In addition, the speed constraints were altered by setting the maximum speed to the same value for both robots, i.e. 1.0, and instead the maximum speed of the predator was constrained by its view angle.

Nolfi and Floreano [12] performed their experiments in order to investigate if more interesting arms races would emerge if the richness of the sensory mechanisms of the prey was increased by giving it a camera. The results showed that "by changing the initial conditions 'arms races' can continue to produce better and better solutions in both populations without falling into cycles" [12]. That is, the prey is able to refine its strategy to escape the predator instead of radically changing it. In our experiments the results varied between replications when considering this aspect, i.e. the prey was not always able to evolve a suitable evasion strategy.

Fig. 2 presents the results of the Master Tournament. The graph presents the average results of ten runs, i.e. each best individual was tested for ten epochs against its opponent. Maximum fitness achievable was 250 points as there were 250 opponents. As Fig. 2 illustrates, both predator and prey make evolutionary progress initially, but in later generations only the prey exhibits steady improvement.

The text on the right in Fig. 2 summarizes the Master Tournament. The two upper columns describe in what generation it is possible to find the predator respectively the

¹ Alternatively, a linear relation between view angle and speed could be used.

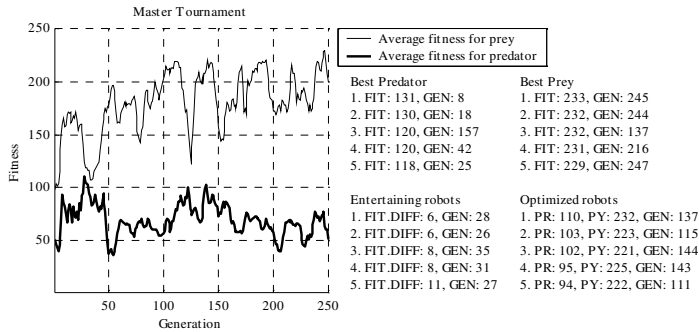


Fig. 2. Master Tournament (cf. Experiment 9 in [1, 2]). The data was smoothed using rolling average over three data points. The same is valid for all following Master Tournament graphs. Observe that the values in the text to the right have not been smoothed, and therefore do not necessarily fit the graph exactly.

best prey with the highest fitness score. The lower left column demonstrates where it is possible to find the most entertaining tournaments, i.e. robots that report similar fitness have a similar chance of winning. The lower right column demonstrates where in the graph the most optimized robots can be found, i.e. generations of robots where both robots have high fitness values.

The left graphs of Fig. 3 display the evolution of view angle and range for the predator and prey, i.e. the evolved values from the best individual from each generation. For the predator the average view range evolved was 344 mm and the average view angle evolved was 111° . It does not seem that the evolutionary process found a balance while evolving the view range as the standard deviation is 105 mm, but the view angle is more balanced with a standard deviation of 48° . The prey evolved an average view range of 247 mm (with a standard deviation of 125 mm) and an average view angle of 200° (with a standard deviation of 86°). These results indicate that the predator prefers a rather narrow view angle with a rather long view range (in the presence of explicit constraints), while the prey evolves a rather wide view angle with a rather short view range (in the absence of explicit constraints) (cf. Fig. 3).

Fig. 3, right graph, presents a histogram over the number of different angle intervals evolved by the predator. The number above each interval represents the maximum speed interval, e.g. in this case most of the predator individuals evolved a view angle between 108 and 144 degrees and therefore the speed were constrained to be within the interval of 0.0 to 0.7. The distribution seems to be rather normalized over the different view angle intervals (between 0 and 252 degrees) (cf. Fig. 3 right).

In other replications of this experiment, the evolutionary process found a different balance between view angle and speed, where a smaller view angle was evolved with high speed. Unlike the distribution in the right graph in Fig. 3 where a large number of predator individuals prefer to evolve a view angle between 108 and 144 degrees, in other replications the distribution was mostly between 0 and 72 degrees, implying

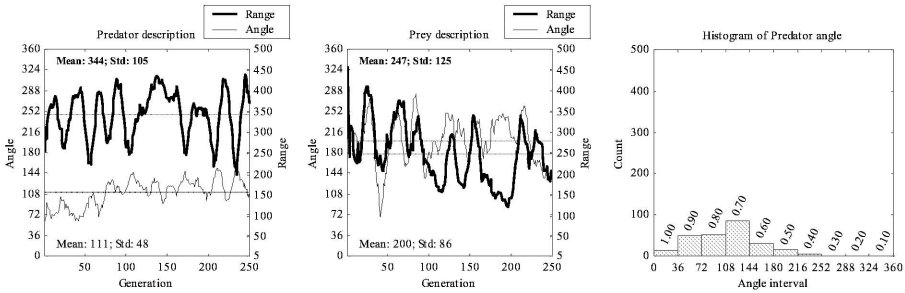


Fig. 3. Left: Morphological description of predator and prey (cf. Experiment 9 in [1, 2]). The graphs present the morphological description of view angle (left y-axis, thin line) and view range (right y-axis, thick line). The values in the upper left corner of the graphs are the mean and standard deviation for the view range over generations, calculated from the best individual from each generation. Corresponding values for the view angle are in the lower left corner. The data was smoothed using rolling average over ten data points. The same is valid for all following morphological description graphs. Right: Histogram over view angle of predator (cf. Experiment 9 in [1, 2]). The graph presents a histogram over view angle, i.e. the number of individuals that preferred a certain view angle. The values above each bin indicate the maximum speed interval.

small, focused view range and high speed. These results, however, depend on the behavior that the prey evolves. If the prey is not successful in evolving its evasion strategy, perhaps crashing into walls, then the predator could evolve a very focused view angle with a high speed. On the other hand, if the prey evolves a successful evasion strategy, moving fast in the environment, then the predator needs a larger view angle in order to be able to follow the prey.

In Fig. 4 a number of trajectories are presented. The first trajectory snapshot is taken from generation 43. This trajectory shows a predator with a view angle of 57° and a view range of 444 mm chasing a prey with a view angle of 136° and a view range of 226 mm. The snapshot is taken after 386 time steps. The prey starts by spinning in place until it notices the predator in its field of vision. Then it starts moving fast in the environment in an elliptical trajectory.

Moving this way the prey (cf. Fig. 4, left part) is able to escape the predator. This is an interesting behavior from the prey as it can only sense the walls with its infrared sensors while the predator needs only to follow the prey in its field of vision in a circular trajectory. However, after few generations the predator loses the ability to follow the prey and never really recovers in later generations. An example of this is the snapshot of a trajectory taken in generation 157 after 458 time steps (Fig 4., right). Here the predator has a 111° view angle and a 437 mm view range while the prey has an 86° view angle and a 251 mm view range. As previously, the prey starts by spinning until it notices the predator in the field of vision. Then it starts moving around in the environment, this time following walls. The predator does not demonstrate any good abilities in capturing the prey. Instead, it spins around in the center of the environment, trying to locate the prey.

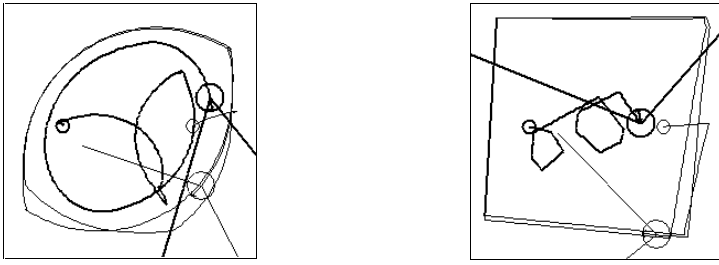


Fig. 4. Trajectories from generation 43 (left) (predator: 57° , 444 mm; prey: 136° , 226 mm) and 157 (right) (predator: 111° , 437 mm; prey: 86° , 251 mm), after 386 and 458 time steps respectively (cf. Experiment 9 in [1]). The predator is marked with a thick black circle and the trajectory with a thick black line. The prey is marked with a thin black circle and the trajectory with a thin black line. Starting positions of both robots are marked with small circles. The view field of the predator is marked with two thick black lines. The angle between the lines represents the current view angle and the length of the lines represents the current view range.

Another interesting observation is that the prey mainly demonstrates the behavior described above, i.e. staying in the same place, spinning, until it sees the predator, and then starts its ‘moving around’ strategy.

Experiment B: Adding Constraints

This experiment (cf. experiment 10 in [1]) extends the previous experiment by adding a dependency between the view angle and the speed of the prey. As previously, the predator is implemented with this dependency. View angle and range of both species are then evolved. The result of this experiment was that the predator became the dominant species (cf. Fig. 5), despite the fact that the prey had certain advantages over the predator considering the starting distance and the fitness function being based on time-to-contact.

A Master Tournament (cf. Fig. 5) illustrates that evolutionary progress only occurs during the first generations and then the species come to a balance where minor changes in the strategy result in a valley in the fitness landscape. To investigate if the species cycle between behaviors, CIAO data was collected. Each competition was run ten times and the results were then averaged, i.e. zero in fitness score is the worst and one in fitness score is the best. The ‘Scottish tartan’ patterns in the graphs (Fig. 6) indicate periods of relative stasis interrupted by short and radical changes of behavior [7]. The CIAO data also shows that the predator is the dominating species. Stripes on the vertical axis in the graph for the prey indicate a good predator where the stripe is black respectively a bad predator where the stripe is white. This is more noticeable for the predator than for the prey, i.e. either the predator is overall good or overall bad while the prey is more balanced.

An interesting aspect is the evolution of the morphology (cf. Fig. 7). The predator, as in the previous experiment, evolves a rather small view angle with a rather long range. The prey also evolves a rather small view angle, in fact a smaller view angle than the predator, and a relative short view range with a relatively high standard deviation.

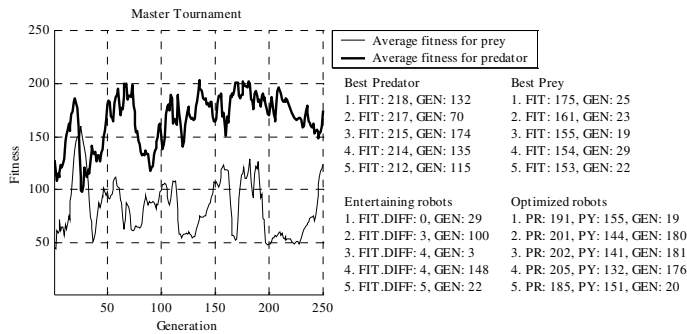


Fig. 5. Master Tournament.

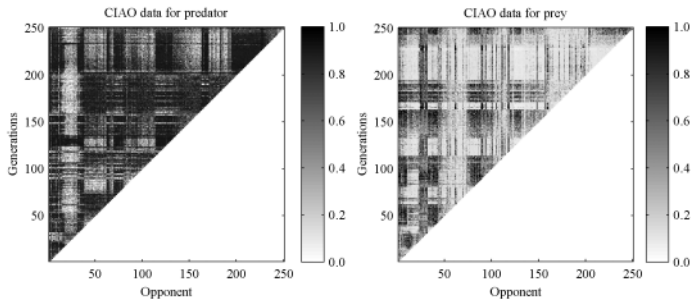


Fig. 6. CIAO data (cf. Experiment 10 in [1, 2]). The colors in the graph represent fitness values of individuals from different tournaments. Higher fitness corresponds to darker colors.

When looking at the relation between view angle and view range in the morphological space then certain clusters can be observed (cf. Fig. 8). The predator descriptions form a cluster in the upper left corner of the area where the view angle is rather focused while the view range is rather long. The interesting part is that the prey also forms clusters with an even smaller view angle, i.e. it ‘chooses’ speed over vision. The clustering of the range varies from small range to very long range, indicating that for the prey the range is not so important.

The evolution of the view angle is further illustrated in Fig. 9. While the predator seems to prefer to evolve a view angle between 36 and 72 degrees, the prey prefers to evolve a view angle between 0 and 36 degrees. This indicates that, in this case the prey prefers speed to vision. The reason behind this lies in the morphology of the robots. The robots have eight infrared sensors, two of them on the rear side of the robots and six of them on the front side. The camera on the robots is placed in a frontal direction, i.e. in the same direction as the six infrared sensors. The robots then mainly use the front infrared sensors for obstacle avoidance. Therefore when the prey evolves a strategy to move fast in the environment because the predator follows it, it

has more use of moving fast than being able to see. Therefore, it more or less ‘ignores’ the camera and evolves the ability to move fast, relying on its infrared sensors.

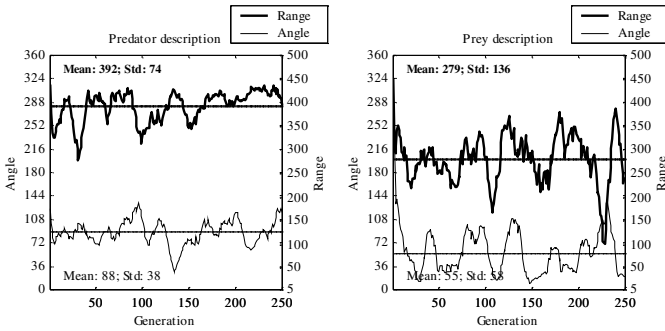


Fig. 7. Morphological descriptions (cf. Experiment 10 in [1, 2]).

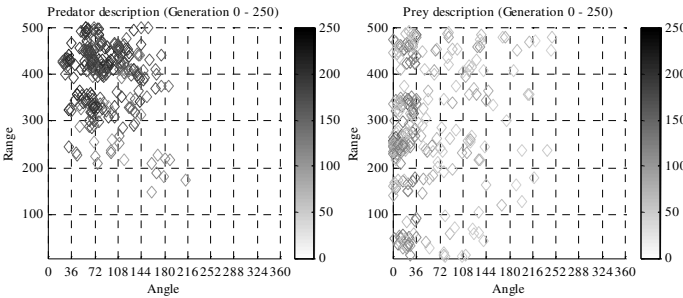


Fig. 8. Morphological space (cf. Experiment 10 in [1, 2]). The graphs present relations between view angle and view range in the morphological space. Each diamond represents an individual from a certain generation. The gray level of the diamond indicates the fitness achieved during a Master Tournament, wit. darker diamonds indicating higher fitness.

A number of trajectories in Fig. 10 display the basic behavior observed during the tournaments. On the left is a trajectory snapshot taken in generation 23 after 377 time steps. The predator has evolved a 99° view angle and a 261 mm view range, while the prey has evolved a 35° view angle and a 484 mm view range. The prey tries to avoid the predator by moving fast in the environment following the walls. The predator tries to chase the prey but the prey is faster than the predator so no capture occurs. In this tournament, the predator also has the strategy of waiting for the prey until it appears in its view field, and then attack (which in this case fails). Although this strategy was successful in a number of tournaments, this strategy was rarely seen in the overall evolutionary process.

preferring the former. The predator, on the other hand, in both experiments preferred a rather narrow view angle with a relative long view range. Unlike the prey it did not make the same trade-off between speed and vision, i.e. although speed was needed to chase the prey, vision was also needed for that task. Therefore, the predator evolved a balance between view angle and speed.

In sum, this paper has demonstrated the possibilities of allowing the evolutionary process to evolve appropriate morphologies suited for the robots' specific tasks. It has also demonstrated how different constraints can affect both the morphology and the behavior of the robots, and how the evolutionary process was able to make trade-offs, finding appropriate balance. Although these experiments definitely have limitations, e.g. concerning the possibilities of transfer to real robots, and only reflect on certain parts of evolving robot morphology, we still consider this work as a further step towards removing the human designer from the loop, suggesting a mixture of CCE and 'co-evolution' of brain and body.

References

1. Buason, G. (2002a). *Competitive co-evolution of sensory-motor systems*. Masters Dissertation HS-IDA-MD-02-004. Department of Computer Science, University of Skövde, Sweden.
2. Buason, G. (2002b). *Competitive co-evolution of sensory-motor systems - Appendix*. Technical Report HS-IDA-TR-02-004. Department of Computer Science, University of Skövde, Sweden.
3. Buason, G. & Ziemke, T. (in press). *Competitive Co-Evolution of Predator and Prey Sensory-Motor Systems*. In: *Second European Workshop on Evolutionary Robotics*. Springer Verlag, to appear.
4. Carlsson, J. & Ziemke, T. (2001). YAKS - Yet Another Khepera Simulator. In: Rückert, Sitte & Witkowski (eds.), *Autonomous minirobots for research and entertainment - Proceedings of the fifth international Heinz Nixdorf Symposium* (pp. 235–241). Paderborn, Germany: HNI-Verlagsschriftenreihe.
5. Cliff, D. & Miller, G. F. (1995). Tracking the Red Queen: Measurements of adaptive progress in co-evolutionary simulations. In: F. Moran, A. Moreano, J. J. Merelo, & P. Chacon, (eds.), *Advances in Artificial Life: Proceedings of the third european conference on Artificial Life*. Berlin: Springer-Verlag.
6. Cliff, D. & Miller, G. F. (1996). Co-evolution of pursuit and evasion II: Simulation methods and results. In: P. Maes, M. Mataric, J.-A. Meyer, J. Pollack & S. W. Wilson (eds.), *From animals to animats IV: Proceedings of the fourth international conference on simulation of adaptive behavior (SAB96)* (pp. 506–515). Cambridge, MA: MIT Press.
7. Floreano, D. & Nolfi, S. (1997a). God save the Red Queen! Competition in co-evolutionary robotics. In: J. R. Koza, D. Kalyanmoy, M. Dorigo, D. B. Fogel, M. Garzon, H. Iba, & R. L. Riolo (eds.), *Genetic programming 1997: Proceedings of the second annual conference*. San Francisco, CA: Morgan Kaufmann.
8. Floreano, D. & Nolfi, S. (1997b). Adaptive behavior in competing co-evolving species. In P. Husbands, & I. Harvey (eds.), *Proceedings of the fourth European Conference on Artificial Life*. Cambridge, MA: MIT Press.
9. Floreano, D., Nolfi, S. & Mondada, F. (1998). Competitive co-evolutionary robotics: From theory to practice. In: R. Pfeifer, B. Blumberg, J.-A. Meyer, & S. W. Wilson (eds.), *From animals to animats V: Proceedings of the fifth international conference on simulation of adaptive behavior*. Cambridge, MA: MIT Press.

10. Lee, W-P, Hallam, J. & Lund, H.H. (1996). A hybrid GP/GA Approach for co-evolving controllers and robot bodies to achieve fitness-specified tasks. In: *Proceedings of IEEE third international conference on evolutionary computation* (pp. 384–389). New York: IEEE Press.
11. Lund, H., Hallam, J. & Lee, W. (1997). Evolving robot morphology. In: IEEE International Conference on Evolutionary Computation (ed.), *Proceedings of IEEE fourth international conference on evolutionary computation* (pp. 197–202). New York: IEEE Press.
12. Nolfi, S. & Floreano, D. (1998). Co-evolving predator and prey robots: Do ‘arms races’ arise in artificial evolution? *Artificial Life*, 4, 311–335.
13. Nolfi, S. & Floreano, D. (2000). *Evolutionary robotics: The biology, intelligence, and technology of self-organizing machines*. Cambridge, MA: MIT Press.
14. Nolfi, S. & Floreano, D. (2002). Synthesis of autonomous robots through artificial evolution. *Trends in Cognitive Sciences*, 6, 31–37.
15. Pollack, J. B., Lipson, H., Hornby, G. & Funes, P. (2001). Three generations of automatically designed robots. *Artificial Life*, 7, 215–223.
16. Sharkey, N. E. & Heemskerk, J. N. H. (1997). The neural mind and the robot. In: Browne, A. (ed.), *Neural network perspectives on cognition and adaptive robotics* (pp. 169–194). Institute of Physics Publishing, Bristol, UK.

Integration of Genetic Programming and Reinforcement Learning for Real Robots

Shotaro Kamio, Hideyuki Mitsuhashi, and Hitoshi Iba

Graduate School of Frontier Science, The University of Tokyo,
7-3-1 Hongo, Bunkyo-ku, Tokyo, 113-8656, Japan.
{kamio,mituhasi,iba}@miv.t.u-tokyo.ac.jp

Abstract. We propose an integrated technique of genetic programming (GP) and reinforcement learning (RL) that allows a real robot to execute real-time learning. Our technique does not need a precise simulator because learning is done with a real robot. Moreover, our technique makes it possible to learn optimal actions in real robots. We show the result of an experiment with a real robot AIBO and represents the result which proves proposed technique performs better than traditional Q-learning method.

1 Introduction

When executing tasks by autonomous robots, we can make the robot learn what to do so as to complete the task from interactions with its environment but not manually pre-program for all situations. We know that such learning techniques as genetic programming (GP)[1] and reinforcement learning (RL)[2] work as means for automatically generating robot programs.

When applying GP, we should repeatedly evaluate many individuals over several generations. Therefore, it is difficult to apply GP to problems that requires too much time for evaluations of individuals. That is why we find very few previous studies on learning with a real robot.

To obtain optimal actions using RL, it is necessary to repeat learning trials time after time. The huge amount of learning time required presents a great problem when using a real robot. Accordingly, most studies deal with the problems of receiving an immediate reward from an action as shown in [3], or loading the results learned with a simulator into a real robot as shown in [4,5].

Although it is generally accepted to learn with a simulator and apply the result to a real robot, there are many tasks that are difficult to make a precise simulator. Applying these methods with an imprecise simulator could result in creating programs which may function optimally on the simulator but cannot provide optimal actions with a real robot. Furthermore, the operating characteristics of a real robot show certain variations due to minor errors in the manufacturing process or to changes with time. We cannot cope with such differences of robots only using a simulator.

Learning process with a real robot is surely necessary, therefore, for it to acquire optimal actions. Moreover, learning with a real robot sometimes makes

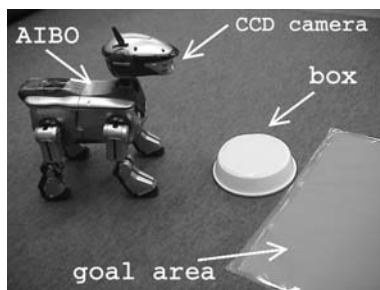


Fig. 1. The robot AIBO, the box and the goal area.

possible to learn even hardware and environmental characteristics, thus allowing the robot to acquire unexpected actions.

To solve the above difficulties, we propose a technique that allows a real robot to execute real-time learning in which GP and RL are integrated. Our proposed technique does not need a precise simulator because learning is done with a real robot. As a result of this idea, we can greatly reduce the cost to make the simulator much precise and acquire the program which acts optimally in the real robot.

The main contributions of this paper are summarized as follows:

1. We propose an integrated method of GP and RL.
2. We give empirical results to show how our approach works well for real-robot learning.
3. We conduct comparative experiments with traditional Q-learning to show the superiority of our method.

The next section gives the definition of a task in this study. After that, Section 3 explains our proposed technique and Section 4 presents experimental results with a real robot. Section 5 provides the result of comparison and future researches. Finally, a conclusion is given.

2 Task Definition

We used an “AIBO ERS-220” (Fig. 1) robot sold by SONY as the real robot in this experiment. AIBO’s development environment is freely available for non-commercial use and we can program with C++ language on it [6]. An AIBO has a CCD camera on its head, and moreover, an AIBO is equipped with image processor. It is able to easily recognize objects of specified colors on a CCD image at high speed.

The task in this experiment was to carry a box to a goal area. One of the difficulties of this task is that the robot has four legs. As a result, when the robot moves ahead, we see cases where the box sometimes is moved ahead or deviates from side to side, depending on the physical relationship between the box and AIBO legs. It is extremely difficult, in fact, to create a precise simulator that accurately expresses this box movements.

3 Proposed Technique

In this paper, we propose a technique that integrates GP and RL. As can be seen in Fig. 2(a), RL as individual learning is outside of the GP loop in the proposed technique. This technique enables us (1) to speed up learning in real robot and (2) to cope with the differences between a simulator and a real robot.

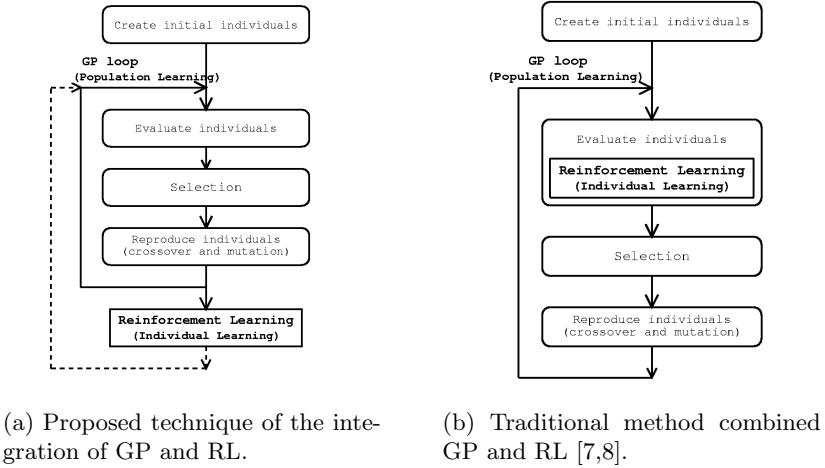


Fig. 2. The flow of the algorithm.

The proposed technique consists of two stages (GP part and RL part).

1. Carry out GP on a simplified simulator, and formulate programs that have the standards for robot actions required for executing a task.
2. Conduct individual learning (= RL) after loading the programs obtained in Step 1 above.

In the first step above, the programs that have the standards for the actions required of a real robot to execute a task are created through the GP process. The learning process of RL can be speeded up in the second step because the state space is divided into partial spaces under the judgment standards obtained in the first step. Moreover, preliminary learning with a simulator allows us to anticipate that a robot performs target-oriented actions from the beginning of the second stage. We used Q-learning as RL method in this study.

Although the process expressed by the external dotted line in Fig.2(a) was not realized in this study, it is a feedback loop. We consider that the parameters in a real environment that have been acquired via individual learning should ideally be fed back through this loop route.

The comparison with the traditional method (Fig. 2(b)) is discussed later in Sect. 5.2.

3.1 RL Part Conducted on the Real Robot

Action set. We prepared six selectable robot actions (move forward, retreat, turn left, turn right, retreat + turn left, and retreat + turn right). These actions are far from ideal ones: e.g. “move forward” action is not only to move the robot straightly forward but also has some deviations from side to side and “turn left” action is not only to turn left but also move the robot a little bit forward. The robot has to learn these characteristics of actions.

Every action takes approximately four seconds and eight seconds including the swinging of the head described below. It is, therefore, advisable that the learning time is as short as possible.

State Space. The state space was structured based on positions from where the box and the goal area can be seen in the CCD image, as described in [4]. The viewing angle of AIBO CCD is so narrow that the box or the goal area cannot be seen well with only one-directional images, in most cases. To avoid this difficulty, we added a mechanism to compensate for the surrounding images by swinging AIBO’s head so that state recognition can be conducted by the head swinging after each action. This head swinging operation was always uniformly given throughout the experiment as it was not an element to be learned in this study.

Figure 3 is the projection of the box state on the ground surface. The “near center” position is where the box fits into the two front legs. The box can be moved if the robot pushes it forward in this state. The box remains same position “near center” after the robot turns left or right in this state because the robot holds the box between two front legs. The state with the box not being in view was defined as “lost”; the state with the box not being in view and one preceding step at the left was defined as “lost into left” and, similarly, “lost into right” was defined.

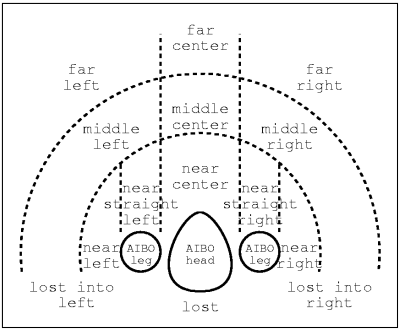


Fig. 3. States in real robot for the box. The front of the robot is upside of this figure.

We should pay special attention to the position of legs. Depending on the physical relationship between the box and AIBO legs, the movement of the box varies from moving forward to deviating from side to side. If an appropriate state

space is not defined, the Markov property of the environment, which is a premise of RL, cannot be met, thereby optimal actions cannot be found. Therefore, we defined “near straight left” and “near straight right” states at the frontal positions of the front legs.

We thus defined 14 states for the box. We similarly defined states of the goal area except that “near straight left” and “near straight right” states do not exist in them. There are 14 states for the box and 12 for the goal area; hence, this environment has states of their product, i.e., 168 states totally.

3.2 GP Part Conducted on the Simulated Robot

Simulator. The simulator in our experiment uses a robot expressed in circle on a two-dimensional plane, a box, and a goal area fixed on a plane. The task is completed when the robot pushes the box forward and overlaps the goal area on this plane.

We defined three actions (move forward, turn left, turn right) as action set and defined the state space in the simulator which is the simplified state space used for a real robot as Fig. 4. While actions of the real robot are not ideal ones, these actions in the simulator are ideal ones.

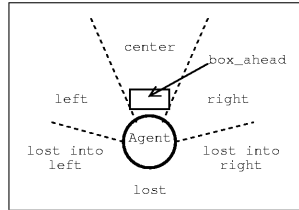


Fig. 4. States for box and goal area in the simulator. The area `box_ahead` is not the state but the place where `if_box_ahead` executes first argument.

Such actions and a state division is similar to that of a real robot, but is not exactly the same. In addition, physical parameters such as box weight and friction were not measured nor was the shape of the robot taken into account. Therefore, this simulator is very simple and it is possible to build it in low cost.

The two transfer characteristics of the box expressed by the simulator are the following.

1. The box moves forward if the box comes in contact with the front of the robot when the robot goes ahead¹.
2. After rotation, the box is near the center of the robot if the box is near the center of the robot when the robot turns².

¹ This corresponds to the situation that real robot pushes the box forward.

² This corresponds to the situation in which the box is placed between the front legs of a real robot when it is turning.

Settings of GP. The terminals and functions used in GP were as follows:

Terminal set: `move_forward`, `turn_left`, `turn_right`

Function set: `if_box_ahead`, `box_where`, `goal_where`, `prog2`

The terminal nodes above respectively correspond to the “move forward”, “turn left”, and “turn right” actions in the simulator. The functional nodes `box_where` and `goal_where` are the functions of six arguments, and they execute one of the six arguments, depending on the states (Fig. 4) of the box and the goal area as seen by the robot’s eyes. The function `if_box_ahead` which has two arguments executes the first argument if the box is positioned at “box ahead” position in Fig. 4. We arranged conditions so that only the `box_where` or the `goal_where` node becomes the head node of a gene of GP. The gene of GP is set to start executing from the head node and the execution is repeated again from the head node if the execution runs over the last leaf node until reaches maximum steps.

A trial starts with the state in which the robot and the box are randomly placed at the initial positions, and ends when the box is placed in the goal area or after a predetermined number of actions are performed by the robot. The following fitness values are allocated to the actions performed in a trial.

- If the task is completed:

$$f_{goal} = 100$$

$$f_{remaining\_moves} = 10 \times \left(0.5 - \frac{(\text{Number of moves})}{(\text{Maximum limit of number of moves})} \right)$$

$$f_{remaining\_turns} = 10 \times \left(0.5 - \frac{(\text{Number of turns})}{(\text{Maximum limit of number of turns})} \right)$$

- If the box is moved at least once: $f_{move} = 10$
- If the robot faces the box at least once: $f_{see\_box} = 1$
- If the robot faces the goal at least once: $f_{see\_goal} = 1$
- $f_{lost} = -\frac{(\text{Number of times having lost sight of the box})}{(\text{Number of steps})}$

The sum of the above figures indicates a fitness value for the i -th trial in an evaluation, or $fitness_i$.

To make robot acquire robust actions that do not depend on the initial position, the average values of 100 trials in which the initial position is randomly changed was taken when calculating the fitness of individuals. The fitness of individuals is calculated by the following equation.

$$fitness = \frac{1}{100} \sum_{i=0}^{99} fitness_i + 2.0 \cdot \frac{(\text{Maximum gene length}) - (\text{Gene length})}{(\text{Maximum gene length})} \quad (1)$$

The second term of the right side of this equation has the meaning that a penalty is given to a longer gene length.

Using the fitness function determined above, learning was executed for 1,000 individuals of 50 generations with maximum gene length = 150. Learning costs about 10 minutes on the Linux system equipped with an Athlon XP 1800+.

We finally applied the individuals that had proven to have the best performance to learning with a real robot.

Table 1. Action nodes and their selectable real actions.

action node	real actions which Q -table can select.
<code>move_forward</code>	"move forward"*, "retreat + turn left", "retreat + turn right"
<code>turn_left</code>	"turn left"*, "retreat + turn left", "retreat"
<code>turn_right</code>	"turn right"*, "retreat + turn right", "retreat"

* The action which Q -table prefers to select with a biased initial value.

3.3 Integration of GP and RL

Q -learning is executed to adapt actions acquired via GP to the operating characteristics of a real robot. This is aimed at revising the `move_forward`, `turn_left` and `turn_right` actions with the simulator to their optimal actions in a real world.

We allocated a Q -table, on which Q -values were listed, to each of the `move_forward`, `turn_left` and `turn_right` action nodes. The states on the Q -tables are regarded as those for a real robot. Therefore, actual actions selected with Q -tables can vary depending on the state, even if the same action nodes are executed by a real robot. Figure 5 illustrates the above situation. The states "near straight left" and "near straight right", which exist only in a real robot, are translated into a "center" state in function nodes of GP.

Each Q -table is arranged to set the limits of selectable actions. This refers to the idea that, for example, "turn right" actions are not necessary to learn in the `turn_left` node. In this study, we defined three selectable robot actions for each action node as Table 1. With this technique, each Q -table was initialized with a biased initial value³. The initial value of 0.0001 was entered into the respective Q -tables so that preferred actions were selected for each Q -table, while 0.0 was entered for other actions. The actions which are preferred to select on each action node are described in Table 1.

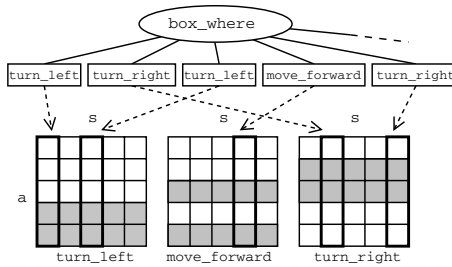


Fig. 5. Action nodes pick up a real action according to the Q -value of a real robot's state.

The total size of the three Q -tables is 1.5 times that of ordinary Q -learning. Theoretically, convergence with the optimal solution is considered to require

³ According to the theory, we can initialize Q -values with arbitrary values, and Q -values converge with the optimum solution regardless of the initial value [2].

more time than ordinary Q-learning. However, the performance of this technique while programs are executed is relatively good. This is because all the states on the Q -table are not necessarily used as the robot performs actions according to the programs obtained via GP and the task-based actions are available after the Q-learning starts.

The “state-action deviation” problem should be taken into account when executing Q-learning with the state constructed from a visual image [4]. This is the problem that optimal actions cannot be achieved due to the dispersion of state transitions because the state composed only of the images remains the same without clearly distinguishing differences in image values. To avoid this problem, we redefined “changes” in states. The redefinition is that the current state is unchanged if the terminal node executed in the program remains the same and so does the executing state of a real robot⁴. Until the current state changes, the Q -value is not updated and the same action is repeated.

As for parameters for Q-learning, the reward was set at 1.0 when the goal is achieved and 0.0 for other states. We set the parameters as the learning rate $\alpha = 0.3$ and the discount factor $\gamma = 0.9$.

4 Experimental Results with AIBO

Just after starting learning: The robot succeeded in completing the task when Q-learning with a real robot started using this technique. This was because the robot could perform actions by taking advantage of the results learned via GP.

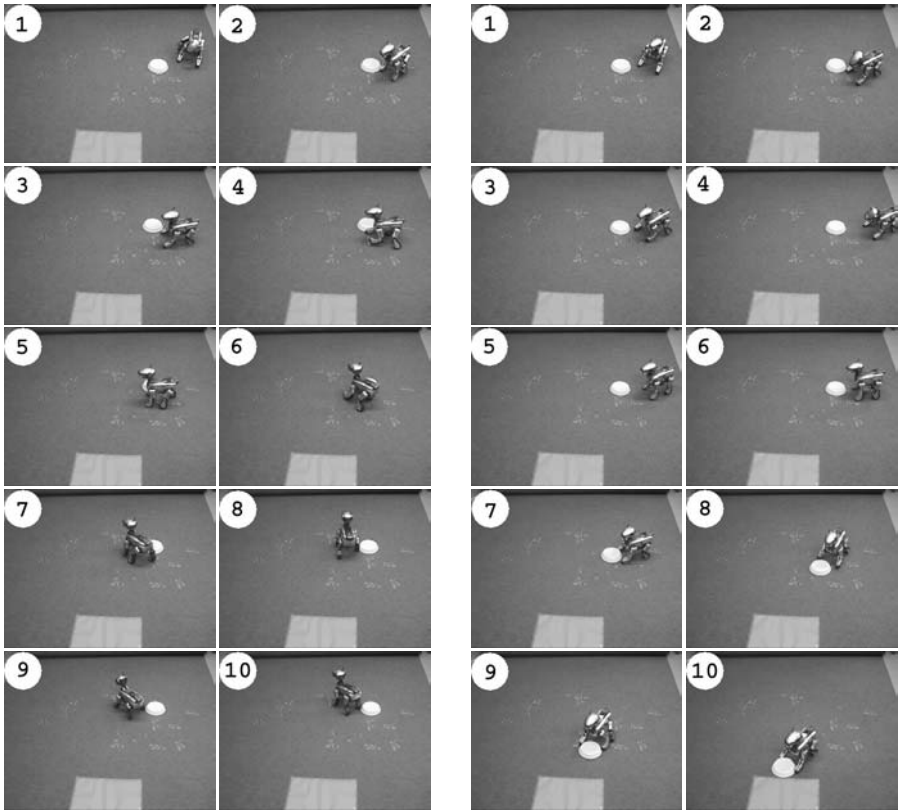
At the situation in which the box was placed near the center of the robot along with robot movements, the robot always achieved the task with regard to all the states tried. Whereas, if the box was not placed near the center of the robot after its displacement (e.g. if the box was slightly outside the legs), the robot sometimes failed to move the box properly. The robot repeatedly turned right to face the box, but continued vain movements going around the box because it did not have a small turning circle, unlike the actions in the simulator. Figure 6(a) shows typical series of actions. In some situation, the robot turned right but could not face the box and lost it in view (at the last of Fig. 6(a)).

This typical example proves that optimal actions with the simulator are not always optimal in a real environment. This is because of differences between the simulator and the real robot.

After ten hours (after about 4000 steps): We observed optimal actions as Fig. 6(b). The robot selected “retreat” or “retreat + turn” action in the situations in which it could not complete the task at the beginning of Q-learning. As a result, the robot could face the box and pushed the box forward to the goal, and finally completed the task.

Learning effects were found in other point, too. As the robot approached the box smoothly, the number of occurrence of “lost” was reduced. This means the robot acts more efficiently than the beginning of learning.

⁴ We modified Asada et al.’s definition [4] in order to deal with several Q -tables.



(a) Failed actions losing the box at the beginning of learning.

(b) Successful actions after 10-hour learning.

Fig. 6. Typical series of actions.

5 Discussion

5.1 Comparison with Q-Learning in Both Simulator and Real Robot

We compared our proposed technique with the method of Q-learning which learns in a simulator and re-learns in a real world (we call this method as RL+RL in this section). For Q-learning in the simulator, we introduced the qualitative distance (“far”, “middle”, and “near”) so that the state space could be similar to the one for the real robot⁵.

For this comparison, we selected ten situations which are difficult to complete at the beginning of Q-learning because of the gap between the simulation and

⁵ This simulator has 12 states for each of the box and the goal area; hence, this environment has 144 states.

Table 2. Comparison of proposed technique (GP+RL) with Q-learning (RL+RL).

#. situation	GP+RL			RL+RL		
	avg. steps	lost box	lost goal	avg. steps	lost box	lost goal
1	19.6	0	1	20.0	0	1
2	14.7	0	0	53.0	2	2
3	24.0	0	1	26.7	0	1
4	10.3	0	0	11.0	0	0
5	21.6	0	0	88.0	3	3
6	13.5	0	0	10.5	0	0
7	26.7	0	1	26.0	0	1
8	23.0	0	1	13.0	0	0
9	21.5	0	0	10.5	0	0
10	13.5	0	0	29.0	0	1

the real robot. We measured action efficiency after ten-hour Q-learning for these ten situations. These tests are executed in a greedy policy in order that the robot always selects the best action in each state.

Table 2 shows the result of both methods, i.e., proposed technique (GP+RL) and Q-learning method (RL+RL). This table represents the average number of steps to complete the task and the number of occurrences when the robot has lost the box or the goal area in completing the task.

While RL+RL performed better than the proposed technique in four situations on the average of the steps, the proposed technique performed much better than RL+RL in other six situations (bold font in Table 2). Moreover, the robot evolved by the proposed technique less often lost the box and the goal area than that by RL+RL. This result proves that our proposed technique learned more efficient actions than RL+RL method.

Figure 7 shows the changes in Q -values when they are updated in Q-learning with the real robot. The absolute value of the Q -value change represents how far the Q -value is from the optimal one. According to Fig. 7, large changes occurred to RL+RL method more frequently than to our technique. This may be because RL+RL has to re-learn optimal Q -values starting from the ones which have already been learned with the simulator. Therefore, we can conclude that RL+RL requires more time to converge to optimal Q -values.

5.2 Related Works

There are many studies combined evolutionary algorithms and RL [9,10]. Although the approaches differ from our proposed technique, we see several studies in which GP and RL are combined [7,8]. With these traditional techniques, Q-learning is adopted as a RL, and individuals of GP represents the structure of the state space to be searched. It is reported that searching efficiency is improved in QGP, compared to traditional Q-learning [7].

However, the techniques used in these studies are also a kind of population learning using numerous individuals. RL must be executed for numerous individuals in the population because RL is inside the GP loop, as shown in Fig. 2(b). A huge amount of time would become necessary for learning if all the processes

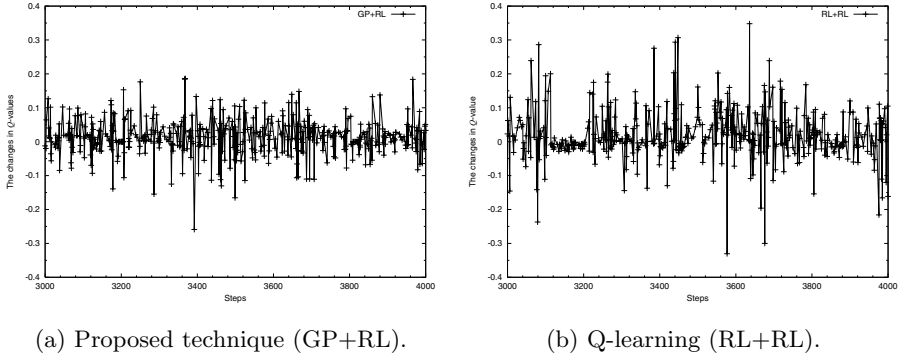


Fig. 7. Comparison of changes in Q -values after about 8-hour to 10-hour Q-learning with a real robot.

are directly applied to a real robot. As a result, no studies using any of these techniques with a real robot have been reported.

Several studies on RL pursue the use of hierarchical state space to enable us to deal with complicated tasks [11,12]. The hierarchical state spaces in such studies is structured manually in advance. It is generally considered difficult to automatically build the hierarchical structure only through RL. We can consider the programs automatically generated in GP of proposed technique represents the hierarchical structure of state space which is manually structured in [12].

Noises in simulators are often effective to overcome the differences between a simulator and real environment [13]. However, the robot learned with our technique showed sufficient performance in noisy real environment, while it learned in ideal simulator. One of the reasons is that the coarse state division absorbs the image processing noise. We plan to perform a comparison the robustness produced by our technique with that by noisy simulators.

5.3 Future Researches

We used only several discrete actions in this study. Although this is simple, continuous actions are more realistic in applications. In that situation, for example, “turn left in 30.0 degrees” in the beginning of RL can be changed to “turn left in 31.5 degrees” after learning, depending on the operating characteristics of the robot. We plan to conduct an experiment with such continuous actions.

We intend to apply the technique to more complicated tasks such as the multi-agent problem and other real-robot learning. Based on our method, it can be possible to use almost the same simulator and settings of RL as described in this paper. Experiments will be conducted with various robots, e.g., a humanoid robot “HOAP-1” (manufactured by Fujitsu Automation Limited) or “Khepera”. The preliminary results were reported in [14]. We are in pursuit of the applicability of the proposed approach to this wide research area.

6 Conclusion

In this paper, we proposed a technique for executing real-time learning with a real robot based on an integration of GP and RL techniques, and verified its effectiveness experimentally.

At the initial stage of Q-learning, we sometimes observed unsuccessful displacements of the box due to a lack of data concerning real robot characteristics, which had not been reproduced by a simulator. The technique, however, was adapted to the operating characteristics of the real robot through the ten hour learning period. This proves that the step of individual learning in this technique performed effectively in our experiment.

This technique, however, still has several points to be improved. One is feeding back data from learning in a real environment to GP and the simulator, which corresponds to the loop represented by the dotted line in Fig.2(a). This may enable us to improve simulator precision automatically in learning. Its realization is one of the future issues.

References

1. John R. Koza: Genetic Programming, On the Programming of Computers by means of Natural Selection. MIT Press (1992)
2. Richard S. Sutton and Andrew G. Barto: Reinforcement Learning: An introduction. MIT Press in Cambridge, MA (1998)
3. Hajime Kimura, Toru Yamashita and Shigenobu Kobayashi: Reinforcement Learning of Walking Behavior for a Four-Legged Robot. In: 40th IEEE Conference on Decision and Control. (2001)
4. Minoru Asada, Shoichi Noda, Sukoya Tawaratsumida and Koh Hosoda: Purposive Behavior Acquisition for a Real Robot by Vision-Based Reinforcement Learning. *Machine Learning* **23** (1996) 279–303
5. Yasutake Takahashi, Minoru Asada, Shoichi Noda and Koh Hosoda: Sensor Space Segmentation for Mobile Robot Learning. In: Proceedings of ICMAS'96 Workshop on Learning, Interaction and Organizations in Multiagent Environment. (1996)
6. OPEN-R Programming Special Interest Group: Introduction to OPEN-R programming (in Japanese). Impress corporation (2002)
7. Hitoshi Iba: Multi-Agent Reinforcement Learning with Genetic Programming. In: Proc. of the Third Annual Genetic Programming Conference. (1998)
8. Keith L. Downing: Adaptive genetic programs via reinforcement learning. In: Proc. of the Third Annual Genetic Programming Conference. (1998)
9. Moriarty, D.E., Schultz, A.C., Grefenstette, J.J.: Evolutionary algorithms for reinforcement learning. *Journal of Artificial Intelligence Research* **11** (1999) 199–229
10. Dorigo, M., Colombetti, M.: Robot Shaping: An Experiment in Behavior Engineering. MIT Press (1998)
11. L.P. Kaelbling: Hierarchical Learning in Stochastic Domains: preliminary Results. In: Proc. 10th Int. Conf. on Machine Learning. (1993) 167–173
12. T.G. Dietterich: Hierarchical reinforcement learning with the MAXQ value function decomposition. *Journal of Artificial Intelligence Research* **13:227–303** (2000)

13. Schultz, A.C., Ramsey, C.L., Grefenstette, J.J.: Simulation-assisted learning by competition: Effects of noise differences between training model and target environment. In: Proc. of Seventh International Conference on Machine Learning, San Mateo, Morgan Kaufmann (1990) 211–215
14. Kohsuke Yanai and Hitoshi Iba: Multi-agent Robot Learning by Means of Genetic Programming: Solving an Escape Problem. In Liu, Y., et al., eds.: *Evolvable Systems: From Biology to Hardware*. Proceedings of the 4th International Conference on Evolvable Systems, ICES'2001, Tokyo, October 3-5, 2001, Springer-Verlag, Berlin, Heidelberg (2001) 192–203

Multi-objectivity as a Tool for Constructing Hierarchical Complexity

Jason Teo, Minh Ha Nguyen, and Hussein A. Abbass

Artificial Life and Adaptive Robotics (A.L.A.R.) Lab,
School of Computer Science, University of New South Wales,
Australian Defence Force Academy Campus, Canberra, Australia.
{j.teo,m.nguyen,h.abbass}@adfa.edu.au

Abstract. This paper presents a novel perspective to the use of multi-objective optimization and in particular evolutionary *multi-objective optimization* (EMO) as a measure of complexity. We show that the partial order feature that is being inherited in the Pareto concept exhibits characteristics which are suitable for studying and measuring the complexities of embodied organisms. We also show that multi-objectivity provides a suitable methodology for investigating complexity in artificially evolved creatures. Moreover, we present a first attempt at quantifying the morphological complexity of quadruped and hexapod robots as well as their locomotion behaviors.

1 Introduction

The study of complex systems has attracted much interest over the last decade and a half. However, the definition of what makes a system complex is still the subject of much debate among researchers [7,19]. There are numerous methods available in the literature for measuring complexity. However, it has been argued that complexity measures are typically too difficult to compute to be of use for any practical purpose or intent [16]. What we are proposing in this paper is a simple and highly accessible methodology for characterizing the complexity of artificially evolved creatures using a multi-objective methodology. This work poses evolutionary multi-objective optimization (EMO) [5] as a convenient platform which researchers can utilize practically in attempting to define, measure or simply characterize the complexity of everyday problems in a useful and purposeful manner.

2 Embodied Cognition and Organisms

The view of intelligence in traditional AI and cognitive science has been that of an agent undertaking some form of information processing within an abstracted representation of the world. This form of understanding intelligence was found to be flawed in that the agent's cognitive abilities were derived purely from a processing unit that manipulates symbols and representations far abstracted from

the agent's real environment [3]. Conversely, the embodied cognitive view considers intelligence as a phenomenon that emerges independently from the parallel and dynamical interactions between an embodied organism and its environment [14]. Such artificial creatures possess two important qualities: embodiment and situatedness.

A subfield of research into embodied cognition involves the use of artificial evolution for automatically generating the morphology and mind of embodied creatures [18]. The term mind as used in this context of research is synonymous with brain and controller - it merely reflects the processing unit that acts to transform the sensory inputs into the motor outputs of the artificial creature. The automatic synthesis of such embodied and situated creatures through artificial evolution has become a key area of research not only in the cognitive sciences but also in robotics [15], artificial life [14], and evolutionary computation [2,10].

Consequently, there has been much research interest in evolving both physically-simulated virtual organisms [2,10,14] and real physical robots [15,8,12]. The main objective of these studies is to evolve increasingly complex behaviors and/or morphologies either through evolutionary or lifetime learning. Needless to say, the term "*complex*" is generally used very loosely since there is currently no general method for comparing between the complexities of these evolved artificial creatures' behaviors and morphologies. As such, without a quantitative measure for behavioral or morphological complexity, an objective evaluation between these artificial evolutionary systems becomes very hard and typically ends up being some sort of subjective argument.

There are generally two widely-accepted views of measuring complexity. The first is an information-theoretic approach based on Shannon's entropy [17] and is commonly referred to as statistical complexity. The entropy $H(X)$ of a random variable X , where the outcomes x_i occur with probability p_i , is given by

$$H(X) = -C \sum_i^N p_i \log p_i \quad (1)$$

where C is the constant related to the base chosen to express the logarithm. Entropy is a measure of disorder present in a system and thus gives us an indication of how much we do not know about a particular system's structure. Shannon's entropy measures the amount of information content present within a given message or more generally any system of interest. Thus a more complex system would be expected to give a much higher information content than a less complex system. In other words, a more complex system would require more bits to describe compared to a less complex system. In this context, a sequence of random numbers will lead to the highest entropy and consequently to the lowest information content. In this sense, complexity is somehow a measure of order or disorder.

A computation-theoretic approach to measuring complexity is based on Kolmogorov's application of universal Turing machines [11] and is commonly known as Kolmogorov complexity. It is concerned with finding the shortest possible computer program or any abstract automaton that is capable of reproducing a given string. The Kolmogorov complexity $K(s)$ of a string s is given by

$$K(s) = \min\{|p| : s = C_T(p)\} \quad (2)$$

where $|p|$ represents the length of program p and $C_T(p)$ represents the result of running program p on Turing machine T . A more complex string would thus require a longer program while a simpler string would require a much shorter program. In essence, the complexity of a particular system is measured by the amount of computation required to recreate the system in question.

3 Complexity in the Eyes of the Beholder

None of the previous measures are sufficient to measure the complexity of embodied systems. As such, we need first to provide a critical view of these measures and why they stand shorthanded in terms of embodied systems.

Take for example a simple behavior such as walking. Let us assume that we are interested in measuring the complexity of walking in different environments and the walking itself is undertaken by an artificial neural network. From Shannon's perspective, the complexity can be measured using the entropy of the data structure holding the neural network. Obviously a drawback for this view is its ignorance of the context and the concepts of embodiment and situatedness. The complexity of walking on a flat landscape is entirely different from walking on a rough landscape. Two neural networks may be represented using the same number of bits but exhibit entirely different behaviors.

Now, let us take another example which will show the limitations of Kolmogorov complexity. Assume we have a sequence of random numbers. Obviously the shortest program which is able to reproduce this sequence is the sequence itself. In other words, a known drawback for Kolmogorov complexity is that it has the highest level of complexity when the system is random. In addition, let us re-visit the neural network example. Assume that the robot is not using a fixed neural network but some form of evolvable hardware (which may be an evolutionary neural network). If the fitness landscape for the problem at hand is monotonically increasing, a hill climber will simply be the shortest program which guarantees to re-produce the behavior. However, if the landscape is rugged, reproducing the behavior is only achievable if we know the seed; otherwise, the problem will require complete enumeration to recreate the behavior.

In this paper, we propose a generic definition for complexity using the multi-objective paradigm. However, before we proceed with our definition, we first remind the reader of the concept of partial order.

Definition 1: Partial and Lexicographic Order Assume the two sets A and B . Assume the l -subsets over A and B such that $A = \{a_1 < \dots < a_l\}$ and $B = \{b_1 < \dots < b_l\}$.
 A partial order is defined as $A \leq_j B$ if $a_j \leq b_j, \forall j \in \{1, \dots, l\}$
 A lexicographic order is defined as $A <_j B$ if $\exists a_k < b_k$ and $a_j = b_j, j < k, \forall j, k \in \{1, \dots, l\}$

In other words, a lexicographic order is a total order. In multi-objective optimization, the concept of Pareto optimality is normally used. A solution x belongs

to the Pareto set if there is not a solution y in the feasible solution set such that y dominates x (*ie.* x has to be at least as good as y when measured on all objectives and better than y on at least one objective). The Pareto concept thus forms partial orders in the objective space.

Let us recall the embodied cognition problem. The problem is to study the relationship between the behavior, controller, environment, learning algorithm, and morphology. A typical question that one may ask is what is the optimal behavior for a given morphology, controller, learning algorithm and environment. We can formally represent the problem of embodied cognition as the five sets B , C , E , L , and M for the five spaces of behavior, controller, environment, learning algorithm, and morphology respectively. Here, we need to differentiate between the robot behavior B and the desired behavior $\hat{B}$. The former can be seen as the actual value of the fitness function and the latter can be seen as the real maximum of the fitness function. For example, if the desired behavior (task) is to maximize the locomotion distance, then the global maximum of this function is the desired behavior, whereas the distance achieved by the robot (what the robot is actually doing) is the actual behavior. In traditional robotics, the problem can be seen as Given the desired behavior $\hat{B}$, find L which optimizes C subject to $E \cup M$. In psychology, the problem can be formulated as Given C , E , L and M , study the characteristics of the set B . In co-evolving morphology and mind, the problem is Given the desired behavior $\hat{B}$ and L , optimize C and M subject to E . A general observation is that the learning algorithm is usually fixed during the experiments.

In asking a question such as “Is a human more complex than a Monkey?”, a natural question that follows would be “in what sense?”. Complexity is not a unique concept. It is usually defined or measured within some context. For example, a human can be seen as more complex than a Monkey if we are looking at the complexity of intelligence, whereas a Monkey can be seen as more complex than the human if we are looking at the number of different gaits the monkey has for locomotion. Therefore, what is important from an artificial life perspective is to establish the complexity hierarchy on different scales. Consequently, we introduce the following definition for complexity.

Definition 2: Complexity is a strict partial order relation.

According to this definition, we can establish an order of complexity between the system’s components/species. We can then compare the complexities of two species $S_1 = (B_1, C_1, E_1, L_1, M_1)$ and $S_2 = (B_2, C_2, E_2, L_2, M_2)$ as:

S_1 is at least as complex as S_2 with respect to concept Ψ iff

$S_2^\Psi = (B_2, C_2, E_2, L_2, M_2) \leq_j S_1^\Psi = (B_1, C_1, E_1, L_1, M_1), \forall j \in \{1, \dots, l\}$, **Given**

$$B_i = \{B_{i1} < \dots < B_{il}\}, C_i = \{C_{i1} < \dots < C_{il}\}, E_i = \{E_{i1} < \dots < E_{il}\},$$

$$L_i = \{L_{i1} < \dots < L_{il}\}, M_i = \{M_{i1} < \dots < M_{il}\}, i \in \{1, 2\} \quad (3)$$

where Ψ partitions the sets into l non-overlapping subsets.

We can even establish a complete order of complexity by using the lexicographic order as:

S_1 is more complex than S_2 with respect to concept Ψ iff

$$S_2^\Psi = (B_2, C_2, E_2, L_2, M_2) <_j S_1^\Psi = (B_1, C_1, E_1, L_1, M_1), \forall j \in \{1, \dots, l\}, \text{ Given} \\ B_i = \{B_{i1} < \dots < B_{il}\}, C_i = \{C_{i1} < \dots < C_{il}\}, E_i = \{E_{i1} < \dots < E_{il}\}, \\ L_i = \{L_{i1} < \dots < L_{il}\}, M_i = \{M_{i1} < \dots < M_{il}\}, i \in \{1, 2\} \quad (4)$$

The lexicographic order is not as flexible as partial order since the former requires a monotonic increase in complexity. The latter however, allows individuals to have similar levels of complexity; therefore, it is more suitable for defining hierarchies of complexity. Some of the characteristics in our definition of complexity here include

Irreflexive: The complexity definition satisfies irreflexivity; that is, x cannot be more complex than itself.

Asymmetric: The complexity definition satisfies asymmetry; that is, if x is more complex than y , then y cannot be more complex than x .

Transitive: The complexity definition satisfies transitivity; that is, if x is more complex than y and y is more complex than z , then x is more complex than z .

The concept of Pareto optimality is similar to the concept of partial order except that Pareto optimality is more strict in the sense that it does not satisfy reflexivity; that is, a solution cannot dominate itself; therefore it cannot exist as a Pareto optimal if there is a copy of it in the solution set. Usually, when we have copies of one solution, we take one of them; therefore this problem does not arise. As a result, we can assume here that Pareto optimality imposes a complexity hierarchy on the solution set.

The previous definition will simply order the sets based on their complexities according to some concept Ψ . However, they do not provide an exact quantitative measure for complexity. In the simple case, given the five sets B, C, E, L , and M ; assume the function f , which maps each element in each set to some value called the fitness, and assuming that C, E and L do not change, a simple measure of morphological change of complexity can be

$$\frac{\partial f(b)}{\partial m}, b \in B, m \in M \quad (5)$$

In other words, assuming that the environment, controller, and the learning algorithm are fixed, the change in morphological complexity can be measured in the eyes of the change in the fitness of the robot (actual behavior). The fitness will be defined later in the paper. Therefore, we introduce the following definition

Definition 3: Change of Complexity Value for the morphology is the rate of change in behavioral fitness when the morphology changes, given that both the environment, learning algorithm and controller are fixed.

The previous definition can be generalized to cover the controller and environment quite easily by simply replacing “morphology” by either “environment”, “learning algorithm”, or “controller”. Based on this definition, if we can come up with a good measure for behavioral complexity, we can use this measure to quantify the change in complexity for morphology, controller, learning algorithm, or environment. In the same manner, if we have a complexity measure for the controller, we can use it to quantify the change of complexity in the other four parameters. Therefore, we propose the notion of defining the complexity of one object as viewed from the perspective of another object. This is not unlike Emmeche’s idea of complexity as put in the eyes of the beholder [6]. However, we formalize and solidify this idea by putting it into practical and quantitative usage through the multi-objective approach. We will demonstrate that results from an EMO run of two conflicting objectives results in a Pareto-front that allows a comparison of the different aspects of an artificial creature’s complexity.

In the literature, there are a number of related topics which can help here. For example, the VC-dimension can be used as a complexity measure for the controller. A feed-forward neural network using a threshold activation function has a VC dimension of $O(W \log W)$ while a similar network with a sigmoid activation has a VC dimension of $O(W^2)$, where W is the number of free parameters in the network [9]. It is apparent from here that one can control the complexity of a network by minimizing the number of free parameters which can be done either by the minimization of the number of synapses or the number of hidden units. It is important to separate between the learning algorithm and the model itself. For example, two identical neural networks with fixed architectures may perform differently if one of them is trained using back-propagation while the other is trained using an evolutionary algorithm. In this case, the separation between the model and the algorithm helps us to isolate their individual effects and gain an understanding of their individual roles.

In this paper, we are essentially posing two questions, what is the change of (1) behavioral complexity and (2) morphological complexity of the artificial creature in the eyes of its controller. In other words, how complex is the behavior and morphology in terms of evolving a successful controller?

3.1 Assumptions

Two assumptions need to be made. First, the Pareto set obtained from evolution is considered to be the actual Pareto set. This means that for the creature on the Pareto set, the maximum amount of locomotion is achieved with the minimum number of hidden units in the ANN. We do note however that the evolved Pareto set in the experiments may not have converged to the optimal set. Nevertheless, it is not the objective of this paper to provide a method which guarantees convergence of EMO but rather to introduce and demonstrate the application of measuring complexity in the eyes of the beholder. It is important to mention that although this assumption may not hold, the results can still be valid. This will be the case when creatures are not on the actual Pareto-front

but the distances between them on the intermediate Pareto-front are similar to that of creatures on the actual Pareto-front.

The second assumption is there are no redundancies present in the ANN architectures of the evolved Pareto set. This simply means that all the input and output units as well as the synaptic connections between layers of the network are actually involved in and required for achieving the observed locomotion competency. We have investigated the amount of redundancy present in evolved ANN controllers and found that the self-adaptive Pareto EMO approach produces networks with practically zero redundancy.

4 Methods

4.1 The Virtual Robots and Simulation Environment

The Vortex physics simulation toolkit [4] was utilized to accurately simulate the physical properties, such as forces, torques, inertia, friction, restitution and damping, of and interactions between the robot and its environment. Two artificial creatures (Figure 1) were used in this study.

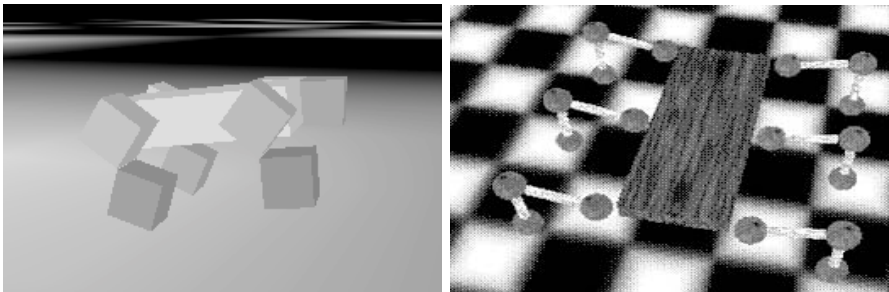


Fig. 1. The four-legged (quadruped) and six-legged (hexapod) creatures.

The first artificial creature is a quadruped with 4 short legs. Each leg consists of an upper limb connected to a lower limb via a hinge (1 degree-of-freedom (DOF)) joint and is in turn connected to the torso via another hinge joint. Each of the hinge joints is actuated by a motor that generates a torque producing rotation of the connected body parts about that hinge joint.

The second artificial creature is a hexapod with 6 long legs, which are connected to the torso by insect hip joints. Each insect hip joint consists of two hinges, making it a 2 DOF joint: one to control the back-and-forth swinging and another for the lifting of the leg. Each leg has an upper limb connected to a lower limb by a hinge (1 DOF) joint. The hinges are actuated by motors in the same fashion as in the first artificial creature.

The Pareto-frontier of our evolutionary runs are obtained from optimizing two conflicting objectives: (1) minimizing the number of hidden units used in

the ANN that act as the creature's controller and (2) maximizing horizontal locomotion distance of the artificial creature. What we obtain at the end of the runs are Pareto sets of ANNs that trade-off between number of hidden units and locomotion distance. The locomotion distances achieved by the different Pareto solutions will provide a common ground where locomotion competency can be used to compare different behaviors and morphologies. It will provide a set of ANNs with the smallest hidden layer capable of achieving a variety of locomotion competencies. The structural definition of the evolved ANNs can now be used as a measure of complexity for the different creature behaviors and morphologies.

The ANN architecture used in this study is a fully-connected feed-forward network with recurrent connections on the hidden units as well as direct input-output connections. Recurrent connections were included to allow the creature's controller to learn time-dependent dynamics of the system. Direct input-output connections were also included in the controller's architecture to allow for direct sensor-motor mappings to evolve that do not require hidden layer transformations. Bias is incorporated in the calculation of the activation of the hidden as well as output layers.

The Self-adaptive Pareto-frontier Differential Evolution algorithm (SPDE) [1] was used to drive the evolutionary optimization process. SPDE is an elitist approach to EMO where both crossover and mutation rates are self-adapted. Our chromosome is a class that contains one matrix Ω and one vector ρ . The matrix Ω is of dimension $(I + H) \times (H + O)$. Each element $\omega_{ij} \in \Omega$, is the weight connecting unit i with unit j , where $i = 0, \dots, (I - 1)$ is the input unit i , $i = I, \dots, (I + H - 1)$ is the hidden unit $(i - I)$, $j = 0, \dots, (H - 1)$ is the hidden unit j , and $j = H, \dots, (H + O - 1)$ is the output unit $(j - H)$.

The vector ρ is of dimension H , where $\rho_h \in \rho$ is a binary value used to indicate if hidden unit h exists in the network or not; that is, it works as a switch to turn a hidden unit on or off. Thus, the architecture of the ANN is variable in the hidden layer: any number of hidden units from 0 to H is permitted. The sum, $\sum_{h=0}^H \rho_h$, represents the actual number of hidden units in a network, where H is the maximum number of hidden units. The last two elements in the chromosome are the crossover rate δ and mutation rate η . This representation allows simultaneous training of the weights in the network and selecting a subset of hidden units as well as allowing for the self-adaptation of crossover and mutation rates during optimization.

4.2 Experimental Setup

Two series of experiments were conducted. Behavioral complexity was investigated in the first series of experiments and morphological complexity was investigated in the second. For both series of experiments, each evolutionary run was allowed to evolve over 1000 generations with a randomly initialized population size of 30. The maximum number of hidden units was fixed at 15 based on preliminary experimentation. The number of hidden units used and maximum locomotion achieved for each genotype evaluated as well as the Pareto set of

solutions obtained in every generation were recorded. The Pareto solutions obtained at the completion of the evolutionary process were compared to obtain a characterization of the behavioral and morphological complexity.

To investigate behavioral complexity in the eyes of the controller, the morphology was fixed by using only the quadruped creature but the desired behavior was varied by having two different fitness functions. The first fitness function measured only the maximum horizontal locomotion achieved but the second fitness function measured both maximum horizontal locomotion and static stability achieved. By static stability, we mean that the creature achieves a statically stable locomotion gait with at least three of its supporting legs touching the ground during each step of its movement. The two problems we have are:

(*P1*)

$$f_1 = d \quad (6)$$

$$f_2 = \sum_{h=0}^H \rho_h \quad (7)$$

(*P2*)

$$f_1 = d/20 + s/500 \quad (8)$$

$$f_2 = \sum_{h=0}^H \rho_h \quad (9)$$

where *P1* and *P2* are the two sets of objectives used. *d* refers to the locomotion distance achieved and *s* is the number of times the creature is statically stable as controlled by the ANN at the end of the evaluation period of 500 timesteps. *P1* is using the locomotion distance as the first objective while *P2* is using a linear combination of the locomotion distance and static stability. Minimizing the number of hidden units is the second objective in both problems.

To investigate morphological complexity, another set of 10 independent runs was carried out but this time using the hexapod creature. This is to enable a comparison with the quadruped creature which has a significantly different morphology in terms of its basic design. The *P1* set of objectives was used to keep the behavior fixed. The results obtained in this second series of experiments were then compared against the results obtained from the first series of experiments where the quadruped creature was used with the *P1* set of objective functions.

5 Results and Discussion

5.1 Morphological Complexity

We first present the results for the quadruped and hexapod evolved under *P1*. Figure 2 compares the Pareto optimal solutions obtained for the two different morphologies over 10 runs. Here we are fixing *E* and *L*; therefore, we can either measure the change of morphological complexity in the eyes of the behavior or the controller; that is, $\frac{\delta f(B)}{\delta M}$ or $\frac{\delta f(C)}{\delta M}$ respectively. If we fix the actual behavior *B* as the locomotion competency of achieving a movement of $13 < d < 15$,

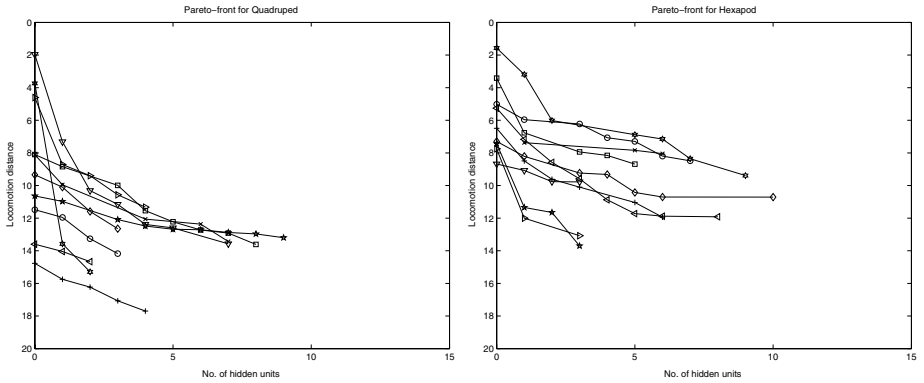


Fig. 2. Pareto-frontier of controllers obtained from 10 runs using the quadruped and hexapod with the $P1$ set of objectives.

then the change in the controller $\delta f(C)$ is measured according to the number of hidden units used in the ANN. At this point of comparison, we find that the quadruped is able to achieve the desired behavior with 0 hidden units whereas the hexapod required 3 hidden units. In terms of the ANN architecture, the quadruped achieved the required level of locomotion competency without using the hidden layer at all, that it relied solely on direct input-output connections as in a perceptron. This phenomenon has been previously observed to occur in wheeled robots as well [13]. Therefore, this is an indication that from the controller's point of view, given the change in morphology δM from the quadruped to the hexapod, there was an increase in complexity for the controller δC from 0 hidden units to 3 hidden units. Hence, the hexapod morphology can be seen as being placed at a higher level of the complexity hierarchy than the quadruped morphology in the eyes of the controller.

If we would like to measure the complexity of the morphology using the behavioral scale, we can notice from the graph that the maximum distance achieved by the quadruped creature is around 17.8 compared to around 13.8 for the hexapod creature. In this case, the quadruped can be seen as being able to achieve a more complex behavior than the hexapod.

5.2 Behavioral Complexity

A comparison of the results obtained using the two different sets of fitness functions $P1$ and $P2$ is presented in Table 1. Here we are fixing M , L and E and looking for the change in behavioral complexity. The morphology M is fixed by using the quadruped creature only. For $P1$, we can see that the Pareto-frontier offers a number of different behaviors. For example, a network with no hidden units can achieve up to 14.7 units of distance while the creature driven by a network with 5 hidden units can achieve 17.7 units of distance within the 500

Table 1. Comparison of global Pareto optimal controllers evolved for the quadruped using the $P1$ and $P2$ objective functions.

Type of Behavior	Pareto Controller	No. of Hidden Units	Locomotion Distance	Static Stability
P1	1	0	14.7	19
	2	1	15.8	24
	3	2	16.2	30
	4	3	17.1	26
	5	4	17.7	14
P2	1	0	5.2	304
	2	1	3.3	408
	3	2	3.6	420
	4	3	3.7	419

timesteps. This is an indication that to achieve a higher speed gait entails a more complex behavior than a lower speed gait.

We can also see the effect of static stability, which requires a walking behavior. By comparing a running behavior using a dynamic gait in $P1$ with no hidden units against a walking behavior using a static gait in $P2$ with no hidden units, we can see that using the same number of hidden units, the creature achieves both a dynamic as well as a quasi-static gait. If more static stability is required, this will necessitate an increase in controller complexity.

At this point of comparison, we find that the behavior achieved with the $P1$ fitness functions consistently produced a higher locomotion distance than the behavior achieved with the $P2$ fitness functions. This meant that it was much harder for the $P2$ behavior to achieve the same level of locomotion competency in terms of distance moved as the $P1$ behavior due to the added sub-objective of having to achieve static stability during locomotion. Thus, the complexity of achieving the $P2$ behavior can be seen as being at a higher level of the complexity hierarchy than the $P1$ fitness function in the eyes of the controller.

6 Conclusion and Future Work

We have shown how EMO can be applied for studying the behavioral and morphological complexities of artificially evolved embodied creatures. The morphological complexity of a quadruped creature was found to be lower than the morphological complexity of a hexapod creature as seen from the perspective of an evolving locomotion controller. At the same time, the quadruped was found to be more complex than the hexapod in terms of behavioral complexity. For future work, we intend to provide an empirical proof of measuring not only behavioral complexity but also environmental complexity by evolving controllers for artificial creatures in varied environments. We also plan to apply these measures for characterizing the complexities of artificial creatures evolved through co-evolution of both morphology and mind.

References

1. Hussein A. Abbass. The self-adaptive pareto differential evolution algorithm. In *Proceedings of the 2002 Congress on Evolutionary Computation (CEC2002)*, volume 1, pages 831–836. IEEE Press, Piscataway, NJ, 2002.
2. Josh C. Bongard. Evolving modular genetic regulatory networks. In *Proceedings of the 2002 Congress on Evolutionary Computation (CEC2002)*, pages 1872–1877. IEEE Press, Piscataway, NJ, 2002.
3. Rodney A. Brooks. Intelligence without reason. In L. Steels and R. Brooks (Eds), *The Artificial Life Route to Artificial Intelligence: Building Embodied, Situated Agents*, pages 25–81. Lawrence Erlbaum Assoc. Publishers, Hillsdale, NJ, 1995.
4. Critical Mass Labs. Vortex [online]. <http://www.cm-labs.com> [cited – 25/1/2002].
5. Kalyanmoy Deb. *Multi-objective Optimization using Evolutionary Algorithms*. John Wiley & Sons, Chicester, UK, 2001.
6. Claus Emmeche. *Garden in the Machine*. Princeton University Press, Princeton, NJ, 1994.
7. David P. Feldman and James P. Crutchfield. Measures of statistical complexity: Why? *Physics Letters A*, 238:244–252, 1998.
8. Dario Floreano and Joseba Urzelai. Evolutionary robotics: The next generation. In T. Gomi, editor, *Proceedings of Evolutionary Robotics III*, pages 231–266. AAI Books, Ontario, 2000.
9. Simon Haykin. *Neural networks – a comprehensive foundation*. Prentice Hall, USA, 2 edition, 1999.
10. Gregory S. Hornby and Jordan B. Pollack. Body-brain coevolution using L-systems as a generative encoding. In L. Spector et al. (Eds), *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, pages 868–875. Morgan Kaufmann, San Francisco, 2001.
11. Andrei N. Kolmogorov. Three approaches to the quantitative definition of information. *Problems of Information Transmission*, 1:1–7, 1965.
12. Hod Lipson and Jordan B. Pollack. Automatic design and manufacture of robotic lifeforms. *Nature*, 406:974–978, 2000.
13. Henrik H. Lund and John Hallam. Evolving sufficient robot controllers. In *Proceedings of the 4th IEEE International Conference on Evolutionary Computation*, pages 495–499. IEEE Press, Piscataway, NJ, 1997.
14. Rolf Pfeifer and Christian Scheier. *Understanding Intelligence*. MIT Press, Cambridge, MA, 1999.
15. Jordan B. Pollack, Hod Lipson, Sevan G. Ficici, Pablo Funes, and Gregory S. Hornby. Evolutionary techniques in physical robotics. In Peter J. Bentley and David W. Corne (Eds), *Creative Evolutionary Systems*, chapter 21, pages 511–523. Morgan Kaufmann Publishers, San Francisco, 2002.
16. Cosma R. Shalizi. *Causal Architecture, Complexity and Self-Organization in Time Series and Cellular Automata*. Unpublished PhD thesis, University of Wisconsin at Madison, Wisconsin, 2001.
17. Claude E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27(3):379–423, 1948.
18. Karl Sims. Evolving 3D morphology and behavior by competition. In R. Brooks and P. Maes (Eds), *Artificial Life IV: Proceedings of the Fourth International Workshop on the Synthesis and Simulation of Living Systems*, pages 28–39. MIT Press, Cambridge, MA, 1994.
19. Russell K. Standish. On complexity and emergence [online]. *Complexity International*, 9, 2001.

Learning Biped Locomotion from First Principles on a Simulated Humanoid Robot Using Linear Genetic Programming

Krister Wolff and Peter Nordin

Dept. of Physical Resource Theory, Complex Systems Group,
Chalmers University of Technology,
S-412 96 Göteborg, Sweden
{wolff, nordin}@fy.chalmers.se
<http://www.frt.fy.chalmers.se/cs/index.html>

Abstract. We describe the first instance of an approach for control programming of humanoid robots, based on evolution as the main adaptation mechanism. In an attempt to overcome some of the difficulties with evolution on real hardware, we use a physically realistic simulation of the robot. The essential idea in this concept is to evolve control programs from first principles on a simulated robot, transfer the resulting programs to the real robot and continue to evolve on the robot. The Genetic Programming system is implemented as a Virtual Register Machine, with 12 internal work registers and 12 external registers for I/O operations. The individual representation scheme is a linear genome, and the selection method is a steady state tournament algorithm. Evolution created controller programs that made the simulated robot produce forward locomotion behavior. An application of this system with two phases of evolution could be for robots working in hazardous environments, or in applications with remote presence robots.

1 Introduction

Dealing with humanoid robots requires supply of expertise in many different areas, such as vision systems, sensor fusion, planning and navigation, mechanical and electrical hardware design, and software design only to mention a few. The objective of this paper, however, is focused on the synthesizing of biped gait. The traditional way of robotics locomotion control is based on derivation of an internal geometric model of the locomotion mechanism, and requires intensive calculations by the controlling computer, to be performed in real time. Robots, designed in such a way that a model can be derived and used for controlling, shows large affinity with complex, highly specialized industrial robots, and thus they are as expensive as conventional industrial robots. Our belief is that for humanoids to become an everyday product in our homes and society, affordable for everyone, there is needed to develop low cost, relatively simple robots. Such robots can hardly be controlled the traditional way; hence this is not our primary design principle.

A basic condition for humanoids to successfully operate in human living environment is that they must be able to deal with unpredictable situations and gather knowledge and information, and adapt to their actual circumstances. For these reasons, among others, we propose an alternative way for control programming of humanoid robots. Our approach is based on evolution as the main adaptation mechanism, utilizing computing techniques from the field of Evolutionary Algorithms.

The first attempt in using a real, physical robot to evolve gait patterns was made at the University of Southern California. Neural networks were evolved as controllers to produce a tripod gait for a hexapod robot with two degrees of freedom for each leg [6]. Researchers at Sony Corporation have worked with evolving locomotion controllers for dynamic gait of their quadruped robot dog AIBO. These results show that evolutionary algorithms can be used on complex, physical robots to evolve non-trivial behaviors on these robots [3] and [4].

However, evolving efficient gaits with real physical hardware is a challenge, and evolving biped gait from first principles is an even more challenging task. It is extremely stressing for the hardware and it is very time consuming [17]. To overcome the difficulties with evolving on real hardware, we introduce a method based on simulation of the actual humanoid robot.

Karl Sims was one of the first to evolve locomotion in a simulated physics environment [13] and [14]. Parker use Cyclic Genetic Algorithms to evolve gait actuation lists for a simulated six legged robot [11], and Jakobi et al has developed a methodology for evolution of robot controllers in simulator, and shown it to be successful when transferred to a real, physical octopod robot [7] and [9]. This method, however, has not been validated on a biped robot. Recently, a research group in Germany reported an experiment relevant to our ideas, where they evolved robot controllers in a physics simulator, and successfully executed them onboard a real biped robot. They were not able to fully realize biped locomotion behavior, but their results were definitely promising [18].

2 Background and Motivation

In this section we summarize an on-line learning experiment performed with a humanoid robot. However this experiment was fairly successful in evolving locomotion controller parameters that optimized the robot's gait, it pointed out some difficulties with on-line learning. We summarize the experiment here in order to exemplify the difficulties of evolving gaits on-line, and let it serve as an illustrative motivation for the work presented in the remainder of this paper.

2.1 Robot Platform

The robot used in the experiments is a simplified, scaled model of a full-size humanoid with body dimensions that mirrors the dimensions of a human. It was originally developed as an alternative, low-cost humanoid robot platform,

intended for research [17]. It is a fully autonomous robot with onboard power supply and computer, and it has 14 degrees of freedom.

The robot has the 32-bit micro-controller EyeBot MK3 [2] onboard, carrying it as a backpack. All signal processing, including control, vision, and evolutionary algorithm, is carried out on the EyeBot controller itself. In its present status, the robot is capable of static walking.

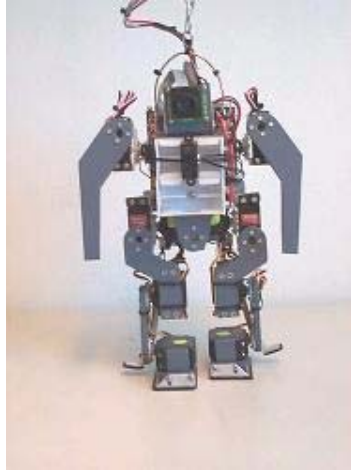


Fig. 1. Image of the real humanoid robot ‘elvina’.

2.2 Gait Control Method

The gait control method for this robot involves repetition of a sequence of integrated steps. Considering fully realistic bipedal walk, two different situations arise in sequence: the statically stable double-support phase in which the robot is supported on both feet simultaneously, and statically unstable single-support phase when only one foot of the robot is in contact with the ground, the other foot being transferred from the back to front position. When this sequence of transitions has been repeated twice, one can consider a single gait cycle to be completed. That is, the locomotion mechanism’s posture and limb’s positions are the same after the completion as it was before it started to move, and hence it’s internal state is the same.

If we now study only static walk, i.e. the projection of the center of mass of the robot on the ground always lie within the support polygon formed by feet on the ground, there is obviously a number of statically stable postures in between the internal state of the robot and it’s final state, during completion of a single gait cycle. By interpolation between numbers of such, statically stable, consecutive states it is possible to make the robot to complete a single gait cycle. Then, by continually looping, biped gait is produced.

2.3 Evolutionary Gait Optimization Experiment

This experiment was performed in order to optimize a by hand developed set of state vectors, defining a static robot gait. The evolutionary algorithm used was a tournament selection, steady state evolutionary strategy [1] and [16] running on the robot's onboard computer. Individuals were evaluated and fitness scores automatically determined using the robots onboard digital camera and proximity sensor.

A population of 30 individuals, stemming from a manually developed individual was created with a uniform distribution over a given search range. The best-evolved individual and the manually developed individual were independently tested, and their performances were compared to each other's. The former one received a fitness score, averaged over three trials, of 0.1707, and the latter one, tested under equal conditions, got a fitness of 0.1051. Within this context, a higher fitness value means a better individual, and thus the best-evolved individual outperformed the manually developed individual both in its ability to maintain the robot in a straight course and in robustness, i.e. with a lesser tendency to fall over [17].

2.4 Observations

To run such an evolutionary experiment as described above span over several days, and requires manual supervision all this time. Between each generation of four individuals evaluated, the experiment was paused for about 15 minutes in order to spare the hardware and especially the actuators. The main reason for this is that the actuators accumulate heat when they are running continuously under heavy stress. They then run the risk of getting overheated and gradually destroyed. One way to handle this problem was, as mentioned above, to run the robot intermittent so that the servos maintain an approximately constant temperature.

Evolving efficient gaits with real physical hardware is a challenging task. During the experiments, the torso and both the ankle actuators were exchanged once as well as the two hip servos. The most vulnerable parts of the robot were proved to be the knee servos. Both these servos were replaced three times.

Obviously there are a number of difficulties related with evolving biped walking behavior on a real, physical robot. In an attempt to overcome some of the problems, we want to use a physically realistic simulation of the robot. The central idea in this concept is to evolve control programs from first principles on a simulated robot, transfer the resulting programs to the real robot and continue to evolve efficient gait on the real robot. Of course, there will arise other problems applying this method, as simulation systems always imply some simplifications of the real world.

3 Evolution of Control Programs

Our primary goal is to utilize Genetic Programming [5] and [8] for evolving locomotion control programs from first principles for our simulated biped robot,

i.e. with no a priori knowledge for the robot on how to walk, information of morphology etc. The evolved programs take the robot's *current* internal state parameter values as input vector and return a vector predicting it's *next* internal state parameter values, in order to produce robust biped gait.

3.1 Dynamic Physics Simulation

The Open Dynamics Engine (ODE) is a free library for simulating articulated rigid body dynamics, developed by Russell Smith [15]. An articulated structure is created when rigid bodies of various shapes are connected together with joints of various kinds.

The robot model is qualitatively consistent with the real robot in the aspect of geometry, mass distribution, and morphology. See [17] for details of the robot. It consists of 12 actuated joints and 13 body elements. It is constructed with its mass concentrated to the main body elements, which in the real robot correspond to the servo actuators, batteries and computer. The plastic body parts, interconnecting the servos to each other, are not rendered in the simulation, since their mass is very low compared to the total mass.

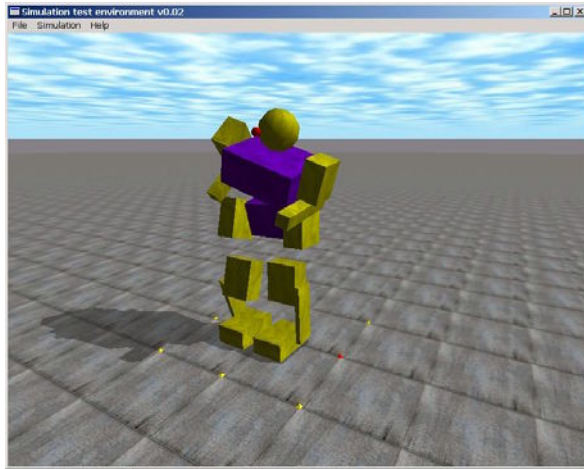


Fig. 2. Snapshot of the simulated humanoid robot. The body elements are directly connected to each other, although this is not visualized here.

3.2 Virtual Register Machine

The Genetic Programming representation used for this problem of robot control program induction is an instance of a Virtual Register Machine, $\text{VRM}(k, l)$ [10].

It has k I/O registers and l internal work registers. In the current implementation of our system, l equals k . The function set consists in the present of arithmetic functions ADD, SUB, MUL, DIV, where DIV is protected division, and SINE. We now define a register state vector $\mathbf{Reg} \equiv [Reg_1, \dots, Reg_k]$ of k integers, each of the elements corresponding to one of the actuated joints of the simulated robot. All program input/output is communicated through the states of the I/O registers. That is, program inputs are supplied in the initial state Reg , and output is taken from the final register state Reg' . Further, the I/O register state vector is initially copied into the internal work registers. We can do this in a straight forward manner, since we have imposed that the number of I/O registers, k , equals the number of work registers, l . The Virtual Register Machine is allowed writing only to the internal work registers when looping the program instructions. The I/O registers are write-protected in this phase, and their final state is updated after the end of the program execution cycle, before they are passed to the robot and then updating it's internal state.

3.3 Linear Genome Representation

Each individual is composed of simple instructions between input and output parameters. Each instruction consists of four elements, encoded as integers, and the whole individual is a linear list of such instructions:

```

8, 22, 3, 12,
19, 11, 2, 16,
15, 12, 3, 12,
8, 3, 4, 19,
12, 12, 4, 21,
1, 6, 5, 12,
20, 3, 1, 19,
9, 12, 2, 21,
23, 5, 3, 19,
16, 9, 3, 14,
13, 21, 5, 19,
6, 13, 5, 14,
16, 22, 3, 16,
16, 3, 4, 18,
8, 19, 2, 13,
20, 5, 3, 20,
13, 6, 1, 14,
```

The encoding scheme is as follows; the first and second elements of an instruction refers to the registers to be used as arguments, the third element corresponds to the operator, i.e. ADD=1, SUB=2, MUL=3, DIV=4, and SINE=5, and the last element is a register reference for where to put the result of the operation. The meaning of the first line (instruction) here is: multiply register 8 with register 22 and put the result in register 12. The operators take two arguments,

except when the operator is SINE, which of course only take one argument. In this case, the SINE operator is applied to the first element in the instruction, and the second element is simply discarded. A mutation on that element will thus have no effect on that individual's genotype. The register references 1-11 are assigned to I/O-registers, and register references 12-23 are assigned for the internal work registers. Parsing the individual above, and print out the first three instructions in 'C-style' looks like this:

```
Reg12 = Reg8 * Reg22;
Reg16 = Reg19 - Reg11;
Reg12 = Reg15 * Reg12;
```

3.4 Evolutionary Algorithm

At the beginning of the evolutionary process, the population is filled with randomly created individuals. The length, or number of instructions, of an individual is chosen randomly with Gaussian distribution, with expectation value 20. The maximum length is restricted to 256 instructions. The genes are created with a uniform distribution over their respective search range; 1-23 for the two first genes of an instruction, 12-23 for the last gene, and 1-5 for the third gene, which corresponds to the function set.

Our GP-system is a steady state tournament selection algorithm, with the following execution cycle:

1. Select four members of the population for tournament.
2. For all members in tournament do:
 - a. Create an instance of the simulated robot.
 - b. Record the position in 3d-space of all the robot's limbs.
 - c. Execute the individual for 2500 simulation time steps.
 - d. Record the final position of all the robot's limbs.
 - e. Compute the fitness value (see below).
 - f. Destroy the simulated robot.
3. Perform tournament selection.
4. Apply genetic operators on the winners to produce two children.
5. Replace the two losers in the population with the offspring.
6. Go to step 1.

The individuals are evaluated (evaluation cycle starting with point 2a. above) under identical conditions, since the simulation is entirely deterministic. They all start from the same standing upright pose, with the same orientation. The execution time for individuals are 2500 simulation time steps (corresponding to approx. 20 seconds of real time simulation), and if an individual cause the robot to fall before this time is completed, the evaluation is terminated. In the beginning of an experiment, a great majority of individuals are terminated before the intended time. Looping an individual once does *not* correspond to a single simulation time step, but to moving the robot's limbs between two consecutive internal states ('states' being referred to as in the subsection *Gait Control Method*).

Table 1. Koza style tableau, showing parameter settings for the evolution of locomotion control programs for the simulated humanoid robot.

Parameter	Value
Objective	Approximate a function that produce robust biped gait
Terminal Set	24 integer registers,
Function Set	ADD, SUB, MUL, DIV, SINE
Raw Fitness	According to eq. (2), scalar value
Standardized Fitness	Same as Raw Fitness
Population Size	800
Initialization Method	Random
Simulation Time	2500 simulation time steps
Crossover Probability	100%
Mutation Probability	80%
Initial Program Length	Gaussian distribution, expectation value 20.
Maximum Program Length	256 instructions
Maximum Tournament Number	None
Selection Scheme	Tournament, size 4
Termination Criteria	None (determined by the experimenter)

Fitness Calculation. As in all GP-applications, finding a proper fitness function that guides the artificial evolution in the desired direction is of great importance. The primary goal for the experiment was to produce a "human-like", bipedal gait without the robot falling. To accomplish this task, the individual controlling the robot should; (i) locomote the robot as straight forward as possible, and (ii) keep the robot in an upright pose during the movement. Hence, the proper measurements to feed the fitness function with are related to the height maintained by the robot, and the covered distance during simulation. Explicitly formulated in mathematical terms, the proper fitness function was found to be:

$$f = W \left[1.0 - \frac{h_{start}}{h_{stop}} \right] + (d_{left} + d_{right}) \quad (1)$$

where h_{start} is the height of the robot at the starting position, h_{stop} is the height when evaluation terminates (either the simulation is fully completed, or it is terminated before the intended time, caused by the robot falling). The height measure is applied to the position of the robot's head, however one could take the height of any body part. The second term is a measure of the distance covered by the robot during evaluation, applied to its feet. The robot is always starting with its feet in origo (in xy -plane). The first term will give a positive contribution to fitness if $h_{stop} > h_{start}$, negative contribution in the case when $h_{stop} < h_{start}$, and zero contribution if $h_{stop} = h_{start}$. Thus we have a fitness function rewarding forward locomotion and keeping the upright pose, and punishing backward movements and falling. The W in the first term is a weight,

scaling the mutual relation of rewarding and punishing. After some tweaking, it was found to work best when set to a value in the order of 10.

Genetic Operators. We use only two-point string crossover, with 100% probability for crossover, divided mutually on the rate 4:1 on homologous and non-homologous crossover.

When an individual is chosen for mutation, the mutation operator works by randomly selecting one single instruction from the individual, and make a change in the selected instruction. It makes that change either by changing any of the register references to another randomly chosen register reference from the register set, or the operator in the instruction may be changed. The probability for an individual to undergo mutation is 80%.

4 Results

When observing the experiments in run-time, it is compelling how quickly the simulated robot learns. In the first couple of hundred tournaments, a great majority of the individuals cause the robot to fall almost immediately in the beginning of the evaluation cycle, and the greater part of them tip over backwards. Maybe one out of ten individuals fall to the fore, which is a good starting point of taking a step ahead. Rather soon, however, one can observe the opposite situation, one out of ten individuals' overturn backwards and the rest fall ahead. This was not the desired goal for the evolution, but we regard this as being the first refined behavior that emerged.

The next observable stage of development in the evolution is when a large fraction of individuals is keeping the robot at a standstill, almost motionless, on its feet. In the beginning of our experiments, we faced some problems with evolution converged to this state. By increasing the population size and making some adjustments to the fitness function (mainly by decreasing the weight w , giving lesser punishment for tipping over), we could guide the evolution towards the desired goal. The mix of individuals showing this behavior, and individuals with a more 'energetic' behavior guarantee sufficient diversity of the population for evolution to proceed.

The final results of these experiment was indeed consistent with our initial objectives. That is, evolution created controller programs that made the simulated robot produce forward locomotion behavior. Some of the resulting programs made the robot walking forward in a spiral manner, with small movements, and others produced gait patterns with more lively movements. When tested, some of the individuals managed to keep the robot on its feet for the whole evaluation time (2500 simulation time steps), but when executed for a longer time, the robot usually ended up overturned. Nevertheless, a division of evolved programs could accomplish the task during the test run, without ever tipping over the robot.

Figures 3 and 4 displays some statistics from a representative run. In these experiments we did more than thirty independent runs, ranging from a few thousand

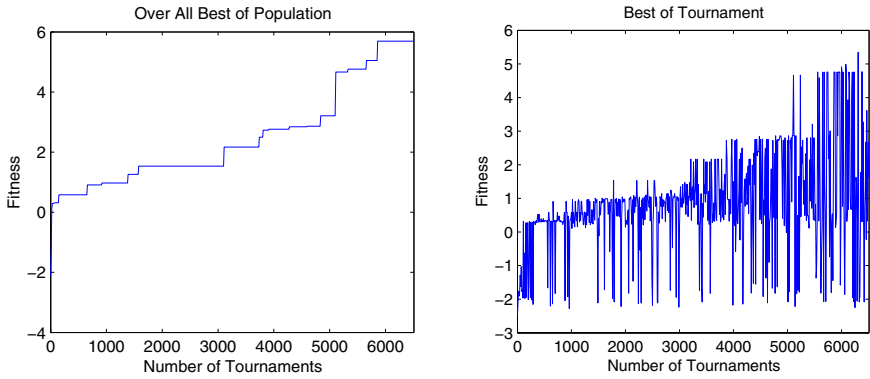


Fig. 3. **a**, Fitness value of over all best individual in the population (left) and **b**, fitness value of the best individual in every tournament (right).

tournaments, up to more than 80000 tournaments. The way fitness was defined (eq. 2), a fitness value < 0 correspond to the robot falling backward, and a small positive value (typically ranging from ~ 0.3 to ~ 0.6) correspond to the robot immediately falling ahead, while a value around 1.5 indicate a standstill. In figure 3a, one can observe how the best individual performed those behaviors; falling backward in the first few hundred tournaments, falling ahead in the first thousand tournaments, and standing still up to the 3000 tournaments. Fitness values in the range of ~ 1.5 to ~ 2.5 indicate some good locomotion, but usually ended up with the robot overturned, and fitness > 2.5 was successful locomotion behavior.

As depicted in figure 3a, the currently best individuals in the population showed progress from the beginning of the evolution and continued to develop over time. The program length typically decrease below the initialization length in the beginning of a run, but after a short while it starts to increase above that threshold, and finally it stabilize around some value. See figure 4. In all experiments we used the same initialization program length, with gaussian distribution and expectation value 20. It was observed that the program length, averaged over the whole population, did never go below the value 13, and never above 50, and it usually stabilized somewhere around 30.

5 Summary and Conclusions

We describe the first instance of an approach for control programming of humanoid robots. It is based on evolution as the main adaptation mechanism, utilizing computing techniques from the field of Evolutionary Algorithms. The central idea in this concept is to evolve control programs from first principles on a simulated robot, transfer the resulting programs to the real robot and continue to evolve efficient gait on the real robot. As the key motivation for using

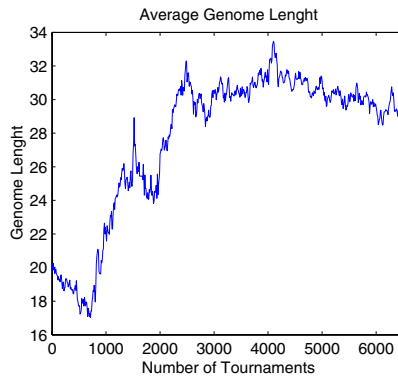


Fig. 4. Average genome length of all individuals in the population, length being defined as the number of instructions in an individual.

simulators, we briefly describe an on-line learning experiment performed with a biped humanoid robot.

The Evolutionary Algorithm is an instance of Genetic Programming, implemented as a Virtual Register Machine with 12 internal work registers and 12 external registers for I/O operations. The individual representation scheme is a linear genome, encoded as an array of integers. The selection method is a steady state tournament algorithm, with size four. The final results of these experiment was consistent with our initial objectives. That is, evolution created controller programs that made the simulated robot produce forward locomotion behavior. Current versions of the simulation system and the robot, however, do not allow the evolved programs to be directly downloaded to the robot. Further investigations and improvements are needed. To begin with, we must implement a subsystem of the simulated robot's control system and program interpreter on the real robots micro controller. Further, the real robot has an active feedback system, consisting of a color camera and a distance sensor, which will be implemented on the simulated robot as well. The development of the robot platform is an ongoing process, hence other sensors will be implemented on the robot. Then, the simulated robot should of course reflect all aspects, morphological and perceptual, of the real robot.

With this system of two phases of evolution, it will be possible to have a flexible adaptation mechanism that can react to hardware failures in the robot, e.g. if an actuator or sensor break down. By extracting information about malfunctioning parts and do off-line evolution with a modified model of the robot, it will become possible to react to the changes in the robot morphology. Another approach in this spirit, called Punctuated Anytime Learning, has been proposed by Parker [12]. For robots working in hazardous environments, or in applications with remote presence robots, this feature would be very useful.

References

1. Banzhaf, W., Nordin, P., Keller, R.E., and Francone F. D.: Genetic Programming An Introduction: On the Automatic Evolution of Computer Programs and Its Applications. San Francisco: Morgan Kaufmann Publishers, Inc. Heidelberg: dpunkt verlag. (1998)
2. Bräunl, T. 2002: EyeBot Online Documentation. Last visited: 01/21/2003. <http://www.ee.uwa.edu.au/~braunl/eyebot/>
3. Hornby, G.S., Fujita, M. Takamura, S., Yamamoto, T., and Hanagata, O.: Autonomous evolution of gaits with the Sony quadruped robot. Proceedings of the Genetic and Evolutionary Computation Conference. San Francisco: Morgan Kaufmann Publishers, Inc. (1999)
4. Hornby, G.S., Takamura, S., Yokono, J., Hanagata, O., Yamamoto, T., and Fujita, M.: Evolving robust gaits with AIBO. IEEE International Conference on Robotics and Automation, New York: IEEE Press, pages 3040–3045. (2000)
5. Koza, J.: Genetic Programming: On the Programming of Computers by Means of Natural Selection. Cambridge, MA, USA: MIT Press. (1992)
6. Lewis, M. A., Fagg, A. H., and Solidum, A.: Genetic programming approach to the construction of a neural network for control of a walking robot. Proceedings of the IEEE International Conference on Robotics and Automation. New York: IEEE Press. (1992)
7. Lund, H., and Miglino, O.: From Simulated to Real Robots. Proceedings of IEEE 3rd International Conference on Evolutionary Computation. New York: IEEE Press. (1996)
8. Langdon, W. B., and Poli, R.: Foundations of Genetic Programming. New York: Springer-Verlag. ISBN 3-540-42451-2, 274 pages. (2002)
9. Miglino, O., Lund, H., and Nolfi S.: Evolving Mobile Robots in Simulated and Real Environments. Technical Report, Institute of Psychology, C.N.R., Rome. (1995)
10. Nordin, P.: Evolutionary Program Induction of Binary Machine Code and its Applications. Ph.D. Thesis, der Universität Dortmund am Fachbereich Informatik, Germany. (1997)
11. Parker, G. and Rawlins, G.: Cyclic Genetic Algorithms for the Locomotion of Hexapod Robots. Proceedings of the World Automation Congress, Volume 3, Robotic and Manufacturing Systems. (1996)
12. Parker, G.: Punctuated Anytime Learning for Hexapod Gait Generation. Proceedings of the 2002 IEEE/RSJ International Conference on Intelligent Robots and Systems. (2002)
13. Sims, K.: Evolving Virtual Creatures. Proceedings of Siggraph, pp.15–22. (1994)
14. Sims, K.: Evolving 3D Morphology and Behavior by Competition. Proceedings of Artificial Life IV, Brooks and Maes, editors, MIT Press, pp.28–39. (1994)
15. Smith, R.: Open Dynamics Engine v0.030 User Guide. Last Visited: 03/27/2003. <http://opende.sourceforge.net/ode-0.03-userguide.html>
16. Schwefel, H. P.: Evolution and Optimum Seeking. New York, USA: Wiley. (1995)
17. Wolff, K., and Nordin, P.: Evolution of Efficient Gait with an Autonomous Biped Robot using Visual Feedback. Proceedings of the Mechatronics Conference. University of Twente, Enschede, the Netherlands. (2002)
18. Ziegler, J., Barnholt, J., Busch, J., and Banzhaf W.: Automatic Evolution of Control Programs for a Small Humanoid Walking Robot. 5th International Conference on Climbing and Walking Robots. (2002)

An Evolutionary Approach to Automatic Construction of the Structure in Hierarchical Reinforcement Learning

Stefan Elfving^{1,2,3}, Eiji Uchibe^{2,3}, and Kenji Doya^{2,3}

¹ KTH, Numerical Analysis and Computer Science department,
KTH, Nada, 100 44 Stockholm, Sweden

² ATR, Human Information Science Laboratories, Department 3

³ CREST, Japan Science and Technology Corporation
2-2-2 Hikaridai, "Keihanna Science City" Kyoto 619-0288, Japan

1 Introduction

Hierarchical reinforcement learning (RL) methods have been developed to cope with large scale problems. However, in most hierarchical RL methods, an appropriate structure of hierarchy has to be hand-coded. This paper presents an evolutionary approach for automatic construction of hierarchical structures in RL.

2 Proposed Method

Our method combines the MAXQ method [1] and Genetic Programming (GP). The MAXQ method learns the policy based on the hierarchy obtained by the GP, while GP explores the appropriate hierarchies using the result of the MAXQ method. Leaf nodes and inner nodes of MAXQ representation are regarded as terminals and functions for GP. We use strongly-typed GP [2] that allows the designer to assign specific types to the arguments and the return value of each function.

3 Task and Experimental Results

We have performed simulation experiments with a rodent like robot, *Cyber Rodent*. The task is to find, approach and capture a battery pack, and then return the battery pack to the nest. We have prepared three different environments shown in Fig. 1. Cyber Rodent has the distance sensors and the vision system. For all sensor readings in the simulated environment, noise (10 % of input range) is added. In this experiment, we prepared seven composite subtasks (root, capture, deliver, find battery, visible battery, find nest, and visible nest), and five primitive subtasks (avoid, wander, approach battery, approach nest, and turn). Each individual in the population performs a fixed number of trials in each generation. The fitness is calculated as the number of

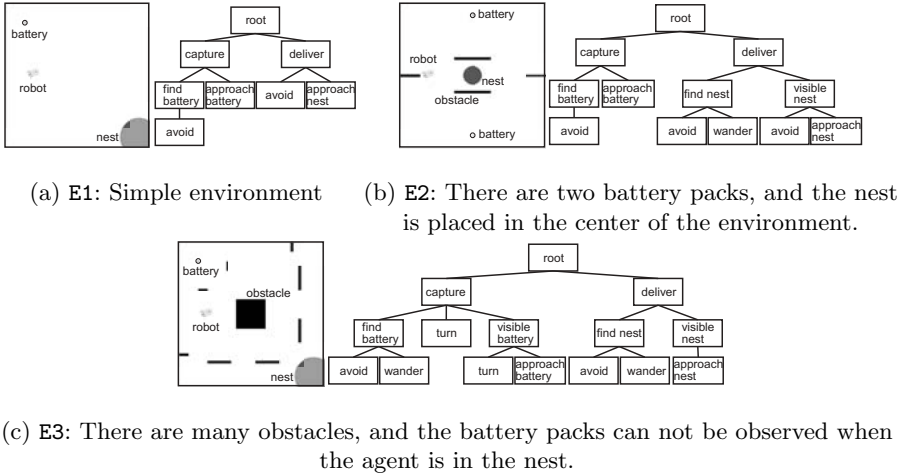


Fig. 1. Tested environments and examples of obtained hierarchies. The small filled light gray circles represent battery packs and the big darker gray circles represent the nests, respectively.

time steps to complete the task. The parent hierarchies for crossover are chosen by tournament selection.

Experimental results showed that GP found suitable hierarchies for all three environments. A remarkable finding was that the complexity of the obtained hierarchical structures was strongly constrained by the complexity of the environment, GP for a simple environment obtained a simple and specialized hierarchy and GP for a complex environment obtained a complex and general task hierarchy. The main difference between **E1** and **E2** was that the nest in **E2** was surrounded by the wall although the battery packs were placed in an open field in both environments. Accordingly, the part of **deliver** in the obtained structure was different, as shown in Fig. 1(a) and (b). In **E3**, since there were many obstacles and the environment was crowded, the obtained structure was the most complicated.

4 Conclusion

We plan to implement our method to the real hardware. A foreseeable extension of this study is to generalize the method as a model of cooperative and competitive mechanisms of the learning modules in the brain.

Acknowledgments. This research was supported in part by the Telecommunications Advancement Organization of Japan.

References

1. T. G. Dietterich. Hierarchical Reinforcement Learning with the MAXQ Value Function Decomposition. *Journal of Artificial Intelligence Research*, 13:227–303, 2000.
2. D. J. Montana. Strongly Typed Genetic Programming. *Evolutionary Computation*, 3(2):199–230, 1995.

Fractional Order Dynamical Phenomena in a GA

E.J. Solteiro Pires¹, J.A. Tenreiro Machado², and P.B. de Moura Oliveira¹

¹ Universidade de Trás-os-Montes e Alto Douro, Dep. de Engenharia Electrotécnica,
Quinta de Prados, 5000-911 Vila Real, Portugal,

{epires,oliveira}@utad.pt, <http://www.utad.pt/~epires>

² Instituto Superior de Engenharia do Porto, Dep. de Engenharia Electrotécnica,
Rua Dr. António Bernadino de Almeida, 4200-072 Porto, Portugal
jtm@dee.isep.ipp.pt, <http://www.dee.isep.ipp.pt/~jtm>

Abstract. This work addresses the fractional-order dynamics during the evolution of a *GA*, which generates a robot manipulator trajectory. In order to investigate the phenomena involved in the *GA* population evolution, the crossover is exposed to excitation perturbations and the corresponding fitness variations are evaluated. The input/output signals are studied revealing a fractional-order dynamic evolution, characteristic of a long-term system memory.

1 The GA Trajectory Planning Scheme

This section presents a *GA* that calculates the trajectory of a two-link manipulator that is required to move between two points. The path is encoded directly, using real codification, as strings in the joint space to be used by the *GA* as: $[\Delta t, (q_{11}, q_{21}), \dots, (q_{1j}, q_{2j}), \dots, (q_{1m}, q_{2m})]$. The i th joint variable for a robot intermediate j th position is q_{ij} , at time $j\Delta t$. The fitness function f adopted for evaluating the trajectories is defined as:

$$f = \beta_1 f_\tau + \beta_2 \sum_{j=2}^m \sum_{i=1}^2 \dot{q}_{ij}^2 + \beta_3 \sum_{j=2}^{m-1} \sum_{i=1}^2 \ddot{q}_{ij}^2 + \beta_4 \sum_{j=2}^m \dot{p}_j^2 + \beta_5 \sum_{j=2}^{m-1} \ddot{p}_j^2 \quad (1)$$

The f_τ index represents the excessive torque that is demanded for the joints motors, p_j is the j cartesian arm position. This simple experiment consists on moving a robotic arm between two points. In the *GA* are adopted $p_c = 0.8$, $p_m = 0.05$, a 200 population size, a string size of $m = 7$ and a 3-tournament selection. The robot parameters are $l_i = 1\text{m}$, $m_i = 1\text{kg}$ and $\tau_{i,max} = \{16, 5\}\text{Nm}$ ($i = 1, 2$). Figure 1ab show the simulation results. The trajectory presents a smooth behavior, both in the space and time evolution and the required joint torques do not exceed the imposed limitations.

2 Fractional-Order Dynamics

The *GA* system is stimulated by perturbing the crossover probability p_c through a white noise signal, with a small amplitude (1%) during a time period T_{exc} , and

the corresponding modification of the population fitness is evaluated. Therefore, the variation of the crossover probability and the resulting fitness modification of the GA population, during the evolution, can be viewed as the system input and output signals versus time, respectively.

The transfer function $H_n(j\omega)$, between the input and output signals, and the fractional order analytical approximation $G_n(j\omega)$ are depicted in figure 1c.

The numerical data of the transfer functions are approximated by analytical expressions with gain $k \in \mathbb{R}$, one zero and one pole $(a, b) \in \mathbb{R}$ of fractional orders $(\alpha, \beta) \in \mathbb{R}$, respectively, given by $G_n(s) = k[(\frac{s}{a})^\alpha + 1]/[(\frac{s}{b})^\beta + 1]$ for the n th fitness percentile P_n . For evaluating the influence of the excitation period T_{exc} several simulations are developed. The relation between the transfer function parameters $\{k, \alpha, \beta\}$ and (T_{exc}, P_n) are showed in figure 1def.

3 Conclusions

Fractional-order models capture phenomena and properties that classical integer-order simply neglect. For the case under study the signal evolution have similarities to those revealed by chaotic systems. This conclusion confirms the requirement for mathematical tools well adapted to the phenomena under investigation. In this line of thought, this article is a step towards the signal and system analysis based on the theory of Fractional Calculus.

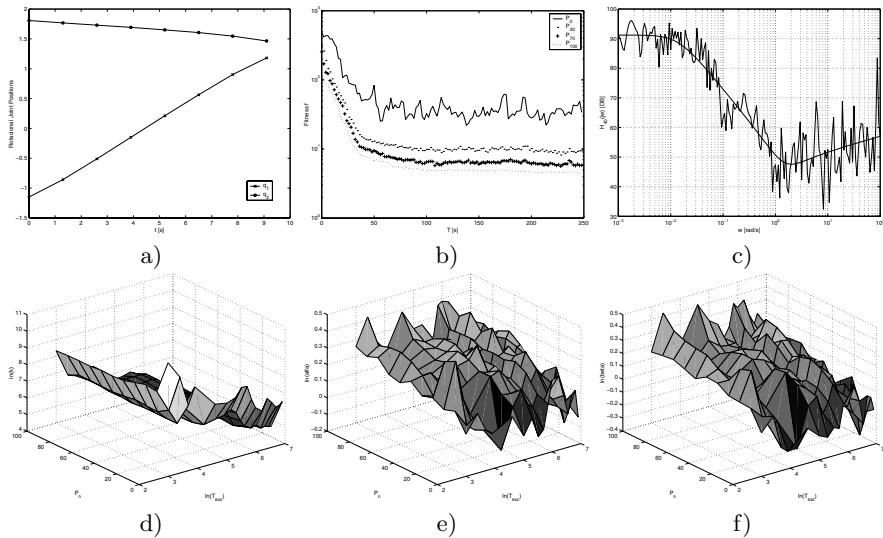


Fig. 1. a) Robot joint positions *vs.* time. b) Percentiles of the population fitness *vs.* T . c) $H_{40}(j\omega) = F\{\delta P_{40}(T)\}/F\{\delta p_c(T)\}$ and $G_{40}(j\omega)$ for the percentile $n = 40\%$. d) Estimated gain $\ln(k)$ *vs.* (T_{exc}, P_n) . e) Estimated zero fractional-order $\ln(\alpha)$ *vs.* (T_{exc}, P_n) . f) Estimated pole fractional-order $\ln(\beta)$ *vs.* (T_{exc}, P_n) .

Dimension-Independent Convergence Rate for Non-isotropic $(1, \lambda) - ES$

Anne Auger^{1,2}, Claude Le Bris^{1,3}, and Marc Schoenauer²

¹ CERMICS – ENPC

Cité Descartes, 77455 Marne-La-Vallée, France
`{auger,lebris}@cermics.enpc.fr`

² INRIA Rocquencourt, Projet Fractales
BP 105, 78153 LE CHESNAY Cedex, France
`marc.schoenauer@inria.fr`

³ INRIA Rocquencourt, Projet MIC MAC
BP 105, 78153 LE CHESNAY Cedex, France

Abstract. Based on the theory of non-negative super martingales, convergence results are proven for adaptive $(1, \lambda) - ES$ (i.e. with Gaussian mutations), and geometrical convergence rates are derived. In the d -dimensional case ($d > 1$), the algorithm studied here uses a different step-size update in each direction. However, the critical value for the step-size, and the resulting convergence rate do not depend on the dimension. Those results are discussed with respect to previous works. Rigorous numerical investigations on some 1-dimensional functions validate the theoretical results. Trends for future research are indicated.

1 Introduction

Since their invention in the mid-sixties (see the seminal books by Rechenberg [7] and Schwefel [10]), Evolution Strategies have been thoroughly studied from the theoretical point of view.

Early studies on two very particular functions (the *sphere* and the *corridor*) have concerned the progress rate of the $(1 + 1) - ES$, and have lead, by extrapolation to any function, to the famous *one-fifth* rule. The huge body of work by Beyer, including many articles, and somehow summarized in his book [3], has pursued along similar lines, studying more general algorithm, from the full $(\mu \nmid \lambda) - ES$ to the $(\mu/\mu, \lambda) - ES$ with recombination and the $(1, \lambda) - \sigma - SA - ES$ with self-adaptation. However, though giving important insights about the way ES actually work, the study of local progress measures, such as the progress rate, does not lead to global convergence results of the algorithm.

Some global convergence results, together with the associated (geometrical) convergence rates have been obtained for convex functions [8,13], and for a class of function slightly more general than quadratic functions, the so-called $(Q - K) - strongly\ convex$ functions [9]. These latter results deal with the so-called *adaptive* version of evolution strategies, in which the step-size is computed

at each iteration according to some measures on the current population (the terminology used here is taken from [6]) – namely the norm of the gradient of the fitness function.

Note that the results in [13] have been criticized in [4], in which an analytical approach is provided in the case of the sphere function when the step-size is the norm of the parent itself. In that case, the strong law of large number gives an almost sure convergence.

The state-of-the-art in practical ES, however, recommends using *self-adaptive* ES, in which the step-size is adjusted by the evolution itself at the individual level. Whereas of course the results by Beyer on the $(1, \lambda) - \sigma - SA - ES$ do address self-adaptive ES [3], only recently some global convergence results regarding self-adaptive ES-like algorithms were published [5,11]. However, the algorithms studied in those works do not consider the standard normal mutation, but rather use a simplified mutation operator: only a finite number of variation of the step-size are allowed in [5], while [11] considers a uniform mutation. Moreover, these papers only consider the simple and symmetrical function $f(x) = |x|$. Finally, [5] does not give any estimation of the convergence rate, and the proof in [11] relies on a numerical estimate of some inequality – though this might probably be improved in the near future. An important point about these latter two results is that they use the theory of super-martingales [12], a somewhat more sophisticated technique than all previously cited works (with the remarkable exception of [8]).

The same super martingale technique will be used in this paper, to analyze some adaptive ES with Gaussian mutation, in which the step-size is adapted either using the distance to the global optimum or using gradient information about the fitness function, but in a different way than in [13,9]. Moreover, the speed of convergence will also be studied: as in previous relevant work, some geometrical upper-bounds will be derived, and their sharpness will be tested through numerical experiments.

The paper is organized as follows. Next section formally describes the adaptive ES under study. We configure ES with an adaptivity that evolves more deterministically than in standard self adaptive ES (see formula (1) below). Section 3 gives the convergence results and the main ideas of the proofs (due to size limitation, the complete proofs cannot be given here, see [2] for all the details). First, the one-dimensional case is thoroughly studied: in the case of the sphere function analytical results are obtained for the sphere function, before two different ways of adapting the step-size are studied in turn for a more general class of functions. It is indeed to be noted that our proofs and techniques are not restricted to the specific cases we deal with here. Next, the optimality of the critical value of the step size and convergence rate obtained is proved for the sphere function. The case of larger dimension is finally presented. The originality is that we derive estimates of the convergence rate that do not depend on the

dimension. This is done on a specific algorithm where the step-size is adapted independently in each dimension.

In section 4, our results are thoroughly discussed, in the light of previous works on adaptive algorithm (already cited in the Introduction). Section 5 next gives experimental evidences (in one dimension only) that demonstrate the validity of the critical value of the step size and of the convergence rate, for more general functions (such as functions that are neither symmetric (w.r.t. their minimum) nor convex). The article closes with some discussion and trends for future work.

2 Notations and Algorithm

For the sake of simplicity, the results will first be presented in dimension 1. The case of higher dimensions will be introduced in section 3.4. Let f be a real-valued function defined on $\mathbb{R}$ to be minimized. The general adaptive $(1, \lambda)$ -Evolution Strategy algorithm we will consider henceforth is of the form:

$$\begin{cases} X^0 \in \mathbb{R}, \\ X^{n+1} = \arg \min \{f(X^n + \sigma H(X^n)N_i^n), i \in [1, \lambda]\}, \end{cases} \quad (1)$$

where X_n is the random variable modeling the parent at the generation n , $(N_i^n)_{i=1, \dots, \lambda}$ are independent standard normal random variables, $H(x)$ is real-valued function (for conciseness, only two cases will be considered in the following: $H(x) = |x|$ or $H(x) = |f'(x)|$, but other cases, such as $H(x) = |f(x) - f^*|$ can be treated by the same technique, see [1]), and σ is a positive real parameter, often referred to as the *step-size* (or normalized step-size e.g. in [10,3], in the case where $H(x) = |x|$).

This paper is concerned with studying the behavior of algorithm (1), or, more precisely, with addressing the issue of the range of values for σ for which the algorithm converges¹. Moreover, whenever convergence takes place, bounds for the convergence rate will also be sought.

Section 3 gives answers to both questions, first for the sphere function (section 3.1), as exact convergence rates can be easily computed, and then for twice continuously differentiable functions with particular properties in the case $H(x) = |x|$ (section 3.2) and $H(x) = |f'(x)|$ (section 3.3).

3 Convergence Results the $(1, \lambda)$ -ES

3.1 The Sphere Function – Again

The sphere function ($f(X) = |X|^2$) has always been the preferred test function of authors studying the theory of Evolution Strategies [7,10,8,3,4,5,11]. Indeed, when f is the sphere function, many things get simpler, and most quantities of interest can be computed analytically.

¹ Both *almost sure* convergence and convergence in L^p (w.r.t. the norm $\mathbb{E}(|X|^p)^{\frac{1}{p}}$) will be looked at.

For instance, it is clear that both cases $H(x) = |x|$ and $H(x) = |f'(x)|$ behave identically (up to a factor 2). But another important simplification concerns the algorithm itself:

Lemma 1. *For the sphere function, the random variable X^n defined by (1) with $H(x) = |x|$ satisfies,*

$$X^{n+1} = X^n(1 + \sigma Y(\lambda)) \quad (2)$$

where $Y(\lambda)$, the random variable defined by

$$1 + \sigma Y(\lambda) = \arg \min\{(1 + \sigma N_1^n)^2, \dots, (1 + \sigma N_\lambda^n)^2\} \quad (3)$$

does not depend on σ .

A detailed proof, with the exact distribution of $Y(\lambda)$ can be found in [4].

Convergence in L^p . The following theorem is an immediate consequence of Lemma 1:

Theorem 1. *For the sphere function, the random variable X^n defined by (1) with $H(x) = |x|$ satisfies,*

$$\mathbb{E}(|X^n|^p) = \mathbb{E}(|X_0|^p) (\mathbb{E}(|1 + \sigma Y(\lambda)|^p))^n. \quad (4)$$

Hence, the algorithm converges or diverges in L^p norm geometrically. Moreover, there exists a value $\sigma_c(\lambda, p)$ such that X^n converges in L^p norm iff $\sigma \in]0, \sigma_c(\lambda, p)]$. This value is defined by

$$\sigma_c(\lambda, p) = \inf\{\sigma \text{ such that } \mathbb{E}(|1 + \sigma Y(\lambda)|^p) \geq 1\}. \quad (5)$$

Remark 1. It can be proved that $\mathbb{E}(|1 + \sigma Y(\lambda)|^p)$ has a unique minimum w.r.t. σ , which gives the best convergence rate. This minimum $\sigma_s(\lambda, p)$ is thus defined by,

$$\sigma_s(\lambda, p) = \operatorname{argmin}\{\mathbb{E}(|1 + \sigma Y(\lambda)|^p), \sigma \in]0, \sigma_c(\lambda, p)]\}. \quad (6)$$

An alternative view on the progress rate. Interestingly, this result meets early studies of ES [7,10,3] that did look at the *progress rate* φ_p , defined by:

$$\varphi_p(X^n, \sigma, \lambda) = \mathbb{E} \left(\frac{|X^{n+1}|^p - |X^n|^p}{|X^n|^p} \middle| X^n \right), \quad (7)$$

The progress rate measures the expectation of change from one iteration of the algorithm to the next one, conditionally to the current parent X_n : Note that this conditional dependency is often left implicit in the cited works. Those early works determine, for a given λ , the optimal step size σ which minimizes the

progress rate. In general, this quantity depends on the current point X^n and will not be very useful to study the dynamics of the algorithm.

However, in the case of the sphere function, things are different. A direct consequence of Lemma 1 is that for the sphere function with $H(x) = |x|$, the progress rate does not depend on the value of X^n and is hence for instance equal to the value for $X^n = 1$: $(\forall n > 0), \varphi_p(X^n, \sigma, \lambda) = \mathbb{E}(|1 + \sigma Y(\lambda)|^p - 1)$.

Hence, minimizing the progress rate as in [10,3] thus amounts to finding the value of σ such that $\mathbb{E}(|1 + \sigma Y(\lambda)|^p)$ is minimal – and this is exactly the value given by equation (6).

Convergence almost surely. For the almost sure convergence, Lemma 1 and the strong law of large numbers gives the following result (see [4] for more details),

Theorem 2. *Assume that $\mathbb{E}(\ln(|1 + \sigma Y(\lambda)|)) < \infty$. Then, for the sphere function, the random variable X^n defined by (1) with $H(x) = |x|$ satisfies,*

$$\frac{1}{n} \ln(|X^n|) \xrightarrow[n \rightarrow \infty]{} \mathbb{E}(\ln(|1 + \sigma Y(\lambda)|)) \quad \text{almost surely.}$$

Thus the critical value $\sigma_c(\lambda, as)$ is here defined as $\sup\{\sigma \setminus \mathbb{E}(\ln(|1 + \sigma Y(\lambda)|)) < 1\}$.

The following two sections will prove similar results for more general functions, for each of the cases $H(x) = |x|$ and $H(x) = |f'(x)|$.

3.2 Convergence of the $(1, \lambda)$ -ES with $H(x) = |x|$

The case where $H(x) = |x|$ (or $H(x) = |x - x^*|$ for some minimizer x^* of f) is the case with constant (normalized) step-size, as defined for instance in [3]. Though this algorithm has not a practical interest because it supposes that a minimum is already known, it will allow us to develop the technique of analysis to be later applied to the more interesting case $H(x) = |f'(x)|$.

The first step of this analysis consists in finding a value σ_c such that $f(X^n)$ is a super martingale for $\sigma \in]0, \sigma_c[$. The convergence of the processes $f(X^n)$ and X^n will immediately follow (see [12]). For this purpose, we state some assumptions on f :

Assumptions (H1).

- (i) *The function f has a unique global minimizer x^* . Without loss of generality, we assume that $x^* = 0$ and $f(0) = 0$, and therefore $\forall x \in \mathbb{R}, f(x) > 0$.*
- (ii) *The function f is twice continuously differentiable.*
- (iii) *There exists M finite such that, for all $x \in \mathbb{R}$, $|f''(x)| \leq M$.*
- (iv) *There exists $\alpha > 0$ such that, for all $x \neq 0$, $|\frac{f'(x)}{x}| \geq \alpha > 0$*

Remark 2. All our proofs (see [2]) still go through when the process X^n is replaced by $\inf(\sup(X^n, -A), A)$ in equation (1) for some large A . Such a modification is an easy trick to render Assumptions (H1) easier to fulfill.

Remark 3. Assumption (H1) above implies that f is monotonously decreasing on $\mathbb{R}^-$ and increasing on $\mathbb{R}^+$.

In the sequel, $\mathcal{F}_n$ denotes the filtration adapted to the process $f(X^n)$.

Lemma 2. Assume $\lambda \leq 2$. Let g be defined by

$$g(\sigma, \lambda, \alpha, M) = \mathbb{E} \left(\min_{1 \leq i \leq \lambda} \left(\alpha N^i + \sigma \frac{M}{2} (N^i)^2 \right) \right), \quad (8)$$

and let $\sigma_c(\lambda, \alpha, M)$ be the solution of

$$g(\sigma_c(\lambda, \alpha, M), \lambda, \alpha, M) = 0 \quad (9)$$

Assume f satisfies Assumption (H1), $f(X^n)$ is a $\mathcal{F}_n$ -super martingale² for $0 \leq \sigma \leq \sigma_c(\lambda, \alpha, M)$.

Remark 4. The value $\sigma_c(\lambda, \alpha, M)$ defined by equation (9) always exists and is unique for $\lambda \geq 2$, α and M given, because $g(\sigma, \lambda, \alpha, M)$ is a strictly increasing and continuous function w.r.t. σ , and satisfies $g(0, \lambda, \alpha, M) < 0$ and $\lim_{\sigma \rightarrow +\infty} g(\sigma, \lambda, \alpha, M) = +\infty$.

Key point of the proof. The demonstration of this result relies on the following inequality, based on Taylor formula:

$$\mathbb{E}(f(X^{n+1})|\mathcal{F}_n) \leq f(X^n) + \sigma |X^n|^2 g(\sigma, \lambda, \alpha, M) \text{ a.s.} \quad (10)$$

Convergence result. From this Lemma, the theory of non-negative super martingale [12] gives the following theorem.

Theorem 3. Assume $\lambda \geq 2$, assume f satisfies Assumption (H1), and $\sigma \in]0, \sigma_c(\lambda, \alpha, M)[$ with $\sigma_c(\lambda, \alpha, M)$ defined by equation (9), then, when n goes to $+\infty$, $f(X^n)$ converges to 0, both almost surely and in L^1 , and X^n converges to 0 both almost surely and in L^2 .

Convergence Speed

Theorem 4. Assume $\lambda \geq 2$, assume f satisfies Assumptions (H1), and that $\sigma \in]0, \sigma_c(\lambda, \alpha, M)[$, with $\sigma_c(\lambda, \alpha, M)$ defined by (9), then $f(X^n)$ converges geometrically to 0 in the following senses:

- (i) **(Convergence a.s.):** $\frac{f(X^n)}{(1 + \sigma C g(\sigma, \lambda, \alpha, M))^n}$ converges to some random variable Y ,
- (ii) **(Convergence in L^1):** $\mathbb{E}(f(X^n)) \leq (1 + \sigma C g(\sigma, \lambda, \alpha, M))^n \mathbb{E}(f(X^0))$,

where $C = \frac{2}{M}$ and M is defined by (H1)(iii). In addition, the best convergence rate is reached for $\sigma = \sigma_s(\lambda, \alpha, M)$ where $\sigma_s(\lambda, \alpha, M)$ is the unique value of σ that minimizes $1 + \sigma C g(\sigma, \lambda, \alpha, M)$.

² Z^n is a super martingale if it satisfies $\mathbb{E}(Z^{n+1}|\mathcal{F}_n) \leq Z^n$

3.3 Convergence of the $(1, \lambda)$ -ES with $H(x) = |f'(x)|$

The general outline of the demonstration in this case is the same as in the previous section: First, find a value σ_c such that $f(X^n)$ is a supermartingale for $\sigma \in]0, \sigma_c[$. Then, derive the convergence and the speed of convergence of $f(X^n)$.

Contrary to the previous section, unimodality is not mandatory in the present section to obtain the convergence result *per se*. But, some local convexity is needed to derive the convergence rate. We consider the following assumptions,

Assumption (H2).

- (i) The function f is bounded from below (say by zero) and is twice continuously differentiable.
- (ii) There exists M finite such that, for all x , $|f''(x)| \leq M$.

Remark 5. Once again, using the truncation trick mentioned in Remark 2 weakens this assumption which is then satisfied for every C^2 function.

Lemma 3. Assume $\lambda \geq 2$. Let h be defined by

$$h(\sigma, \lambda, M) = \mathbb{E} \left(\min_{1 \leq i \leq \lambda} \left(N^i + \sigma \frac{M}{2} (N^i)^2 \right) \right) \quad (11)$$

and let $\sigma'_c(\lambda, M)$ be the solution of

$$h(\sigma'_c(\lambda, M), \lambda, M) = 0 \quad (12)$$

Then, if f satisfies Assumption (H2), $f(X^n)$ is a $\mathcal{F}_n$ -super martingale for $0 \leq \sigma \leq \sigma'_c(\lambda, M)$.

Remark 6. The proof of the existence of $\sigma'_c(\lambda, M)$ is exactly the same as in Remark 4.

Key point of the proof. Once again, the demonstration of the above result relies on the following inequality.

$$\mathbb{E}(f(X^{n+1})|\mathcal{F}_n) \leq f(X^n) + \sigma |f'(X^n)|^2 h(\sigma, \lambda, M) \text{ a.s.} \quad (13)$$

Convergence result. A straightforward corollary of this Lemma is that $f(X^n)$ converges almost surely. The following theorem then gives the convergence of $f'(X^n)$.

Theorem 5. Assume f satisfies Assumption (H2). Assume $\lambda \geq 2$ and $\sigma \in]0, \sigma'_c(\lambda, M)[$. Then $f'(X^n)$ converges to 0 in L^2 . If we moreover assume that $f(X^n)$ is bounded then $f'(X^n)$ converges almost surely.

Remark 7. If we moreover suppose that f is unimodal and that the only minimum is 0, then the algorithm converges globally: $f(X^n)$ converges to 0 a.s.

Convergence speed. An additional hypothesis, somewhat connected to convexity, is now needed to estimate the convergence speed. Before we state it, we set by convention that $\inf_{\mathbb{R}} f = 0$, otherwise $f(x)$ should be replaced by $f(x) - \inf_{\mathbb{R}} f$ in the assumption below.

Assumption (H3). *There exists $C > 0$ such that $\inf_{\mathbb{R}} \frac{|f'(x)|^2}{f(x)} \geq C$.*

Remark 8. Example of non-trivial functions satisfying both Assumptions (H2) and (H3) will be given in the numerical experiments (see section 5).

Theorem 6. *Assume $\lambda \geq 2$. Assume f satisfies Assumptions (H2)(H3) and that $\sigma \in]0, \sigma'_c(\lambda, M)[$. Then $f(X^n)$ converges geometrically to 0 at the rate $(1 + \sigma Ch(\sigma, \lambda, M))$ both almost surely and L^1 (in the sense of Theorem 4, and with the constant C defined in (H3)). The best convergence rate is reached for $\sigma = \sigma'_s(\lambda, M)$ where $\sigma'_s(\lambda, M)$ minimizes $1 + \sigma Ch(\sigma, \lambda, M)$.*

On the optimality of the general estimates when applied to the sphere function. Going back to the sphere function, the values in Assumption (H1),(H2) and (H3) are $M = 2$, $\alpha = 2$ and $C = 4$, and straightforward calculus gives $\mathbb{E}(|1 + \sigma Y(\lambda)|^2) = \mathbb{E}(\min_{1 \leq i \leq \lambda} (1 + \sigma N_i)^2) = 1 + \sigma g(\sigma, \lambda, 2, 2) = 1 + 2\sigma h(\frac{\sigma}{2}, \lambda, 2)$. It is thus easy to show that the critical values given in Theorems 3, 4, 5 and 6 are the optimal values given by equations (5) and (6).

3.4 Results in Higher Dimensions

The algorithm defined in equation 1 must be slightly modified when going to dimension $d > 1$. The general form of the *non-isotropic* ES algorithm considered here is:

$$\begin{cases} X^0 \in \mathbb{R}^d, \\ X^{n+1} = \arg \min \{f(X^n + \sigma(H_k(X^n)N_k^{n,i}))_{k \in [1,d], i \in [1,\lambda]}\}, \end{cases} \quad (14)$$

where X_n is the random variable modeling the parent at the generation n , $(N_k^{n,i}, k \in [1, d], i \in [1, \lambda])$ are independent standard normal random variables, and $H_k(x), k \in [1, d]$ are d real-valued functions. Different step-sizes are here applied to the different directions, similarly with what can be done as far as self-adaptation is concerned [10].

Only the case of practical interest where $H_k(x) = \frac{\partial f(x)}{\partial x_k}$ will be considered here. The situation is then similar to that studied in section 3.3. Assumption (H2)(ii) then becomes, $\|D^2 f\|_d = \sup_{x \in \mathbb{R}^d} \frac{\|D^2 f x\|_d}{\|x\|_d} \leq M$. Similar derivations allow one to prove the following equation which is the equivalent of equation (13),

$$f(X^{n+1}) \leq f(X^n) + \sigma \sum_{k=1}^d \left(\frac{\partial f(X^n)}{\partial x_k} \right)^2 [N_k^{n,i} + \frac{M}{2} \sigma (N_k^{n,i})^2] \quad a.s.$$

from which derives exactly the same result than that of Lemma 3. In particular, the critical value σ_c , below which convergence takes place, is again defined by equations (11) and (12). The more remarkable fact here is that this critical value (and hence the convergence rate that comes with it) does not depend on the dimension!

4 Discussion

This section will discuss the results of previous section in the light of past related work from the literature.

First, it should be clear that only works proposing global convergence results are relevant for comparison here, as opposed to all work studying local convergence (see section 3.1 for a link with those works).

The work whose results are most similar to the ones presented here are by far Rudolph's work, either using also super martingale [8], or somehow simplified and based on order statistics [9]. There are however quite a few differences.

First, Rudolph's results are based on some strong convexity of function f – but it is fair to say that on the other hand, he only needs f to be differentiable once – whereas convexity is not required here for the convergence result, and, as expected, only weak convexity is necessary to obtain the geometrical convergence rate.³

Second, whereas Rudolph chooses all offspring uniformly on some hypersphere (or radius σ), the algorithm considered here uses the “true” Gaussian mutation. A common argument is that both mutations behave similarly in high dimension. However, when it comes to theoretical results, such a consideration is of no help. Indeed, the method used by Rudolph based on order statistics [9] can also be applied with Gaussian mutation, and gives the same kind of convergence result: there exists a critical value σ_c such that whenever σ lies in $]0, \sigma_c[$ the algorithm converges. Unfortunately, this constant σ_c is then defined as $2 \frac{\mathbb{E}(N^{\lambda:\lambda})}{ME((N^{\lambda:\lambda})^2)}$, where $N^{\lambda:\lambda}$ is the λ^{th} order statistics for standard normal random variables. The problem is that this quantity is a very poor upper bound: for instance, it decreases for large values of λ , making the result almost useless.

A noticeable difference with Rudolph's algorithm in [9] lies in the case where the dimension is greater than 1: the offspring of parent X^n in Rudolph's algorithms are chosen using $H(x) = \sigma \|\nabla f(x)\| N$ (notation of equation (1)), for some vector of standard normal random variables N . The approach proposed here is different (see section 3.4), and the results are indeed far more appealing: the upper-bound geometrical rate obtained by Rudolph goes to 1 when the dimension goes to ∞ (despite the fact that he does not use Gaussian mutation), while the one proposed here does not depend on the dimension. However, the

³ In this line, we would like to mention that there seems to be a lot of room for improvement in the proofs we present here (see [1]). Assumptions of regularity and convexity are likely to be relaxed. We are currently working on such extensions. Definite conclusions are however yet to be obtained

gap between the two approaches remains open, as it has not been possible up to now to analyze the algorithm 1 with Rudolph's H function.

5 Numerical Experiments

All numerical experiments presented in the sequel are based on the Monte Carlo approximation of the expectation of a random variable. The expectation $\mathbb{E}(Z)$ of a random variable Z is approximated by $\frac{1}{K} \sum_{k=1}^K Z_k$, where Z_k are K independent random variables with the same law than Z . Then, for instance, from the central limit theorem, for large values of K ($K = 1500$ in all numerical experiments presented here), with probability 0.95,

$$\mathbb{E}(Z) \in \left[\frac{1}{K} \sum_{k=1}^K Z_k - \sqrt{\text{Var} Z} \frac{1.96}{\sqrt{K}}, \frac{1}{K} \sum_{k=1}^K Z_k + \sqrt{\text{Var} Z} \frac{1.96}{\sqrt{K}} \right] \quad (15)$$

5.1 Computation of the Constants

The Monte-Carlo method described above has been used to compute approximate values of the constants σ_c and σ_s from section 3.

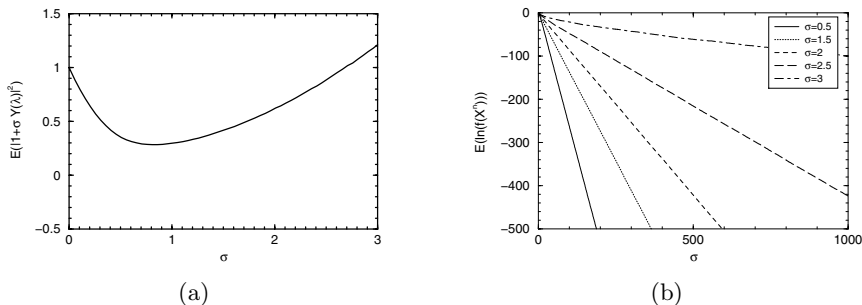


Fig. 1. (a) $\mathbb{E}(|1 + \sigma Y(4)|^2)$ vs σ – see equation (3). (b) $\mathbb{E}(\ln(f(X^n)))$ with respect to the number of generations n .

A first example is given by the plot of $\mathbb{E}(|1 + \sigma Y(\lambda)|^2)$ against σ for the sphere function on Figure 1 (a), for $\lambda = 4$. The limit value of σ for which $\mathbb{E}(|1 + \sigma Y(\lambda)|^2) \leq 1$ is $\sigma_c(\lambda, 2) = 2.7$, and the corresponding minimal value for $\mathbb{E}(|1 + \sigma Y(\lambda)|^2)$ is $\sigma_s(\lambda, 2) \approx 0.8$. Note that this method allows us to plot the progress rate (3.1) for any dimension d , as in [10,3], without any assumption regarding $d \rightarrow +\infty$.

5.2 Optimality of the Constants

The idea here is to compare the constants σ_c , σ_s , σ'_c and σ'_s for some functions that are not quadratic, in order to test their optimality (whereas these constants

are known to be optimal in the case of quadratic functions, see section 3.3, where optimal means here that theses constants are the limit values between convergence and divergence.)

First, we need to circumvent a difficulty. Indeed, when evaluating $\mathbb{E}(f(X^n))$ with the Monte Carlo method, the relative error given by the Central Limit Theorem ($\sqrt{\text{Var}(f(X^n))} \frac{1.96}{\sqrt{K\mathbb{E}(f(X^n))}}$) grows geometrically with the number of generations n (the exact computation can be made easily on the sphere function). On the other hand, that of evaluating $\mathbb{E}(\ln(f(X^n)))$ decreases in $\frac{1}{\sqrt{n}}$. Hence, all numerical tests have been performed on the process $\ln(f(X^n))$. This fact in turn requires to come back to the convergence analysis. Indeed, it turns out that the arguments used to treat the minimization of f also hold for the minimization of $\ln(f)$. Of course, since the a.s. convergence of $f(X^n)$ implies that of $\ln(f(X^n))$, we know sufficient conditions for such a convergence. But, more than that, $\ln(f(X^n))$ converges in the same fashion and under the same conditions as $f(X^n)$ with an arithmetic rate replacing the geometric rate of Theorems 4 and 6.

Only numerical results concerning the case $H(x) = |f'(x)|$ will be shown here.

The functions f_M , defined by equation (16) below, are examples among the class of non symetrical functions satisfying both Assumptions (H2) and (H3) that will be used for all experiments (where $M > 0$ is the value used in Assumption (H2)-(ii)).

$$f_M(x) = \frac{M}{2} \begin{cases} x^2 & \text{if } x < 0 \\ x \arctan(x) & \text{if } x > 0 \end{cases} \quad (16)$$

Figure 1 (b) plots $\frac{1}{K} \sum_{k=1}^K \ln(f_2(X_k^n))$ against the number of generations for

different values of σ . The relative error $\frac{\mathbb{E}(\ln(f_2(X^n))) - \frac{1}{K} \sum_{k=1}^K \ln(f_2(X_k^n))}{\frac{1}{K} \sum_{k=1}^K \ln(f_2(X_k^n))}$ given by equation (15), is here bounded by 0.01. This corroborates the linear rate of convergence predicted by our theoretical study.

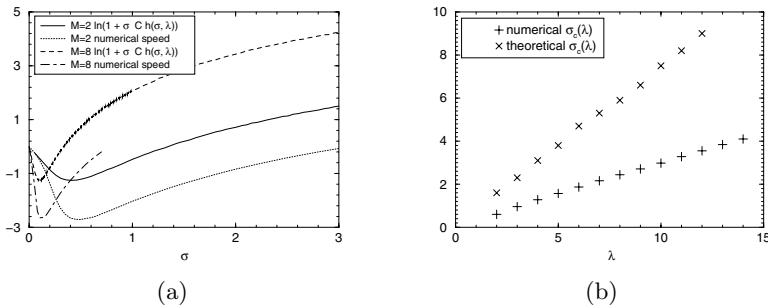


Fig. 2. (a) Theoretical and numerical speeds of convergence for functions f_2 and f_8 . (b) Numerical $\sigma'_{cnun}(\lambda)$ and theoretical $\sigma'_c(\lambda)$.

Figure 2 (a) plots the slopes of those linear functions (determined using linear regressions), and the theoretical values $\sigma Ch(\sigma, \lambda, \alpha, M)$, for $\lambda = 4$ and for both functions f_2 and f_8 . Both curves have the same shapes. Moreover, on these functions, the theoretical bounds indeed underestimate the threshold, as expected.

Studying only function f_2 , the intersection between the theoretical curve and the x-axis gives a numerical approximation $\sigma'_c(4) \approx 1.4$ of the theoretical value $\sigma_c(4)$ – and in the sequel, $\sigma'_{cnum}(4)$ will denote the intersection between the experimental curve and the x-axis. From Figure 2 (a), it comes that $\sigma'_{cnum}(4) \approx 3.1$. Defining similarly $\sigma'_{snum}(4)$ as the critical point of the numerical curve, it may also be noted on the same Figure that $\sigma'_s(4) \leq \sigma'_{snum}(4)$.

It may be observed from the same Figure 2 (a) that both theoretical and numerical curves present the same scaling transformation when M is increased – even though the theoretical bound still seems pessimistic. Last, Figure 2 (b) shows, for function f_2 , the numerical $\sigma'_{cnum}(\lambda)$ and theoretical $\sigma'_c(\lambda)$ for $\lambda = 2, \dots, 13$. Both are linear increasing functions in λ .

6 Conclusions and Perspectives

Convergence results and geometrical convergence rates for adaptive $(1, \lambda) - ES$ have been proved for a sub-class of C^2 functions. The optimality of the critical value for the step size and the resulting convergence rate have been proved for the sphere function and numerical experiments have demonstrated their validity for more general functions. The extension of the results to the d -dimensional case with a *non-isotropic* ES algorithm (14) leads to a critical value of the step-size and a convergence rate that are independent of the dimension, improving over previous work. On-going work is concerned with relaxing the regularity and convexity assumptions: it should be possible to nevertheless obtain similar results for convergence and convergence rates. In addition one can envision the extension to a more practically useful algorithm, where the step-size is adapted proportionally to $|f(x) - f^*|$ (where f^* is the value at the global optimum). However, the d -dimension case of this latter algorithm will probably lead to dimension-dependent convergence rate. Finally, similar analysis should be possible for self-adaptive $(1, \lambda) - ES$, but probably requiring regularity assumptions on the objective function.

References

1. A. Auger. *ES, théorie et applications au contrôle en chimie*. PhD thesis, Université Paris 6, in preparation.
2. A. Auger, C. Le Bris, and M. Schoenauer. Rigorous analysis of some simple adaptive es. *Technical Report INRIA*, <http://cermics.enpc.fr/~auger/>.
3. H.-G. Beyer. *The Theory of Evolution Strategies*. Springer, Heidelberg, 2001.
4. A. Bienvenüe and O. François. Global convergence for evolution strategies in spherical problems: Some simple proofs and pitfalls. *Submitted*, 2001. <http://www-lmc.imag.fr/lmc-sms/Alexis.Bienvenue/>.

5. J.M DeLaurentis, L. A. Ferguson, and W.E. Hart. On the convergence properties of a simple self-adaptive evolutionary algorithm. In W.B. Langdon & al., editor, *Proceedings of the Genetic and Evolutionary Conference*, pages 229–237. Morgan Kaufmann, 2002.
6. A. E. Eiben, R. Hinterding, and Z. Michalewicz. Parameter control in Evolutionary Algorithms. *IEEE Transactions on Evolutionary Computation*, 3(2):124, 1999.
7. I. Rechenberg. *Evolutionstrategie: Optimierung Technischer Systeme nach Prinzipien des Biologischen Evolution*. Fromman-Holzboog Verlag, Stuttgart, 1973.
8. G. Rudolph. Convergence of non-elitist strategies. In Z. Michalewicz, J. D. Schaffer, H.-P. Schwefel, D. B. Fogel, and H. Kitano, editors, *Proceedings of the First IEEE International Conference on Evolutionary Computation*, pages 63–66. IEEE Press, 1994.
9. G. Rudolph. Convergence rates of evolutionary algorithms for a class of convex objective functions. *Control and Cybernetics*, 26(3):375–390, 1997.
10. H.-P. Schwefel. *Numerical Optimization of Computer Models*. John Wiley & Sons, New-York, 1981. 1995 – 2nd edition.
11. M.A. Semenov. Convergence velocity of an evolutionary algorithm with self-adaptation. In W.B. Langdon & al., editor, *Proceedings of the Genetic and Evolutionary Conference*, pages 210–213. Morgan Kaufmann, 2002.
12. D. Williams. *Probability with Martingales*. Cambridge University Press, Cambridge, 2000.
13. G. Yin, G. Rudolph, and H.-P Schwefel. Analysing $(1, \lambda)$ evolution strategy via stochastic approximation methods. *Evolutionary Computation*, 3(4):473–489, 1996.

The Steady State Behavior of $(\mu/\mu_I, \lambda)$ -ES on Ellipsoidal Fitness Models Disturbed by Noise^{*}

Hans-Georg Beyer and Dirk V. Arnold

Department of Computer Science XI,
University of Dortmund, D-44221 Dortmund, Germany
{hans-georg.beyer, dirk.arnold}@cs.uni-dortmund.de

Abstract. The method of differential-geometry is applied for deriving steady state conditions for the $(\mu/\mu_I, \lambda)$ -ES on the general quadratic test function disturbed by fitness noise of constant strength. A new approach for estimating the expected final fitness deviation observed under such conditions is presented. The theoretical results obtained are compared with real ES runs showing a surprisingly excellent agreement.

1 Introduction

Understanding the impact of noise on the optimization behavior of evolutionary algorithms (EAs) is of great interest: There is a certain belief that EAs are especially good at coping with noisy information due to the use of a population of candidate solutions. There is empirical evidence as well as some theoretical support for this belief [3]. Furthermore, noise models on the level of the control parameters to be optimized, also called *actuator noise models* in [11], are of interest in the context of *robust optimization* [17,18,12].

While there is a need for a deeper understanding of the behavior of EAs on such noisy problems, a theoretical analysis is still at its beginning. Up to now, only the behavior of evolution strategies (ES) on the sphere model has been analyzed [1]. Performing similar analyses on other test functions still remain to be done. However, such analyses starting from scratch are expensive. Therefore, it would be desirable to use results obtained from the sphere theory as a starting point for deriving statements on the behavior of ES on other test functions.

This article is exactly in that spirit by taking up the thread from [8] where $(1, \lambda)$ -ES has been considered. First, it applies the differential-geometrical model [7] in order to derive the condition for the zero progress rate in *recombinant* ES on general quadratic models disturbed by fitness noise of constant strength. Second, it provides a new and simple but surprisingly accurate method for estimating the expected final fitness deviation observed under such conditions.

The paper is organized as follows. After introducing the general quadratic test function disturbed by fitness noise we will determine the steady state condition

^{*} This work was supported by the Deutsche Forschungsgemeinschaft (DFG), grant Be1578/6-3, and by the Collaborative Research Center (SFB) 531.

starting from the standard noisy sphere model. Then we will provide the new approach for determining the expected final fitness deviation. The predictions of this model will be compared with $(\mu/\mu_I, \lambda)$ -ES runs. In the concluding section an outlook will be given emphasizing the potential of the methods presented.

2 The Steady State Condition of $(\mu/\mu_I, \lambda)$ -ES on Noisy Quadratic Functions

2.1 The General Quadratic Fitness Noise Model

We consider the general quadratic fitness model based on the quality function

$$Q_g(\mathbf{y}) := \mathbf{b}^T \mathbf{y} - \mathbf{y}^T \mathbf{Q} \mathbf{y} \quad (1)$$

where $\mathbf{b}$ and $\mathbf{y}$ are N -dimensional real-valued vectors and $\mathbf{Q}$ is a symmetric, (w.l.o.g.) positive definite matrix. Given an object vector $\mathbf{y}$ the actually observed objective value, i.e. fitness F_{ng} , is disturbed by Gaussian noise of strength σ_δ

$$F_{ng}(\mathbf{y}) := Q_g(\mathbf{y}) + \mathcal{N}(0, \sigma_\delta^2). \quad (2)$$

It is assumed that σ_δ is constant for each single generation. That is, all offspring within the same generation experience the same noise strength.

2.2 Determining the Steady State Condition

It is a common phenomenon that EA optimizing fitness functions disturbed by noise of constant strength exhibit some kind of steady state behavior (after a certain transient time of approaching the optimum) which is – on average – away from the optimal solution [6]. If this steady state regime has been reached, the expected fitness improvement will be zero. In order to determine the steady state condition of this behavior, we will reconsider the standard noisy sphere model and apply the differential-geometrical model [7] to it.

Results from the Sphere Model Theory. The qualitative properties of an ES can be characterized by evolution criteria [7, p. 90] which describe the approach toward the optimum in terms of inequalities in the space of the endogenous strategy parameters such as the mutation strength and the noise strength. This concept has been developed for the $(1, \lambda)$ -ES on the noisy sphere model

$$F_{nsp}(\mathbf{y}) := f(\|\mathbf{y}\|) + \mathcal{N}(0, \sigma_\delta^2), \quad f = f(r) \text{ monotonic function}, \quad (3)$$

in [5] and recently extended for the $(\mu/\mu_I, \lambda)$ -ES in [2]. The asymptotically correct $(N \rightarrow \infty)$ evolution criterion reads

$$\sigma_\delta^{*2} + \sigma^{*2} \leq (2\mu c_{\mu/\mu_I, \lambda})^2 \quad (4)$$

where

$$\sigma^* = \sigma \frac{N}{R} \quad \text{and} \quad \sigma_\delta^* = \sigma_\delta \frac{N}{R|f'|} \quad \text{with} \quad f' = \left. \frac{df}{dr} \right|_{r=R} \quad (5)$$

are the normalized mutation strength (isotropic mutations assumed) and the normalized noise strength, respectively. $R := \|\mathbf{y}_p\|$ is the distance of the parental centroid (center of mass of the μ parents $\mathbf{y}_m$) to the optimum, and $c_{\mu/\mu, \lambda}$ is the progress coefficient (see e.g. [7, p. 247]).

Criterion (4) characterizes those endogenous strategy parameter states which guarantee convergence toward the optimum. That is, when the “ $<$ ” relation is fulfilled in (4), the expected R value decreases from one generation to the next. Criterion (4) necessarily implies

$$\sigma_\delta^* \leq 2\mu c_{\mu/\mu, \lambda}. \quad (6)$$

Using (6) and the σ_δ^* definition in (5), the evolution criterion becomes

$$R|f'| \geq \frac{\sigma_\delta N}{2\mu c_{\mu/\mu, \lambda}}. \quad (7)$$

If the equal sign holds in (4) and (6, 7), respectively, the expected R value remains constant (from one generation to the next) and the evolution stagnates (on average). The latter case is of particular interest because it appears as the steady state of the ES working in a fitness environment with constant noise strength $\sigma_\delta = \text{const}$. Considering the special case

$$f(r) = \beta r^\alpha, \quad (8)$$

(7) can be solved for R yielding for the steady state

$$R \geq \sqrt[\alpha]{\frac{\sigma_\delta N}{2\alpha|\beta|\mu c_{\mu/\mu, \lambda}}} =: R_\infty. \quad (9)$$

It appears that R_∞ in (9) provides a good approximation for the expected steady state value of R in ES runs. This is so because the standard mutation strength adaptation techniques yield comparatively small normalized mutation strengths σ^* . This holds for the mutative σ self-adaptation ES (σ SA-ES) as well as for the cumulative step-size adaptation ES (CSA-ES).

Distorting the Sphere – the General Quadratic Model. In order to transfer the idea behind the evolution criterion (7) to the general quadratic case (2), we have to introduce the generalized quantities for R and f' . This step is by analogy. A mathematically rigorous proof for the correctness of this step is still pending, however, there is experimental evidence for its accuracy (see Section 2.4).

The analogue of the absolute value of the first derivative f' in the $\mathbb{R}^N$ space is the length of the gradient. Similarly, the radius R must be replaced by the differential-geometrical mean radius $\bar{R}$. Calculating the gradient in (1) yields

$$\nabla Q_g = \mathbf{b} - 2\mathbf{Q}\mathbf{y}. \quad (10)$$

Since the optimal state $\hat{\mathbf{y}}$ is given by $\nabla Q_g = \mathbf{0}$, we obtain from (10) $\mathbf{b} = 2\mathbf{Q}\hat{\mathbf{y}}$ and therefore we get

$$\nabla Q_g = 2\mathbf{Q}(\hat{\mathbf{y}} - \mathbf{y}) =: \mathbf{a}. \quad (11)$$

Using the mean radius formula from [7, p.46], one gets

$$\bar{R} = \frac{N-1}{2} \frac{\|\mathbf{a}\|}{\left| \text{Tr}[\mathbf{Q}] - \frac{\mathbf{a}^T \mathbf{Q} \mathbf{a}}{\|\mathbf{a}\|^2} \right|}. \quad (12)$$

In order to obtain the necessary evolution criterion we now substitute $\|\nabla Q_g\|$ for $|f'|$ and $\bar{R}$ for R in (7). After rearranging terms and considering $N \rightarrow \infty$, the evolution criterion reads

$$\frac{\|\mathbf{a}\|^2}{\left| \text{Tr}[\mathbf{Q}] - \frac{\mathbf{a}^T \mathbf{Q} \mathbf{a}}{\|\mathbf{a}\|^2} \right|} \geq \frac{\sigma_\delta}{\mu c_{\mu/\mu, \lambda}}. \quad (13)$$

Provided that the Rayleigh quotient $\mathbf{a}^T \mathbf{Q} \mathbf{a} / \|\mathbf{a}\|^2$ can be neglected, this expression can be further simplified using (11). One obtains

$$\|\mathbf{Q}(\hat{\mathbf{y}} - \mathbf{y})\|^2 \geq \frac{\sigma_\delta \text{Tr}[\mathbf{Q}]}{4\mu c_{\mu/\mu, \lambda}}. \quad (14)$$

Unlike the sphere model where the steady state is characterized by a constant (expected) residual distance to the optimal state, the general case is characterized by $\mathbf{y}$ -states located on an ellipsoidal hypersurface (equal sign in (14)). It is important to realize that these ellipsoidal hypersurfaces are *not* geometrically similar to the ellipsoid defined by $Q_g(\mathbf{y}) = \text{const.}$ (in contrast to the sphere).

2.3 Estimating the Expected Stationary Fitness Error

As we have seen in Section 2.2, noisy fitness information implies a localization error of the optimizer in the $\mathbf{y}$ (object) parameter space. Therefore, choosing a parental state $\mathbf{y}_p$ produced by the ES after reaching the vicinity of the steady state regime results in a $\mathbf{y}_p \neq \hat{\mathbf{y}}$. Thus, the actually obtained *undisturbed* objective function value $Q = Q_g(\mathbf{y}_p)$ will also deviate from the optimum value $\hat{Q} = Q_g(\hat{\mathbf{y}})$. Since $\Delta Q := \hat{Q} - Q_g(\mathbf{y}_p)$ is a random variate one can ask for its expected value and its variance.

A first attempt for estimating ΔQ by neglecting its random character has been presented in [8]. The idea was to use the respective stationarity condition (similar to Eq. (14) with $\mu = 1$) as a constraint on the optimization of Q_g given by (1). While this approach yielded a first rough lower bound on ΔQ , the predicted strong dependency of the Q deviation on the largest eigenvalue of $\mathbf{Q}$ and $\text{Tr}[\mathbf{Q}]$ was not observed in experiments. Astonishingly, one observed ΔQ values the expected values of which were almost independent of the $\mathbf{Q}$ matrix.

We will now present an approach for estimating the expected value of ΔQ in accordance with the experimental observations mentioned above. In order to simplify the formulae, we will first switch to an appropriate coordinate system by performing a principal axes transformation. Let $\mathbf{e}_i$ ($i = 1, \dots, N$) be the normalized eigenvectors and q_i the corresponding eigenvalues of $\mathbf{Q}$. Eq. (1) can be transformed using the completeness condition $\mathbf{1} = \sum_{i=1}^N \mathbf{e}_i \mathbf{e}_i^T$ into

$$Q_g(\mathbf{y}) := \sum_{i=1}^N (b_i y_i - q_i y_i^2) \quad (15)$$

with $b_i = \mathbf{e}_i^T \mathbf{b}$ and $y_i = \mathbf{e}_i^T \mathbf{y}$. Performing quadratic completion in (15) yields

$$Q_g(\mathbf{y}) := \sum_{i=1}^N \frac{b_i^2}{4q_i} - \sum_{i=1}^N q_i \left(y_i - \frac{b_i}{2q_i} \right)^2. \quad (16)$$

One can easily prove that the optimum state of (15) is given by

$$\hat{y}_i = \frac{b_i}{2q_i} \quad \text{and} \quad \hat{Q} = \max[Q_g] = \sum_{i=1}^N \frac{b_i^2}{4q_i}. \quad (17)$$

Thus, using Eq. (16) ΔQ can be expressed in terms of

$$\Delta Q = \hat{Q} - Q_g(\mathbf{y}) = \sum_{i=1}^N q_i (y_i - \hat{y}_i)^2. \quad (18)$$

As one can see, the principal axes transformation decomposes ΔQ in a sum of N independent fitness contributions f_i

$$\Delta Q = \sum_{i=1}^N f_i \quad \text{where} \quad f_i = q_i (y_i - \hat{y}_i)^2. \quad (19)$$

Using the same transformation, the evolution criterion (14) can be expressed similarly. Since $\|\mathbf{Q}(\hat{\mathbf{y}} - \mathbf{y})\|^2 = (\hat{\mathbf{y}} - \mathbf{y})^T \mathbf{Q}^2 (\hat{\mathbf{y}} - \mathbf{y})$ we immediately obtain

$$\|\mathbf{Q}(\hat{\mathbf{y}} - \mathbf{y})\|^2 = \sum_{i=1}^N q_i^2 (y_i - \hat{y}_i)^2 \quad (20)$$

and therefore with (14)

$$\sum_{i=1}^N q_i^2 (y_i - \hat{y}_i)^2 \geq \frac{\sigma_\delta \text{Tr}[\mathbf{Q}]}{4\mu c_{\mu/\mu, \lambda}}. \quad (21)$$

We are now at that position where we can formulate the core ideas for the derivation of the expected Q deviation. At first we note that condition (21) does

hold for all states y_i . Since y_i are random variates, we can take the expected value in (21) leading to

$$\sum_{i=1}^N q_i^2 \mathbb{E}[(y_i - \hat{y}_i)^2] \geq \frac{\sigma_\delta \text{Tr}[\mathbf{Q}]}{4\mu c_{\mu/\mu, \lambda}}. \quad (22)$$

Provided that $\hat{y}_i = \mathbb{E}[y_i]$, the $\mathbb{E}[(y_i - \hat{y}_i)^2]$ expressions can be interpreted as the variances of the respective y_i variates. This is admissible for the steady state because of symmetry of the y_i states in (21). In other words, after reaching the vicinity of the steady state, the y_i fluctuate around the optimizer state $\hat{y}_i$.

Consider the expected value of ΔQ . Using (19) we have

$$\mathbb{E}[\Delta Q] = \sum_{i=1}^N \mathbb{E}[f_i] \quad \text{where} \quad \mathbb{E}[f_i] = q_i \mathbb{E}[(y_i - \hat{y}_i)^2]. \quad (23)$$

Now comes the crucial assumption which is quite similar to the *equipartition theorem* in statistical thermodynamics: Each degree of freedom in (23) contributes on average the same effect to the whole system. That is,

$$\text{EQUIPARTITION ASSUMPTION: } \forall i, j : \mathbb{E}[f_i] = \mathbb{E}[f_j]. \quad (24)$$

It is quite clear that this assumption can only hold under equilibrium conditions, i.e. after reaching the steady state. However, considering the steady state, it should also be clear that the $\mathbb{E}[f_i] = \mathbb{E}[f_j]$ assumption is a natural one: First, the mutations generating new y_i states are not directed. Second, selection only “sees” the whole fitness. Therefore it cannot prefer a specific f_i degree and the f_i degrees fluctuate independently of each other. If a specific f_i degree dominated the others (i.e. having had a much larger $\mathbb{E}[f_i]$) then this would mean that its f_i contributions to the actual Q values were much higher. Such states, however, are likely to go extinct because selection prefers $\mathbf{y}$ realizations with lower Q values. Third, since the y_i states fluctuate independently around the optimum state $\hat{y}_i$, selection does not prefer any specific $\mathbf{y}$ direction (if this were not the case we would not be at the steady state but – on average – still move through the search space in a specific direction).

Accepting the validity of (24) we obtain with (23) $\mathbb{E}[f_i] = \mathbb{E}[\Delta Q]/N$ and therefore

$$\mathbb{E}[(y_i - \hat{y}_i)^2] = \frac{\mathbb{E}[\Delta Q]}{N q_i}. \quad (25)$$

Taking into account that at the steady state $\mathbb{E}[y_i] = \hat{y}_i$ does hold (recall the discussion above), (25) is also the variance of y_i

$$\text{Var}[y_i] = \frac{\mathbb{E}[\Delta Q]}{N q_i}. \quad (26)$$

After insertion of (25) into the evolution criterion (22) we end up with a surprisingly simple condition (recall that $\sum_i q_i = \text{Tr}[\mathbf{Q}]$)

$$\mathbb{E}[\Delta Q] \geq \frac{\sigma_\delta N}{4\mu c_{\mu/\mu, \lambda}}. \quad (27)$$

As can be checked by experiments (see Section 2.4), the equal sign in (27) predicts the average steady state ΔQ well as long as the mutation strength of the ES is controlled appropriately. The most astonishing message from (27) is the independence of $E[\Delta Q]$ on the $\mathbf{Q}$ matrix. This was already observed in $(1, \lambda)$ -ES runs in [8]. However, the reason for this interesting behavior remained obscure. Now we are able to explain this behavior by the equipartition effect which decomposes the (arbitrarily oriented) ellipsoid into its principal fitness components.

Inserting (27) into (26) yields an estimate for the object parameter fluctuations at the steady state. Since the steady state is characterized by the equal sign in (27), we obtain under steady state conditions

$$\text{Var}[y_i] = \frac{\sigma_\delta}{4\mu c_{\mu/\mu, \lambda} q_i}. \quad (28)$$

This result is also in accordance with experiments. The main conclusion that can be drawn from (28) is that y_i parameter fluctuations decrease with the increase of the corresponding eigenvalue q_i . This is reasonable: A large eigenvalue q_i (compared to a smaller q_j) results in a higher sensitivity of the fitness on the particular parameter space direction $\mathbf{e}_i$. That is, it produces (on average) a larger deviation from the optimal fitness value. Such deviations, however, are singled out by the (μ, λ) selection, only small deviations will survive. On the other hand, small eigenvalues reduce the influence of the y_i fluctuations on the fitness. Therefore, y fluctuations in such $\mathbf{e}$ -directions will be larger.

2.4 ES-Dynamics and Comparison with Experiments

The dynamical behavior of the ES maximizing the noisy function class (1), (2) has been investigated on three ellipsoidal test functions

$$Q_{g1}(\mathbf{y}) = - \sum_{i=1}^N y_i^2, \quad (\mathbf{Q})_{ij} = i\delta_{ij}, \quad (q_i = i), \quad (1.29a)$$

$$Q_{g2}(\mathbf{y}) = - \sum_{i=1}^N i^2 y_i^2, \quad (\mathbf{Q})_{ij} = i^2 \delta_{ij}, \quad (q_i = i^2), \quad (1.29b)$$

$$Q_{g3}(\mathbf{y}) = - \sum_{j=1}^N \left(\sum_{i=1}^j y_i \right)^2, \quad (\mathbf{Q})_{ij} = \min[N - i + 1, N - j + 1], \quad (1.29c)$$

for dimensionalities $N = 30$ and 100 . While $Q_{g1}(\mathbf{y})$ and $Q_{g2}(\mathbf{y})$ define axis-parallel ellipsoids, the third function, also known as ‘‘Schwefel’s’’ function, has a certain non-parallel orientation. Since we are using isotropic mutations (sphere-symmetrical mutations) the orientation of the ellipsoids does not affect the performance of the ES on these test functions. However, $Q_{g3}(\mathbf{y})$ has the peculiarity that the eigenvalue spectrum of $\mathbf{Q}$ possesses a dominating eigenvalue. Thus the shape of this ellipsoid resembles a distorted discus. This might influence the dynamical behavior, however, in the experiments performed using the noise model (2) no peculiarities concerning the steady-state behavior have been observed.

Differences are observed, however, concerning the dynamic behavior of the different mutation strength σ control rules used. We have tested the standard mutative σ self-adaptation (σ SA, see e.g. [4]) and the cumulative step-length adaptation (CSA) proposed by Gawelczyk, Hansen, and Ostermeier [14,15,16] without covariance matrix adaptation, i.e. using isotropic mutations. Figures 1a–d show the typical behaviors. Both ES versions end up with a steady state behavior where the fitness values are in expectation away from the global optimum $\Delta Q = 0$, i.e. $E[\Delta Q] > 0$. This is the typical behavior when noise is involved in the fitness evaluations. Clearly, the main aim is to have $E[\Delta Q]$ as small as possible. When comparing the resulting steady-state $E[\Delta Q]$ in Fig. 1c,d one notices that the CSA-ES yields a much larger $E[\Delta Q]$ than the σ SA-ES. From this point of view, the σ SA-ES should be preferred. However, as one can see this is brought at the expense of a slower approach to the steady state. Comparing the steady state behavior of the two ES types on the two test functions (1.29a) and (1.29b) one also sees that, using CSA-ES, the effect of larger $E[\Delta Q]$ gets larger with

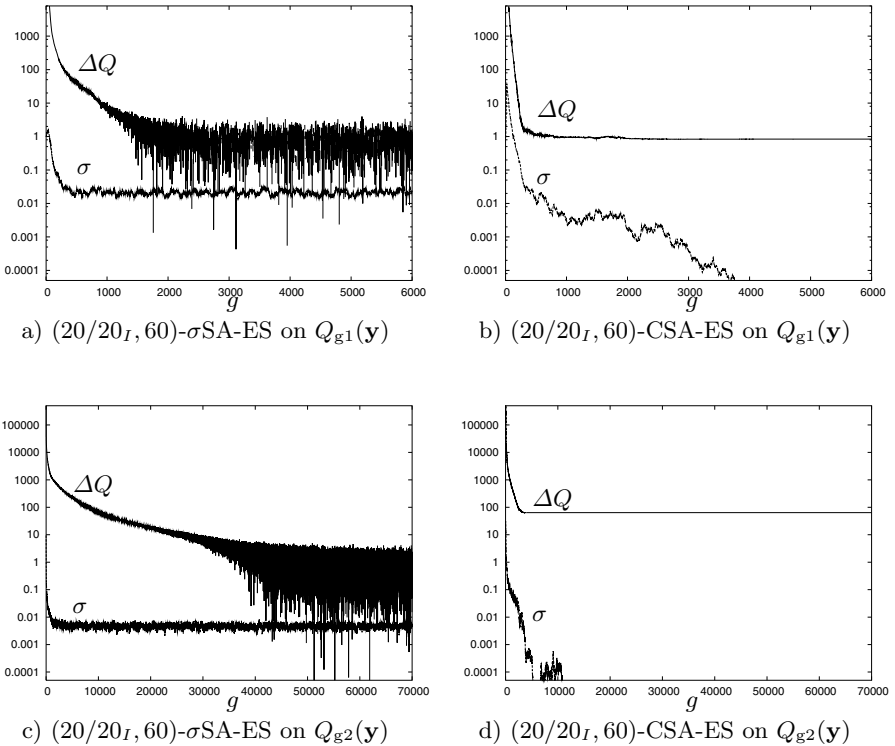


Fig. 1. Evolution dynamics by $(20/20_I, 60)$ -ES on (1.29a, b) ($N = 30$) using mutative self-adaptation (σ SA, Figs. a and c) and cumulative step length adaptation (CSA, Figs. b and d). The CSA exhibits premature convergence on test function $Q_{g2}(\mathbf{y})$ (Fig. d). As noise strength $\sigma_\delta = 1$ has been chosen.

increasing non-sphericity.¹ The reason for this undesirable behavior can be explained when considering the mutation strengths σ actually realized during the evolution. While the σ SA-ES produces a quasi-constant steady state mutation strength, the CSA-ES produces an almost random walk like σ behavior on the logarithmic scale with very small σ values. That is, the CSA σ control rule produces a nearly premature convergence behavior and the ES is not able to further evolve towards the optimizer state. The reason for this – at first glance – astonishing behavior can be traced back to the optimality condition the CSA control rule is based upon [13]: Consecutive changes of the parental centroids should be – on average – perpendicular to each other in order to have maximal progress on the sphere model. The analysis in [9] shows, however, that this assumption leads to a wrong adaptation behavior when fitness information is disturbed by noise. As a result, σ is decreased even though it should be kept nearly constant (for an in-depth discussion on the sphere model, see [9]).

There is a remedy for the undesired σ decrease in CSA-ES: Simply keep the mutation strength σ above a certain (but small) limit σ_0 . Figure 2 shows the effect of this remedy. The CSA-ES is prevented from premature convergence.

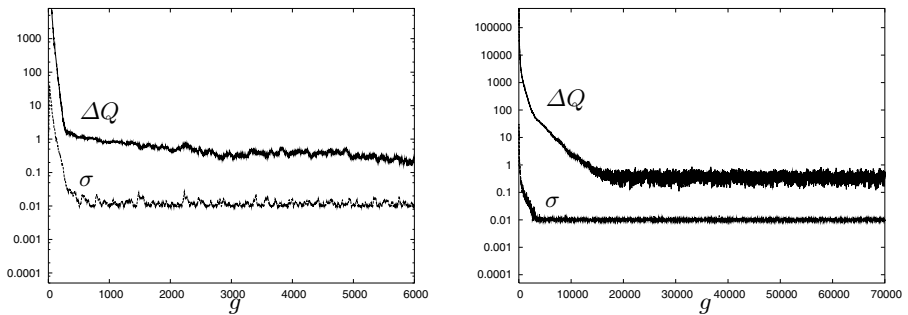


Fig. 2. Evolution dynamics of the $(20/20_I, 60)$ -CSA-ES keeping σ explicitly above $\sigma_0 = 0.01$. Left: test function (1.29a), $N = 30$; right: test function (1.29b), $N = 30$. As noise strength $\sigma_\delta = 1$ has been chosen.

The problem is, however, that fixing σ_0 is a difficult task and also that the approach to the steady state is slowed down. Therefore, this method cannot be recommended as a clever strategy. In the following we will not consider the CSA-ES further because in this article we are mainly interested in the expected steady state ΔQ . Therefore, our simulations will be performed using the old σ SA-ES.

Figure 3 compares the predictive quality of the equal sign in (27) as an estimate for the expected steady state ΔQ . The $(\mu/\mu_I, 60)$ - σ SA-ES has been used for the simulations. ΔQ was recorded at each generation after a number of transient generations g_0 by evaluating the (noisy) fitness (1), (2) of the parental centroid

¹ This might be an argument for using the covariance matrix adaptation (CMA) [15], however, this is not the focus of this paper.

using the test functions (1.29a,b,c). The number of generations used for averaging ΔQ is 200,000. The noise strength used is $\sigma_\delta = 1$. There is a good agreement between experiments and the lower bound of $E[\Delta Q]$ given by the curve obtained from (27). Recall that the lower bound corresponds to vanishing normalized mutation strength in the original evolution criterion (7). Considering the actually realized σ values (see, e.g., the figures on the left-hand sides of Figs. 1 and 2) one realizes that the σ SA-ES exhibits a behavior where σ is obviously that small at the steady state such that the equal sign in (27) is roughly fulfilled. That is

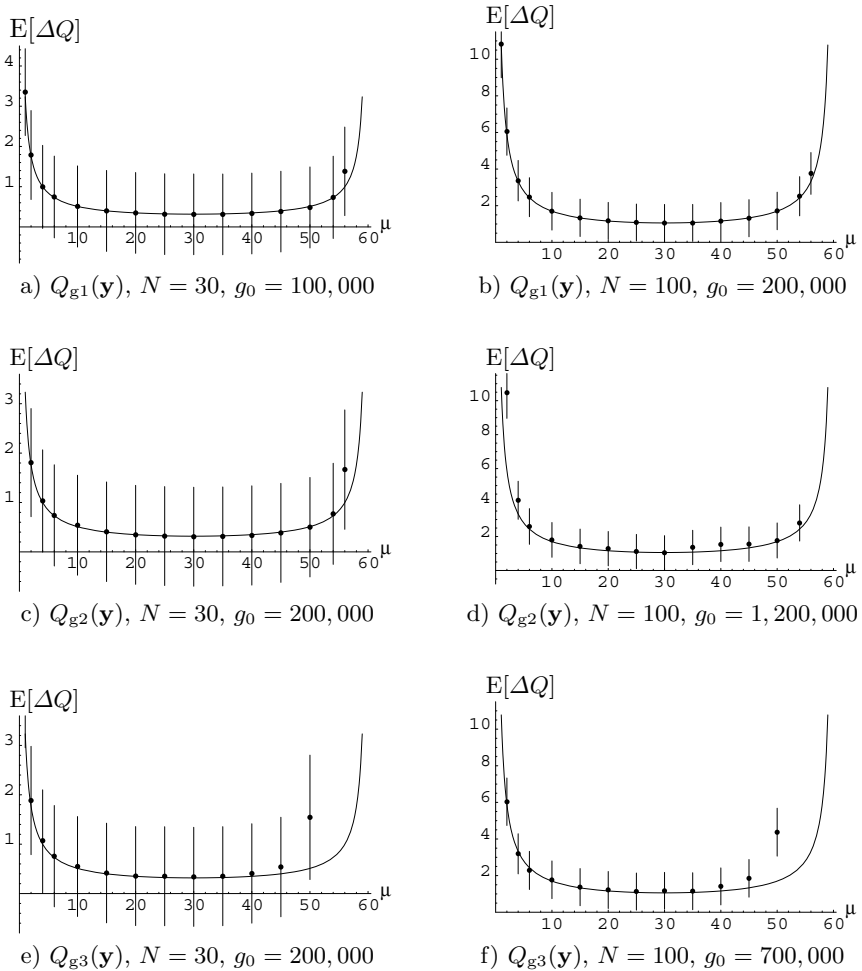


Fig. 3. Dependence of the expected steady state fitness error $E[\Delta Q]$ on the parent numbers $\mu = 1, 2, 4, 6, 10, 15, 20, 25, 30, 35, 40, 45, 50, 54, 56, 58, 59$ given fixed offspring number $\lambda = 60$. The vertical bars indicate the measured $\pm$ standard deviation of ΔQ . Note, some data points are missing, see explanation in the text.

why we observe such a good agreement between theory and experiments. On the other hand the mutation strength is large enough to ensure convergence to the vicinity of the steady state described by (27). This is in contrast to the CSA-ES where the mutation strength goes down very rapidly when reaching the vicinity of the steady state. Violating the smallness assumption of σ , however, will result in a similar behavior: The ES cannot approach states which are described by the equal sign in (27). This can be observed in σ SA-ES with μ/λ near 1, i.e. in strategies with low selective pressure, and can also be seen in the plots: Depending on the test function and the dimensionality there are some data points missing (usually $\mu = 59$ and $\mu = 58$, sometimes even for smaller μ) due to divergence. The behavior of the σ SA-ES is diametrically opposite to the CSA-ES under this condition. Having a very small selection pressure results in an almost random selection behavior. As has been shown in [10], random selection results in an exponential increase of the mutation strength of the σ SA-ES. Therefore, one observes a continuously increasing mutation strength if $\lambda - \mu$ is chosen too small. This effect starts gradually with increasing μ (keeping λ constant) and can be observed in the experiments presented.

3 Conclusions and Outlook

Using the equipartition assumption we were able to derive a simple formula which predicts the final expected fitness deviation surprisingly well. While the σ SA-ES reaches the predicted fitness deviation, the CSA-ES exhibits premature convergence on ellipsoidal test function with a high degree of non-sphericity.

Formula (27) can be used for population sizing. In order to get to the optimizer as closely as possible $\mu/\lambda = 0.5$ should be chosen. Getting to the steady state as fast as possible, however, requires $\mu/\lambda \approx 0.27$ (sphere model assumption and $N \rightarrow \infty$, not considered in this paper). Considering the plots in Fig. 3, $\mu/\lambda = 0.3$ seems to be a good compromise.

Since both CSA-ES and σ SA-ES use isotropic mutations, in a next step ES with nonisotropic mutations should be investigated. One might expect an improved ES behavior using covariance matrix adaptation (CMA) [15]. While the CMA-ES may yield better results than the CSA-ES, theoretically, CMA can *not* significantly improve the *steady state results* of the σ SA-ES (basically, CMA-ES transforms $\mathbf{Q}$ into another $\tilde{\mathbf{Q}}$, but (27) does *not* depend on $\mathbf{Q}$ or $\tilde{\mathbf{Q}}$ at all). However, we can expect an improved transient behavior (decreasing g_0) of the CMA-ES compared to the ES with isotropic mutations. This remains to be investigated in the future.

References

1. D. V. Arnold. *Noisy Optimization with Evolution Strategies*. Kluwer Academic Publishers, Dordrecht, 2002.
2. D. V. Arnold and H.-G. Beyer. Performance Analysis of Evolution Strategies with Multi-Recombination in High-Dimensional $\mathbb{R}^N$ -Search Spaces Disturbed by Noise. *Theoretical Computer Science*, 289:629–647, 2002.

3. D. V. Arnold and H.-G. Beyer. A Comparison of Evolution Strategies with Other Direct Search Methods in the Presence of Noise. *Computational Optimization and Applications*, 24:135–159, 2003.
4. T. Bäck, U. Hammel, and H.-P. Schwefel. Evolutionary computation: comments on the history and current state. *IEEE Transactions on Evolutionary Computation*, 1(1):3–17, 1997.
5. H.-G. Beyer. Toward a Theory of Evolution Strategies: Some Asymptotical Results from the $(1, +\lambda)$ -Theory. *Evolutionary Computation*, 1(2):165–188, 1993.
6. H.-G. Beyer. Evolutionary Algorithms in Noisy Environments: Theoretical Issues and Guidelines for Practice. *Computer Methods in Applied Mechanics and Engineering*, 186(2–4):239–267, 2000.
7. H.-G. Beyer. *The Theory of Evolution Strategies*. Natural Computing Series. Springer, Heidelberg, 2001.
8. H.-G. Beyer and D. V. Arnold. Fitness Noise and Localization Errors of the Optimum in General Quadratic Fitness Models. In W. Banzhaf, J. Daida, A.E. Eiben, M.H. Garzon, V. Honavar, M. Jakiela, and R.E. Smith, editors, *GECCO-99: Proceedings of the Genetic and Evolutionary Computation Conference*, pages 817–824, San Francisco, CA, 1999. Morgan Kaufmann.
9. H.-G. Beyer and D.V. Arnold. Qualms Regarding the Optimality of Cumulative Path Length Control in CSA/CMA-Evolution Strategies. *Evolutionary Computation*, 11(1):19–28, 2003.
10. H.-G. Beyer and K. Deb. On Self-Adaptive Features in Real-Parameter Evolutionary Algorithms. *IEEE Transactions on Evolutionary Computation*, 5(3):250–270, 2001.
11. H.-G. Beyer, M. Olhofer, and B. Sendhoff. On the Behavior of $(\mu/\mu_I, \lambda)$ -ES Optimizing Functions Disturbed by Generalized Noise. In K. De Jong, R. Poli, and J. Rowe, editors, *Foundations of Genetic Algorithms, 7*, San Francisco, CA, 2003. Morgan Kaufmann. in print.
12. J. Branke. *Evolutionary Optimization in Dynamic Environments*. Kluwer Academic Publishers, Dordrecht, 2001.
13. N. Hansen and A. Ostermeier. Adapting Arbitrary Normal Mutation Distributions in Evolution Strategies: The Covariance Matrix Adaptation. In *Proceedings of 1996 IEEE Int'l Conf. on Evolutionary Computation (ICEC '96)*, pages 312–317. IEEE Press, NY, 1996.
14. N. Hansen and A. Ostermeier. Convergence Properties of Evolution Strategies with the Derandomized Covariance Matrix Adaptation: The $(\mu/\mu_I, \lambda)$ -CMA-ES. In H.-J. Zimmermann, editor, *5th European Congress on Intelligent Techniques and Soft Computing (EUFIT'97)*, pages 650–654, Aachen, Germany, 1997. Verlag Mainz.
15. N. Hansen and A. Ostermeier. Completely Derandomized Self-Adaptation in Evolution Strategies. *Evolutionary Computation*, 9(2):159–195, 2001.
16. A. Ostermeier, A. Gawelczyk, and N. Hansen. A Derandomized Approach to Self-Adaptation of Evolution Strategies. *Evolutionary Computation*, 2(4):369–380, 1995.
17. S. Tsutsui and A. Ghosh. Genetic Algorithms with a Robust Solution Searching Scheme. *IEEE Transactions on Evolutionary Computation*, 1(3):201–208, 1997.
18. D. Wiesmann, U. Hammel, and T. Bäck. Robust Design of Multilayer Optical Coatings by Means of Evolutionary Algorithms. *IEEE Transactions on Evolutionary Computation*, 2(4):162–167, 1998.

Theoretical Analysis of Simple Evolution Strategies in Quickly Changing Environments

Jürgen Branke¹ and Wei Wang²

¹ Institute AIFB, University of Karlsruhe, 76128 Karlsruhe, Germany
branke@aifb.uni-karlsruhe.de

² Department of Education Technologies
Nanjing University of Posts and Telecommunications
P.O.Box 73, 38 GuangDong Road, 210003 Nanjing, China
wwang@jlonline.com

Abstract. Evolutionary algorithms applied to dynamic optimization problems has become a promising research area. So far, all papers in the area have assumed that the environment changes only *between* generations. In this paper, we take a first look at possibilities to handle a change during a generation. For that purpose, we derive an analytical model for a (1, 2) evolution strategy and show that sometimes it is better to ignore the environmental change until the end of the generation, than to evaluate each individual with the most up-to-date fitness function.

1 Introduction

Many optimization problems are dynamic and change over time. A suitable optimization algorithm has to reflect these changes by repeatedly adapting the solution to the changed environment. Evolutionary algorithms (EAs) are inspired by natural evolution, which can be regarded as adaptation in an inherently dynamic and stochastic environment. Given this background, EAs seem to be naturally suited to be applied to dynamic optimization problems, and have already shown great promise (for an overview on the area, see e.g. [2]).

EAs are iterative algorithms. In each “generation”, a number of new solutions (individuals) are generated, evaluated, and inserted into the population. So far, at least to the authors’ knowledge, all publications on EAs for dynamic optimization problems assume that the environment (fitness function) changes *between* generations. Although this assumption is convenient, we consider it an oversimplification, because generally the environment is independent of the EA, and thus can change at any time, i.e. also *within* a generation.

In this paper, we specifically address the issue of how to handle an environmental change during a generation. For that purpose, we develop an analytical model for a (1, 2) evolution strategy applied to the dynamic bit matching problem. The dynamic bit-matching problem is the dynamic variant of the well known onemax problem. The goal is to reproduce a binary target template, with the template changing over time. The fitness is just the number of bits identical with the template. We will analytically compare two ways to deal with a change

of the target within a generation (i.e. after the first child has been evaluated): One possibility is to use the new fitness function for the second individual, while the other possibility would be to ignore the change and use the same old fitness function also for the second individual.

The first approach uses up-to-date fitness information, but potentially suffers from the fact that the selection is based on fitness from two different fitness functions. The second approach deliberately ignores new information about the environment, but on the other hand, selection chooses between individuals evaluated with the same fitness function. Note that the second approach assumes that the old fitness function can still be used after the environment has changed. This seems justified, since in most practical applications, the fitness is evaluated by *simulating* the environment on a computer. Continuing to use the old fitness function then simply means to delay the update of the computer model.

The paper is structured as follows: in the next section, we will briefly mention a number of papers related to our work. In Section 3 we derive the theoretical model for the $(1, 2)$ reproduction scheme and compare its performance to the $(1+1)$ reproduction scheme, assuming the environment changes *between* generations. Then, we turn to the issue of changes *within* a generation and compare the ideas of always using the up-to-date fitness function or using an old fitness function for the whole generation for the $(1, 2)$ reproduction scheme. The paper concludes with a summary and an outlook for future work.

2 Related Work

Our paper is largely based on the work by Stanhope and Daida published in [5,6] but extends it significantly. Stanhope and Daida derive transition probabilities of the individual's fitness for a $(1+1)$ EA on the dynamic bit-matching problem. Let us briefly summarize their results, which will serve as a baseline for our extensions.

We will be using the following notation:

- L : number of bits in the bit string. In our results reported below, we generally assume $L = 100$.
- r : mutation rate in terms of the number of bits changed by mutation (fixed number, not probability per bit)
- d : number of bits the target changes
- a : parent individual
- $f_t(b)$: fitness of individual b against the t -th target. The fitness after a change, i.e. against the next target, is usually denoted as $f_{t+1}(b)$. For reasons of readability, the subscript t is often omitted
- $m_r(b)$: result of mutating individual b by r bits

The probability that an offspring has fitness x , given that it was generated by mutating an individual a with fitness $f(a)$, is the probability that the number of wrong bits mutated minus the number of correct bits mutated is equal to $x - f(a)$, which can be calculated as

$$P(f(m_r(a)) = x) = \frac{\binom{L-f(a)}{(x-f(a)+r)/2} \binom{f(a)}{r-(x-f(a)+r)/2}}{\binom{L}{r}} \quad (1)$$

In a $(1+1)$ EA, the better of the two individuals (parent and offspring) is kept, and the probability that the surviving individual has fitness x can thus be calculated as

$$P(\max_{b \in \{a, m_r(a)\}} f(b) = x) = \begin{cases} 0 & : x < f(a) \\ \sum_{i=0}^x P(f(m_r(a)) = i) & : x = f(a) \\ P(f(m_r(a)) = x) & : x > f(a) \end{cases} \quad (2)$$

To account for changes of the environment, first note that a change of the target of d bits is equivalent (in terms of the individual's fitness distribution) to a d bit mutation of the individual. Thus we can calculate the distribution function of the fitness of a selected individual after an environmental change as:

$$P(f_{t+1}(\arg \max_{b \in \{a, m_r(a)\}} f(b)) = x) = \sum_{i=0}^L P(\max_{b \in \{a, m_r(a)\}} f(b) = i) P(f(m_d(c_i)) = x) \quad (3)$$

with f_{t+1} denoting the fitness against the new target (i.e. a target with d bits changed), and c_i denoting any individual with fitness i .

Further related work includes the paper by Droste [4]. There, the expected time to encounter the optimal solution for the first time is derived for a $(1+1)$ EA on the dynamic bit matching problem. The model is different in that it assumes a mutation probability for each bit instead of a fixed number of bits inverted as we do. A brief survey on EA approaches to dynamic optimization problems has been presented e.g. in [1], a thorough treatment of different aspects of this subject can be found in [2].

3 Comparing $(1, 2)$ and $(1+1)$ on an Environment Changing *between* Generations

Let us now derive similar equations for the $(1, 2)$ reproduction scheme. Note that although now two new individuals are generated in every iteration, the total number of evaluations per generation is equal to the $(1+1)$ reproduction scheme when a change occurs after every generation, because the implicit assumption above was that the old individual is re-evaluated in every iteration in order to allow for correct selections.

Ignoring a change of the fitness function for now, the fitness distribution of the better of the two children can be described as:

$$P\left(\max_{b \in \{m_{r_1}(a), m_{r_2}(a)\}} f(b) = x\right) = P^2(f(m_r(a)) \leq x) - P^2(f(m_r(a)) < x) \quad (4)$$

which can be calculated using Equation 1.

The following equation takes a change after each generation into account:

$$\begin{aligned}
 & P \left(f_{t+1} \left(\arg \max_{b \in \{m_{r_1}(a), m_{r_2}(a)\}} f(b) \right) = x \right) \\
 &= \sum_{i=0}^L P \left(\max_{b \in \{m_{r_1}(a), m_{r_2}(a)\}} f(b) = i \right) P(f(m_d(c_i)) = x)
 \end{aligned} \tag{5}$$

3.1 Optimal Mutation Rate

Figure 1 compares the optimal mutation rate (yielding the highest expected fitness of the selected individual) for $(1+1)$ and $(1,2)$ depending on the fitness of the current parent individual and assuming a string length of 100. The results are identical for a change severity of $d = 0, 1, 2, 3$. Obviously, for both approaches, with decreasing current fitness the optimal mutation rate increases quickly, as there is a higher chance for improvement and a lower chance for destroying valuable bits.

Since $(1+1)$ will never accept an individual worse than the current parent individual, it is safe to allow some mutation even when the parent has a high fitness. This is not the case for $(1,2)$, which risks a significant loss in fitness when mutation is introduced. For the examined case of a string length of 100, it is optimal to have a mutation rate of 0 as long as the parent's fitness is greater or equal to 73, and have a lower mutation rate than $(1+1)$ over the whole range.

It is also interesting to note that at least for $(1+1)$, mutating an even number of bits is never optimal. The reason is probably that an improvement can only occur when the child's fitness is actually *greater* than the parent's fitness. With an even number of bit-flips there is a relatively high probability that the effects of the different bit flips cancel out and the fitness of the individual is not changed at all (i.e. there can be no improvement). Using an odd number of bit flips "forces" mutation to change the individual's fitness. This leads to a larger number of actually better individuals, while the also larger number of worse individuals doesn't matter (since only better individuals are accepted).

3.2 Convergence Plots

The expected fitness distribution of next generation's parent individual corresponds to a transition matrix of a Markov chain. Assuming e.g. an initial random individual with fitness 50 and a total string length of 100, we can then compute the expected fitness of the parent individual over the generations. For the comparisons in this section, we used the optimal mutation rate for both approaches. Figure 2 shows the derived convergence plots for $d = 0, 1, 3$.

As would be expected, in a stationary environment, $(1+1)$ clearly outperforms $(1,2)$. The onemax fitness function has no local optima and thus a local hill-climber such as $(1+1)$ works great. Our analysis here is restricted to the simple dynamic bit matching benchmark, but it would be interesting to compare

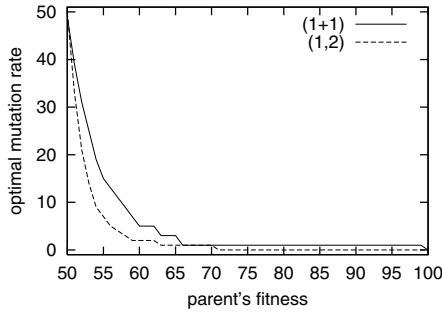


Fig. 1. Optimal mutation rate for $(1, 2)$ and $(1 + 1)$ reproduction depending on the fitness of the current parent individual. Total string length is assumed to be 100.

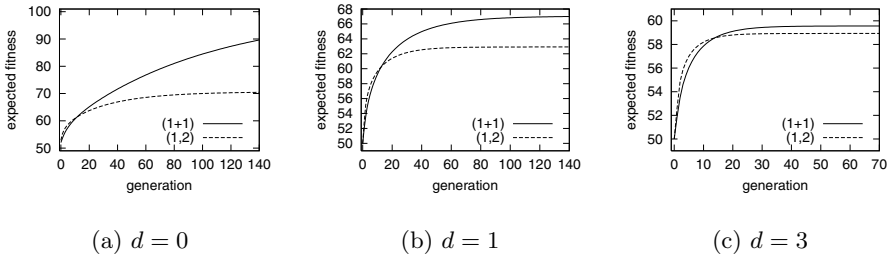


Fig. 2. Comparison of the convergence curves for $(1 + 1)$ and $(1, 2)$ for different change severities d . Total string length is assumed to be 100.

the two reproduction schemes also on a more rugged fitness landscape, where $(1 + 1)$ will get stuck in a local optimum.

As the environment starts to change, it is interesting to see that with increasing dynamism, the exploratory $(1, 2)$ reproduction scheme comes closer and closer to the exploitative $(1 + 1)$ reproduction scheme. In any case, $(1, 2)$ outperforms $(1 + 1)$ while the parental fitness is still low, therefore it would be beneficial to use $(1, 2)$ in the beginning of a run and then switch to $(1 + 1)$ later on.

4 Change *within* a Generation

In this section, we will consider the case where the fitness function changes during a generation of a $(1, 2)$ reproduction scheme, i.e. after the first child has been evaluated. Given this simple framework, we will analytically compare the two strategies already mentioned in the introduction, namely to evaluate the two individuals with the respective (different) current fitness functions, or to artificially delay the change and use the old fitness function also for the second child.

As has been explained in the introduction, each of the above approaches has its advantages and its drawbacks, and we would like to know which approach is preferable given specific circumstances. Let us first consider the following illustrative example: Let x_1 and x_2 be the two children generated, and let $f(x_1) = 10$ and $f(x_2) = 12$ be the respective fitnesses *before* the environmental change, $f_{t+1}(x_1) = 11$ and $f_{t+1}(x_2) = 9$ be the respective fitnesses *after* the environmental change. The approach using the up-to-date fitness will select between $f(x_1) = 10$ and $f_{t+1}(x_2) = 9$, i.e. correctly select x_1 assuming a maximization problem. The approach delaying the change will compare $f(x_1) = 10$ and $f(x_2) = 12$ and select x_2 , which is actually worse than x_1 . On the other hand, in a situation where $f_{t+1}(x_1) = 8$ and $f_{t+1}(x_2) = 9$, the approach using up-to-date fitness is mistaken, while the approach which delays the change correctly selects x_2 .

More generally, delaying the change will make the correct decision as long as the change does not affect the relative order of the two children. For the dynamic bit matching problem this is probably the case for very large bit strings and small mutation rates, because then x_1 and x_2 are identical for the vast majority of bits, and a change of the template will most likely affect them in the same way. In particular, if the environmental change is severe compared to mutation, the (undesirable) effect of comparing individuals with different fitness functions will “override” the fitness difference due to mutation, and mistakes are likely. In the following, we will compare the two strategies analytically.

For the strategy which delays the change, the fitness distribution will be identical to the case considered in Section 3 with a change at the end of a generation. The situation for the other approach is depicted in Figure 3: First, a child is generated and evaluated, then the environment changes, a second child is generated and evaluated, and the child with the higher fitness is selected.

For analytical comparison, we need the *actual* fitness of the selected child, i.e. its fitness in the new environment. If the second child is selected, that is no problem, since it has already been evaluated against the new environment. However, if the first child is selected, the assigned fitness is outdated, and it has to be re-evaluated in the new environment (note that this re-evaluation is only necessary for the theoretical investigation and is not part of the implemented EA). The two cases are compared in Figure 3.

The difficulty is that the change before re-evaluating the first child has to be identical to the one which occurred before the evaluation of the second child. We have to be able to replicate that change, while at the same time we would like to continue using the high-level approach from the previous sections, avoiding to enumerate all possible changes on a bit-level. To solve this difficulty, let us first introduce the concept of two-step mutation.

4.1 Two-Step Mutation

Basically, a mutation of r bits can be regarded as first mutating $s < r$ bits, and then mutating the remaining $r - s$ bits on a shorter string, not containing the s bits mutated first (to avoid flipping the same bits twice).

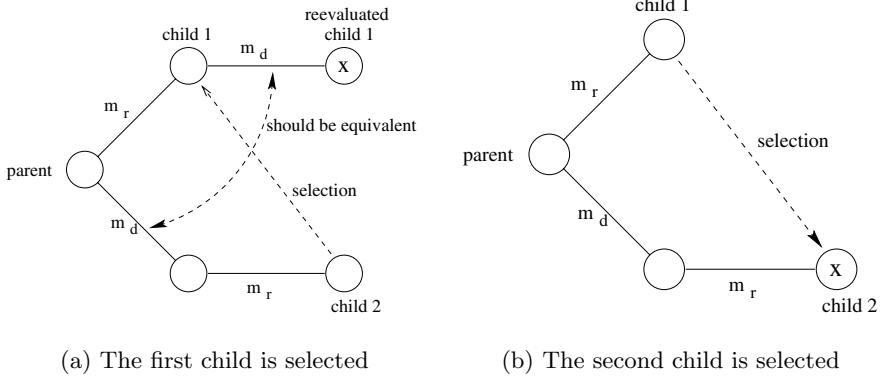


Fig. 3. Illustration of the process when the fitness function changes after the first child is evaluated, depending on whether (a) the first child is selected or (b) the second child is selected. We would like to derive the probability distribution for the fitness of the individual marked “x”. Note that there is only one change of the environment, thus the two changes in (a) need to be identical.

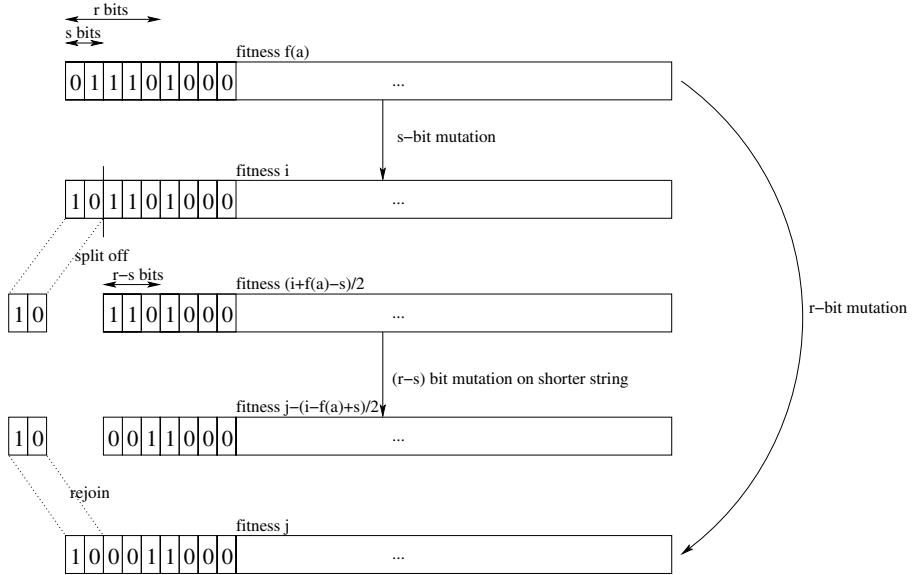


Fig. 4. Illustration of the concept of two-step mutation. Note that the order of the bits is irrelevant, thus w.l.o.g. we assume in this figure that the first bits are mutated.

The concept is illustrated in Figure 4: First, we do an ordinary s -bit mutation on the whole string. Let us assume that the resulting individual has fitness i . Then, the $r - s$ -bit mutation can be captured by the basic mutation operation

described by Equation 1, with the following parameters: The length of the substring is $L - s$, the mutation rate is $r - s$, the initial fitness of the substring is $(i + f(a) - s)/2$, and the fitness after mutation should be $j - (i - f(a) + s)/2$ (assuming the fitness of the individual after the whole two-step mutation is equal to j). We will denote such a mutation on a string of fitness $(i + f(a) - s)/2$ and length $L - s$ as $m_{r-s}(c_{(i+f(a)-s)/2}^{L-s})$. The probability distribution of the complete two-step mutation can then be expressed as

$$P(f(m_r(a)) = j) = \sum_{i=\max\{0, f(a)-s\}}^{\min\{L, f(a)+s\}} P(f(m_s(a)) = i) \cdot P\left(f(m_{r-s}(c_{(i+f(a)-s)/2}^{L-s})) = j - \frac{i - f(a) + s}{2}\right) \quad (6)$$

4.2 Using Two-Step Mutation to Model Change within a Generation

The above proposed two-step mutation can now be used to recover the change that has happened before the second child was evaluated, and apply it to the first child as well. The basic idea is to split the mutation into two steps: first, the s bits are mutated that are common to the r -bit mutation and the d -bit change of the target. Then, the remaining $r - s$ bits for mutation and the remaining $d - s$ bits to account for the environmental change are applied (see Figure 5 for illustration). Then, if we would like to apply the environmental change to the first child, we can do so by reversing the effect of the first s -bit mutation, and then applying a $d - s$ bit mutation on a string of length $L - r$ (since this last mutation has no common bits with the r -bit mutation).

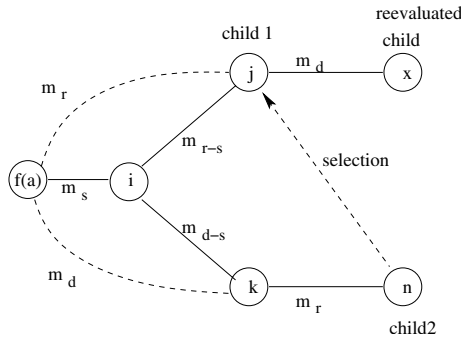


Fig. 5. Illustration of the use of two-step mutation for a (1,2) EA with a change of the environment after the first child has been evaluated, and assuming that the first child is selected. The letters inside the circles denote the fitnesses of the corresponding individuals.

The s -bit mutation and the $r - s$ bit mutation have already been discussed above. For the $d - s$ bit mutation, the string length is $L - r$ (since it has no common bits with either the s -bit mutation nor the $r - s$ bit mutation), the initial fitness is $(j + f(a) - r)/2$, and the fitness after mutation should be $k - i + (j + f(a) - r)/2$ (assuming the fitness of the individual after the whole two-step mutation is equal to k). Overall,

$$\begin{aligned} & P((f(m_d(a)) = k) | (f(m_r(a)) = j) \wedge (f(m_s(a)) = i)) \\ &= P\left(f(m_{d-s}(c_{(j+f(a)-r)/2}^{L-r})) = k - i + \frac{j + f(a) - r}{2}\right) \end{aligned} \quad (7)$$

Let us first consider the case when the first child has an observed fitness equal or better than the second child, and let us assume that in this case, the first child is selected (selecting either child with equal probability in the case of equal fitness could also be handled, but is omitted here for clarity). As usual, we would like to calculate the probability that the child (in this case the first one) has fitness x .

The probability that the first child is selected can be calculated as

$$\begin{aligned} & \sum_{j=\max\{0, f(a)-r\}}^{\min\{L, f(a)+r\}} P(f(m_r(a)) = j) P(f(m_r(m_d(a))) \leq j) \\ &= \sum_{j=\max\{0, f(a)-r\}}^{\min\{L, f(a)+r\}} P(f(m_r(a)) = j) \\ & \quad \cdot \sum_{k=\max\{0, f(a)-d\}}^{\min\{L, f(a)+d\}} P(f(m_d(a)) = k) P(f(m_r(c_k)) \leq j) \end{aligned} \quad (8)$$

Using two-step mutation, this can be re-formulated as

$$\begin{aligned} & \sum_{s=0}^{\min\{r, d\}} P(v(r, d) = s) \sum_{i=\max\{0, f(a)-s\}}^{\min\{L, f(a)+s\}} P(f(m_s(a)) = i) \\ & \cdot \sum_{j=\max\{0, i-r+s\}}^{\min\{L, i+r-s\}} P\left(f(m_{r-s}(c_{(i+f(a)-s)/2}^{L-s})) = j - \frac{i - f(a) + s}{2}\right) \\ & \cdot \sum_{k=\max\{0, i-d+s\}}^{\min\{L, i+d-s\}} P\left(f(m_{d-s}(c_{(j+f(a)-r)/2}^{L-r})) = k - i + \frac{j + f(a) - r}{2}\right) \\ & \cdot P(f(m_r(c_k)) \leq j) \end{aligned} \quad (9)$$

where $P(v(r, d) = s)$ denotes the probability that a d -bit mutation and a r -bit mutation have exactly s common bits and can be calculated as

$$P(v(r, d) = s) = \frac{\binom{L}{s} \cdot \binom{L-s}{r-s} \cdot \binom{L-r}{d-s}}{\binom{L}{r} \cdot \binom{L}{d}} \quad (10)$$

The fitness of the first child after re-evaluation is determined by $x = f(a) + j + k - 2i$. Since x is assumed to be known, k can be replaced by $k = x - f(a) + 2i - j$.

The probability that the first child has been selected and has fitness x after re-evaluation can thus be calculated as

$$\begin{aligned}
 & \sum_{s=0}^{\min\{r,d\}} P(v(r,d) = s) \sum_{i=\max\{0,f(a)-s\}}^{\min\{L,f(a)+s\}} P(f(m_s(a)) = i) \\
 & \cdot \sum_{j=\max\{0,i-r+s\}}^{\min\{L,i+r-s\}} P\left(f(m_{r-s}(c_{(i+f(a)-s)/2}^{L-s})) = j - \frac{i - f(a) + s}{2}\right) \\
 & \cdot P\left(f(m_{d-s}(c_{(j+f(a)-r)/2}^{L-r})) = x + i - \frac{j + f(a) + r}{2}\right) \\
 & \cdot P(f(m_r(c_{x-f(a)+2i-j})) \leq j) \tag{11}
 \end{aligned}$$

The second case, namely that the second child has a better observed fitness and is selected is much easier to handle. Since we don't have to re-evaluate the second individual, we just need to calculate the probability that it has fitness x while the first individual has fitness smaller than x . In terms of equations, this can be expressed as

$$\sum_{k=\max\{0,f(a)-d\}}^{\min\{L,f(a)+d\}} P(f(m_d(a) = k)P(f(m_r(c_k)) = x)P(f(m_r(a)) < x) \tag{12}$$

The total probability of the new parent having fitness x is then just the probability that the first child is selected and has actual fitness x (Equation 11) plus the probability that the second child is selected and has fitness x (Equation 12). As has been shown in [3], the presented framework can also be adapted to the general case of $(1, \lambda)$ evolution strategies with $\lambda > 2$.

4.3 Comparisons

The above equations allow us to compare the two strategies, namely to use the old fitness function for both individuals or to always use the latest fitness function, in terms of the expected fitness of the next generation's parent. Again, we assume a bit string of length 100 for the results reported below.

Figure 6 compares the difference in expected fitness of the next generation's parent individual depending on the current parent individual's fitness and the change severity d of the environment. As can be seen, for $d = 1$ it is always somewhat better to use the up-to-date information when evaluating individuals. This can also be derived differently: The fitness difference of the two children before the change can never be equal to 1, since in both mutations, an equal number of bits are flipped. If the fitness difference is 0, it is important to know the effect of the environment to correctly select the better individual. If the fitness difference ≥ 2 , a change of a single bit can not reverse the ordering of the children and thus both fitness evaluation schemes will make the same decision.

However, for $d > 1$ such a simple analysis is not possible. Clearly according to Figure 6, with $d = 2$ or $d = 3$, using the old environment for the second child yields better results unless the parent's fitness is very low. As long as the parent's fitness is very low, the optimal mutation rate is very high, and the two children are likely to differ significantly in their true fitness. But with growing parental fitness and smaller optimal mutation rate, the children's true fitnesses may be quite similar, and when a severe change ($d > 1$) is only taken into account for one individual, there is the danger that it will “hide” the true fitness difference, misleading selection. The “steps” in the lines in Figure 6 correspond to the points where the optimal mutation probability changes. The effect of different fitness evaluation schemes on the convergence curves can be seen in Figure 7.

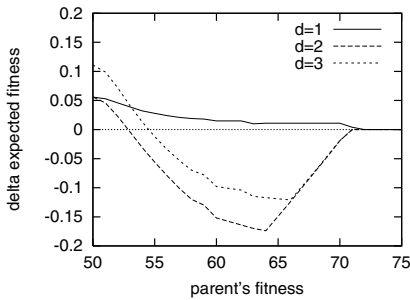


Fig. 6. Difference of expected fitness for next generation's parent individual depending on whether new or old fitness function is used for second child. If value is greater 0, it is better to use new fitness function and vice versa.

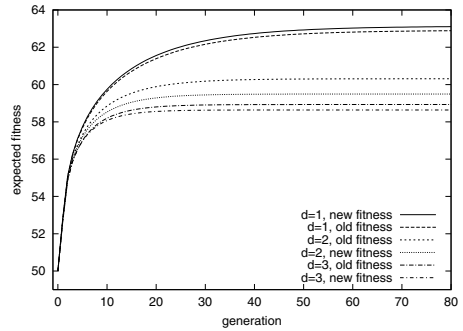


Fig. 7. Convergence curves of (1,2) reproduction strategy with either new or old fitness function used for the second child, for different change severities d . Lines are labeled in the order they appear in the plot.

5 Conclusion and Future Work

In this paper, we have approached the problem of quickly changing environments, and in particular the issue of how to handle a change of the fitness function that occurs *within* a generation of an evolutionary algorithm. Besides a general description of the issues and some ideas of how to handle such changes, we have derived analytic expressions which allow to calculate the probability distribution for fitnesses of the next generation's parent individual for the (1,2) EA on the dynamic bit matching problem. Using this framework, we compared two strategies to handle changes within a generation, namely to always use the up-to-date fitness information, or to keep the old fitness function despite the change of the environment. As has been shown, depending on the current fitness and the severity of the environmental change, one or the other strategy may be beneficial.

We are currently examining the issue of changes within a generation also from an empirical point of view, allowing us to look at much more complex problems and EA variants as suggested in the introduction.

Acknowledgments. We would like to thank Christopher Ronnewinkel for helpful discussions in the early phases of this research.

References

1. J. Branke. Evolutionary approaches to dynamic optimization problems - updated survey. In *GECCO Workshop on Evolutionary Algorithms for Dynamic Optimization Problems*, pages 27–30, 2001.
2. J. Branke. *Evolutionary Optimization in Dynamic Environments*. Kluwer, 2001.
3. J. Branke and W. Wang. Theoretical analysis of simple evolution strategies in quickly changing environments. Technical Report 423, Institut AIFB, University of Karlsruhe, 76128 Karlsruhe, Germany, 2002.
4. S. Droste. Analysis of the $(1 + 1)$ EA for a dynamically changing onemax-variant. In *Congress on Evolutionary Computation*, pages 55–60, 2002.
5. S. A. Stanhope and J. M. Daida. Optimal mutation and crossover rates for a genetic algorithm operating in a dynamic environment. In *Evolutionary Programming VII*, volume 1447 of *LNC3*, pages 693–702. Springer, 1998.
6. S. A. Stanhope and J. M. Daida. Genetic algorithm fitness dynamics in a changing environment. In *Congress on Evolutionary Computation*, volume 3, pages 1851–1858. IEEE, 1999.

Evolutionary Computing as a Tool for Grammar Development

Guy De Pauw

CNTS – Language Technology Group
UIA – University of Antwerp
Antwerp – Belgium
guy.depauw@ua.ac.be

Abstract. In this paper, an agent-based evolutionary computing technique is introduced, that is geared towards the automatic induction and optimization of grammars for natural language (GRAEL). We outline three instantiations of the GRAEL-environment: the GRAEL-1 system uses large annotated corpora to bootstrap grammatical structure in a society of autonomous agents, that tries to optimally redistribute grammatical information to reflect accurate probabilistic values for the task of parsing. In GRAEL-2, agents are allowed to mutate grammatical information, effectively implementing grammar rule discovery in a practical context. Finally, by employing a separate grammar induction module at the onset of the society, GRAEL-3 can be used as an unsupervised grammar induction technique.

1 Introduction

An important trend in the field of Machine Learning sees researchers employing combinatory methods to improve the classification accuracies of their algorithms. Natural language problems in particular benefit from combining classifiers to deal with the large datasets and expansive arrays of features that are paramount in describing this difficult and disparate domain that typically features a considerable amount of sub-regularities and exceptions [1]. Not only system combination and cascaded classifiers are well-established methods in the field of Machine Learning for natural language [2,3], also the techniques of bagging and boosting [4] have been used successfully on a number of natural language classification tasks [5,6]. These techniques hold in common that in no way do they alter the actual content of the information source of the predictor. Simply by re-distributing the data, different resamplings of the same classifier are generated to create a combination of classifiers.

The field of evolutionary computing has been applying problem-solving techniques that are similar in intent to the aforementioned Machine Learning recombination methods. Most evolutionary computing approaches hold in common that they try and find a solution to a particular problem, by recombining and mutating individuals in a society of possible solutions. This provides an attractive technique for problems involving large, complicated and non-linearly divisible search spaces. The evolutionary computing paradigm has however always seemed reluctant to deal with issues of natural language syntax. The fact that syntax is in essence a recursive, non-propositional system, dealing with complex issues such as long-distance dependencies and constraints, has made it

difficult to incorporate it in typically propositional evolutionary systems such as genetic algorithms. Most GA syntactic research so far has focused on non-linguistic data, with some notable exceptions [7,8,9,10]. Yet none of these systems are suited to a generic grammar optimization task, mainly because the grammatical formalism and evolutionary processes underlying these systems are designed to fit a particular task, such as information retrieval [11]. Yet so far, little or no progress has been achieved in evaluating evolutionary computing as a tool for the induction or optimization of data-driven parsing techniques.

The GRAEL (**GRAM**mar **Evo**Lution) framework [12] attempts to combine the sensibilities of the recombination machine learning methods and the attractive evolutionary properties of the concepts of genetic programming. It provides a suitable framework for the induction and optimization of any type of grammar for natural language in an evolutionary setting. In this paper we want to provide a general overview of GRAEL as a natural language grammar development technique. We will first identify the basic problem in Section 2, after which we outline the general architecture of the GRAEL environment in Section 3. Next, we will introduce three different instantiations of the GRAEL environment: in GRAEL-1 (Section 4) large annotated corpora are used to bootstrap grammatical structure in a society of agents, who engage in a series of communicative attempts, during which they redistribute grammatical information to reflect optimized probabilistic values for the task of parsing. In GRAEL-2 (Section 5), agents are allowed to mutate grammatical information, effectively implementing grammar rule discovery in a practical context. Finally, we look at GRAEL-3 in Section 6, which provides a method for unsupervised grammar induction.

2 Natural Language Grammar Development

Syntactic processing has always been deemed to be paramount to a wide range of applications, such as machine translation, information retrieval, speech recognition and the like. It is therefore not surprising that natural language syntax has always been one of the most active research areas in the field of language technology. All of the typical pitfalls in language like ambiguity, recursion and long-distance dependencies, are prominent problems in describing syntax in a computational context. Historically, most computational systems for syntactic *parsing*, employ hand-written grammars, consisting of a laboriously crafted set of grammar rules to apply syntactic structure to a sentence¹. But in recent years, a lot of research efforts are trying to automatically induce workable grammars from annotated corpora, i.e. large collections of pre-parsed sentences [13]. Since the tree-structures in these annotated corpus already implicitly contain a grammar, it is a relatively trivial task to induce a large-scale grammar and parser that is able to acquire reasonably high parsing accuracies on a held-out set of data [14,15,16].

Yet, data-analysis of the output generated by these parsers still brings to light fundamental limitations to these corpus-based methods. Even though they generally provide a much broader coverage as well as higher accuracy than hand-built grammars, corpus-induced grammars will still not hold enough grammatical information to provide

¹ Syntactic structure is typically presented as a parse tree, such as the ones in Figure 1.

structures for a large number of sentences in language, as some rules that are needed to generate the correct tree-structures are not induced from the original corpus. But even if there were such a thing as a full-coverage corpus-induced grammar, performance would still be limited by the probabilistic weights attributed to its rules. The GRAEL system described in this paper tries to alleviate the problems inherent to corpus-induced grammars, by establishing a distributed evolutionary computing method for grammar induction and optimization. Generally, GRAEL can be considered as a system that allows for the simultaneous development of a range of alternative solutions to a grammatical problem, optimized in a series of practical interactions in a society of agents controlled by evolutionary parameters.

3 Grammar Evolution

A typical GRAEL society consists of a population of agents in a virtual environment, each of which holds a number of structures that allow them to generate sentences as well as analyze other agents' sentences. These grammars are updated through an extended series of inter-agent interactions, using a form of error-driven learning. The evolutionary parameters are able to define the content and quality of the grammars that are being developed over time, by imposing fitness functions on the society. By embedding the grammars in autonomous agents, GRAEL ensures that the grammar development is grounded in the practical task of parsing itself.

The grammatical knowledge of the agents is typically bootstrapped by using an annotated natural language corpus [13]. At the onset of such a corpus-based GRAEL society, the syntactic structures of the corpus are randomly distributed over the agents, so that each agent holds a number of tree-structures in memory. The actual communication between agents is implemented in *language games* [17]: an agent (**ag1**) presents a sentence to another agent (**ag2**). If **ag2** is able to correctly analyze **ag1**'s sentence, the communication is successful. If on the other hand, **ag2** is lacking the proper grammatical information to parse the sentence correctly, **ag1** shares the necessary information for **ag2** to arrive at the proper solution.

A Toy Example

We take a look at an example of a very basic language game: Figure 1 shows a typical interaction between two agents. In this example, an annotated corpus of two sentences has been distributed over two agents. The two agents engage in a language game, in which **ag1** provides an assignment to **ag2**: **ag1** presents the sentence "I offered some bear hugs" to **ag2** for parsing. **ag2**'s knowledge does not contain the proper grammatical information to interpret this sentence the way **ag1** intended and so **ag2** will return an incorrect parse, albeit consistent with its own grammar.

ag1 will consequently try and help **ag2** out by revealing the minimal correct sub-structure of the correct parse that should enable **ag2** to arrive at the correct solution. **ag2** will incorporate this information in its grammar and try to parse the sentence again with the updated knowledge. Once **ag2** is able to provide the correct analysis (or is not able to after a certain number of attempts) either **ag1**'s next sentence will be parsed, or two other agents in the GRAEL society will be randomly selected to play a language game.

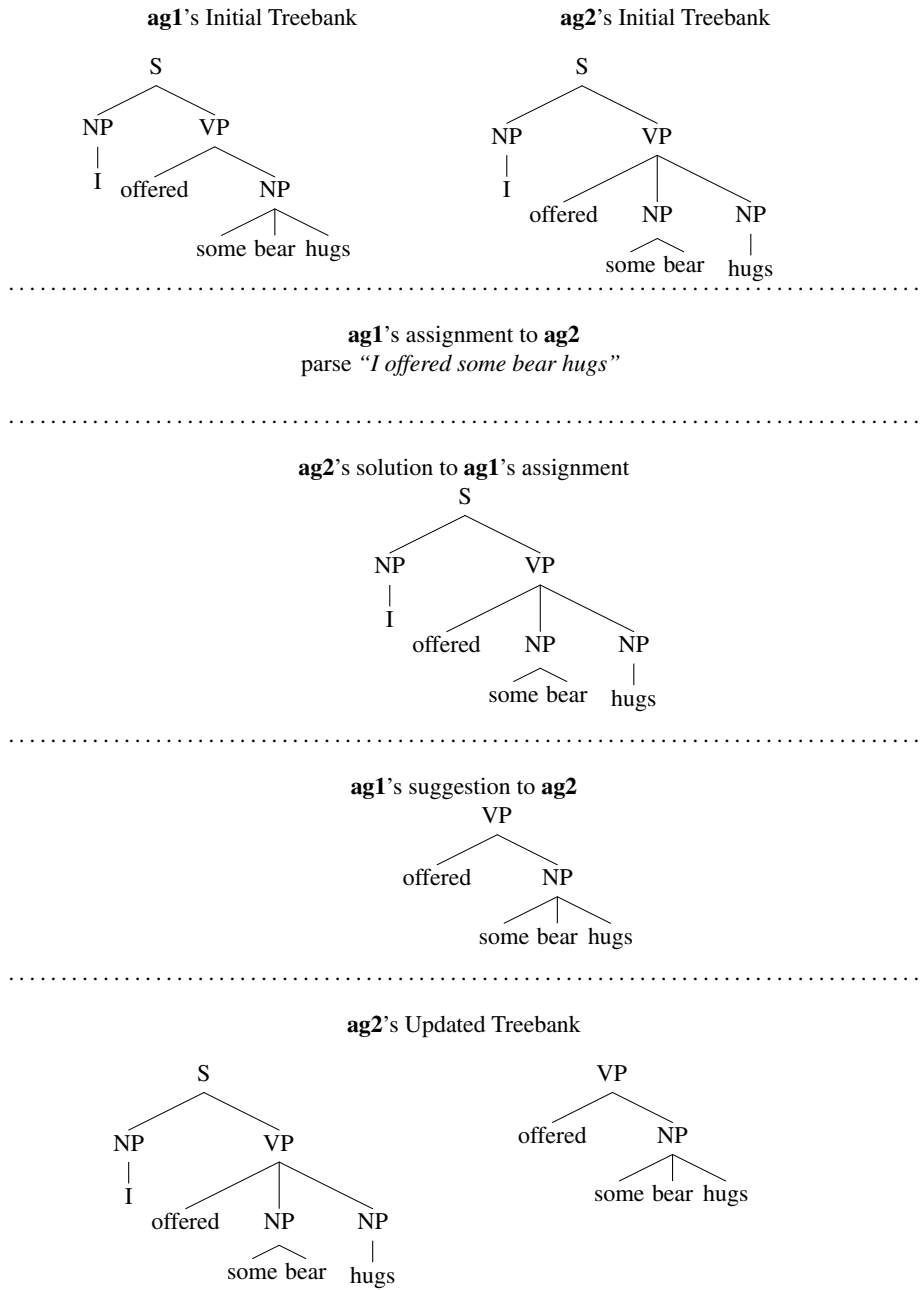


Fig. 1. A GRAEL language game

Generations

This type of interaction is based on a high amount of knowledge sharing between agents. This extends the agents' grammars very fast, so that their datasets can grow very large in a short period of time. It is therefore beneficial to introduce new generations in the GRAEL society from time to time. This not only allows for tractable computational processing times, but also allows the society to purge itself of bad agents and build new generations of good parser agents, who contain a fortuitous distribution of grammatical knowledge.

This introduces a neo-darwinist aspect in the system and involves the use of fitness functions that can distinguish good agents from bad ones. Typically, we define the fitness of an agent in terms of its parsing accuracy (i.e. the number of correct analyses), but we can also require the agents to have fast and efficient grammars and the like. The use of fitness functions and generations ideally makes sure that the required type of grammatical knowledge is retained throughout different generations, while useless grammatical knowledge can be marginalized over time.

Unfortunately, it is not feasible to provide a detailed description of the evolutionary parameters in the context of this overview paper. We would like to refer to [12] for specific details on the architecture and parameters of the GRAEL environment.

4 GRAEL-1: Probabilistic Grammar Optimization

GRAEL-1 is the most straightforward instantiation of GRAEL and deals with probabilistic grammar optimization. Developing a corpus-based parser requires inducing a grammar from an annotated corpus and using it to parse new sentences. Typically, these grammars are very large, so that for any given sentence a huge amount of possible parses is generated (a *parse forest*), only one of which provides the correct analysis for the sentence. Parsing can therefore be considered as a two-step process: first a parser generates all possible parses for a sentence, after which a disambiguation step makes sure the correct analysis is retrieved from the parse forest. Fortunately, we can also induce probabilistic information from the annotated corpus² that provides a way to rank the analyses in the parse forests in order of probabilistic preference. Even though these statistics go a long way in providing well ordered parse forests, it can be observed in many cases that the ranking of the parse forest is sometimes counter-intuitive in that correct constructs are often overtaken by obviously erroneous, but highly frequent structures.

With GRAEL-1, we propose an agent-based evolutionary computing method to resolve the issue of suboptimal probability mass distribution: by distributing the knowledge over a group of agents and having them interact with each other, we basically create a multiple-route model for probabilistic grammar optimization. Grammatical structures extracted from the training corpus, will be present in different quantities and variations throughout the GRAEL society (similarly to the aforementioned machine learning method of *bagging*). While the agents interact with each other and in effect practice the task on each other's grammar, a varied range of probabilistic grammars are optimized in a situation that directly relates to the task at hand (similarly to the machine learning method of *boosting*).

² This is achieved by observing the relative frequency of grammatical constructs in the annotated corpus.

4.1 Experimental Setup

In the GRAEL-I experiments, we measure two types of accuracy: the baseline accuracy is measured by directly inducing a grammar from the training set to power a parser, which disambiguates the test set. The same training set is then randomly distributed over a number of agents in the GRAEL society, who will consequently engage in a number of *language games*. At some point, the society is halted and the fittest agent is selected from the society. This agent effectively constitutes a redistributed and probabilistically optimized grammar, which can be used to power another parser. GRAEL-I accuracy is achieved by having this parser disambiguate the same test set as the baseline parser.

Data and tools. Two data sets from the Penn Treebank [13] were used. The main batch of experiments was conducted on an edited version of the small, homogeneous ATIS-corpus, which consists of a collection of annotated sentences recorded by a spoken-dialogue system. The larger Wall Street Journal Corpus (henceforth WSJ), a collection of annotated newspaper articles, was used to test the system on a larger scale corpus. We used the parsing system PMPG [16], which combines a CKY parser [18] and a post-parsing parse forest reranking scheme that employs probabilistic information as well as a memory-based operator that ensures that larger syntactic contexts are considered during parsing.

Apart from different corpora, we also experimented on different society sizes (5, 10, 20, 50, 100 agents), generation methods, fitness functions and methods to determine when to halt a GRAEL society. An exhaustive overview of all experimental parameters can be found in [12], but we will briefly outline some key notions. New generations are created as follows: if an agent is observed not to acquire any more rules over the course of n communicative attempts³, it is considered to be an end-of-life agent. As soon as two end-of-life agents are available that belong to the 50% fittest agents in the society, they are allowed to procreate by crossing over the grammars they have acquired during their lifespan. This operation yields three new agents, two of which will take their ancestors' slots in the society, while the other one takes the slot of the oldest agent among the 50% unfit agents at that point in time. The fitness of an agent is defined by recording a weighted average of the F-score (see below) during inter-agent communication and the F-score of the agent's parser on a held-out validation set. This information was also used to try and halt the society at a global maximum and to select the fittest agent from the society. For computational reasons, the experiments on the WSJ-corpus were limited to two different population sizes (50 and 100) and used an approximation of GRAEL that can deal with large datasets in a reasonable amount of time.

Note that GRAEL-I includes two different notions of crossover, although either one is a far stretch from the classic GA-type definition of the concept. The first type of crossover occurs during the language game when parts of syntactic tree-structures are being shared between agents. This operation relates to the recombination of knowledge aspect of crossover. The second type of crossover occurs when new agents are created by crossing over the grammars of two end-of-life agents. Again, the aspect of recombination is apparent in this operation. Note however the distinction with crossover in the context

³ n is by default set to the number of agents in a society.

of genetic algorithms in that neither crossover operation in the GRAEL system occurs on the level of the genotype and that it is in fact the phenotype that is being adapted and which evolves over time⁴.

Table 1. Baseline vs. GRAEL-I results

	ATIS		WSJ	
	Exact Match	$F_{\beta=1}$ -score	Exact Match	$F_{\beta=1}$ -score
Baseline	70.7	89.3	16.0	80.5
GRAEL (5)	72.4	90.9	—	—
GRAEL (10)	77.6	92.1	—	—
GRAEL (20)	77.6	92.1	—	—
GRAEL (50)	75.9	92.2	22.2	80.7
GRAEL (100)	75.9	92.0	22.8	81.1

4.2 Results

Table 1 displays the results of these experiments. Exact Match Accuracy expresses the percentage of sentences that were parsed completely correct, while the F-score is a measure of how well the parsers work on a constituent level. The baseline model is a standard PMPG parser using a grammar directly induced from the training set. Table 1 also displays scores of the GRAEL system for different population sizes. We notice a significant gain for all GRAEL models over the baseline model on the ATIS corpus. The small society of 5 agents achieves only a very limited improvement over the baseline method. Data analysis showed that the best moment to halt the society and select the fittest agent from the society, is a relatively brief period right before actual convergence sets and grammars throughout the society are starting to resemble each other more closely. The size of the society seems to be the determining factor controlling the duration of this period. In smaller societies, it may occur that convergence sets in too fast, since there is a narrower spread of data throughout the society. This causes convergence to set in prematurely, before the halting procedures even had a chance to register a proper halting point for the society. Hence, the low accuracy for the 5 agent society on the ATIS corpus.

Some preliminary experiments on a subset of the WSJ corpus had shown that society sizes of 20 agents and less to be unsuitable for a large-scale corpus, again ending up in a harmful premature stagnation. The gain achieved by the GRAEL society is less spectacular than on the ATIS corpus, but it is still statistically significant. Larger society sizes and full GRAEL processing on the WSJ corpus should achieve a larger gain, but is not currently feasible due to computational constraints.

The results show that GRAEL-I is indeed an interesting method for probabilistic grammar redistribution and optimization. Data analysis shows that many of the counter-intuitive parse forest orderings that were apparent in the baseline model, are being

⁴ Theoretically, this method relates to the empiricist point of view of language acquisition, rather than the nativist point of view.

resolved after GRAEL-1 processing. It is also interesting to point out that we are achieving an error reduction rate of more than 26% over the baseline method, without introducing any new grammatical information in the society, but solely by redistributing what is already there.

5 GRAEL-2: Grammar Rule Discovery

Any type of grammar, be it corpus-induced or hand-written will not be able to cover all sentences of language. Some sentences will indeed require a rule that is not available in the grammar. Even for a large corpus such as the *wsj*, missing grammar rules provide a serious accuracy bottleneck. We therefore set out to find a method that can take a grammar and improve its coverage by generating new rules. But doing so in an unguided manner, would yield huge, over-generating grammars, containing many nonsensical rules. The GRAEL-2 system described in this section provides a guidance mechanism to grammar rule discovery. In GRAEL-2, the original grammar is distributed among a group of agents, who can randomly *mutate* the grammatical structures they hold. The new grammatical information they create is tried and tested by interacting with each other. The neo-darwinist aspect of this evolutionary system tries to retain any useful mutated grammatical information throughout the population, while noise is filtered out over time. This method provides a way to create new grammatical structures previously unavailable in the corpus, while at the same time evaluating them in a practical context, without the need for an external information source.

Some minor alterations need to be made to the initial GRAEL-1 system to accomplish this, most notably the addition of an element of mutation. This occurs in the context of a language game (cf. Figure 1) at the point where **ag1** suggests the minimal correct substructure to **ag2**. In GRAEL-1 this step introduced a form of error-driven learning, making sure that the probabilistic value of this grammatical structure is increased. The functionality of GRAEL-2 however is different: we assume that there is a virtual noisy channel between **ag1** and **ag2** which may cause **ag2** to misunderstand **ag1**'s structure. Small mutations on different levels of the substructure may occur, such as the deletion, addition and replacement of nodes in the tree-structure. This mutation introduces previously unseen grammatical data in the GRAEL society, some of which will be useless (and will hopefully disappear over time), some of which will actually constitute good grammar rules.

Note again that the concept of mutation in the GRAEL system, stretches the classic GA-notion. Mutation in GRAEL-2 does not occur on the level of the genotype at all. It is the actual grammatical information that is being mutated and consequently communicated throughout the society. This provides a significant speed-up of grammatical evolution over time, as well as enable a transparent insight into the grammar rule discovery mechanism itself.

Experimental Setup and Results. The GRAEL-2 experiments have a similar setup to the GRAEL-1 experiments (Section 4). For the experiments on the ATIS corpus, we compiled a special worst-case scenario test set to specifically test the grammar-rule discovery capabilities of GRAEL-2. This test set consists of 97 sentences that require a grammar

Table 2. Baseline vs GRAEL-1 vs GRAEL-2 vs GRAEL2+1 Results

	ATIS		WSJ	
	$F_{\beta=1}$	Ex. Match	$F_{\beta=1}$	Ex. Match
Baseline	69.8	0	80.5	16
grael-1	73.8	0	81.4	22.8
grael-2	83.0	7.2	76.5	19.3
grael2+1	85.7	11.3	81.6	23.4

rule that cannot be induced from the training set. For the WSJ-experiments the standard test set was used. A 20-agent and a 100 agent society were respectively used for the ATIS and WSJ experiments.

Table 2 compares the GRAEL-2 results to the baseline and GRAEL-1 systems. The latter systems trivially achieve an exact match accuracy of 0% on the ATIS test set, which also has a negative effect on the F-score (Table 2). GRAEL-2 is indeed able to improve on this significantly. The results on the WSJ corpus show however that GRAEL-2 has lost the beneficial probabilistic optimization effect that was paramount to GRAEL-1. Another experiment was therefore conducted in which we turned the GRAEL-2 society into a GRAEL-1 society after the former's halting point. In other words: we take a society of agents using mutated information and consequently apply GRAEL-1 probabilistic redistribution on these grammars. This achieves a significant improvement on all data sets and establishes an interesting grammar development technique that is able to extend and optimize any given grammar without the need for an external information source.

6 GRAEL-3: Unsupervised Grammar Induction

In GRAEL-2 we started off with an initial grammar that is induced from an annotated corpus, so that we could consider it to be a form of supervised grammar induction, since we still require annotated data. Yet, annotated data is hard to come by and resources are necessarily limited. Recent research efforts however try to implement methods that can apply structure to raw data, simply on the basis of distributional properties and grammatical principles [19,20,21]. Further alterations to the GRAEL-2 system can extend its functionality to include this type of task.

The GRAEL-3 system requires us to develop a basic, but workable grammar induction module, that can build tree-structures on the basis of mutual information content calculated on bigrams. This module should not be considered a part of GRAEL-3 proper: any kind of grammar induction method can in principle be used to bootstrap structure in the society. Next, we performed three types of experiment for each data set: GRAEL-3A takes the whole training set and applies structure to those sentences based on information content values calculated on the entire data set. GRAEL-3B first distributes the sentences over the agents, after which the grammar induction module applies structure to these sentences based on information content values calculated for each agent individually. Since this grammar induction module effectively constitutes a parser, we also conducted some experiments in which parsing was only performed using this method (GRAEL-3AB-2).

The same training set/test set divisions were used as in the GRAEL-1 experiments, while the experiments were performed on a 20-agent society for the ATIS-corpus and a 100-agent society for the WSJ-corpus. Due to reasons of time, we did not perform the baseline experiment using the PMPG for the WSJ experiments, nor the GRAEL-3A-1 experiment. We measure the F-score and the zero-crossing brackets measure which is typically used to evaluate unsupervised grammar induction methods.

Table 3. GRAEL-3 Results

	ATIS		WSJ	
	$F_{\beta=1}$	OCB	$F_{\beta=1}$	OCB
Baseline (PMPG)	22.4	22.9	–	–
GRAEL-3A-1	25.6	24.9	–	–
GRAEL-3B-1	22.7	22.9	31.8	32.8
Baseline (GIM)	28.4	30.8	32.2	32.5
GRAEL-3AB-2	31.0	31.1	33.8	34.0

The first line of Table 3 shows the baseline accuracy using a PMPG on the training set annotated by the grammar induction module. These figures are very low, which is mainly due to the problematic labeling properties the grammar induction method imposes. A parser using ps-type rules such as PMPG does indeed need accurate node labels to be able to process grammatical structure in an accurate manner. The GRAEL-3A-1 system however is able to improve on this grammar significantly. Unsupervised grammar induction methods typically need a lot of data to achieve reasonable accuracy. It is therefore not surprising that GRAEL-3B does not achieve any improvement over the baseline, since the grammar induction method only has a very limited amount of data to extract useful information content values from. Using the grammar induction method as a parser itself circumnavigates the problem of labeling and this has a positive effect on parsing accuracy. More importantly, GRAEL-3 seems again able to improve parsing accuracy significantly, both on the ATIS and the WSJ corpus.

7 Concluding Remarks

This paper presented a broad overview of the GRAEL system, which can be used for different grammar optimization and induction tasks. We believe this to be one of the first research efforts that employs agent-based evolutionary computing as a machine learning method for data-driven grammar development. Using the same architecture and only applying minor alterations, we were able to implement three different tasks: GRAEL-1 provides a beneficial re-distribution of the probability mass of a probabilistic grammar by using a form of error-driven learning in the context of interactions between autonomous agents. By introducing an element of mutation, we extended GRAEL-1's functionality and projected GRAEL-2 as a workable grammar rule discovery method, significantly improving grammatical coverage on corpus-induced grammars. Following up GRAEL-2's grammar rule discovery method with GRAEL-1's probabilistic grammar optimization

proved to be an interesting optimization toolkit for corpus-induced grammars. Finally, we described GRAEL-3 as a first attempt to provide an unsupervised grammar induction technique. Even though the scores achieved by GRAEL-3 are rather modest compared to supervised approaches, the experiments show that the GRAEL environment is again able to take a collection of deficient grammars and turn them into better grammars through an extended process of inter-agent interaction.

The GRAEL framework provides an agent-based evolutionary computing approach to natural language grammar optimization and induction. It integrates the sensibilities of combinatory machine learning methods such as bagging and boosting, with the dynamics of evolutionary computing and agent-based processing. We have shown that GRAEL-1 and GRAEL-2 are able to take a collection of annotated data, providing an already well-balanced grammar, and squeeze more performance out of them without using an external information source. The experiments with GRAEL-3 showed however that it is equally able to improve on a collection of poor initial grammars, proving that the GRAEL framework is indeed able to provide an optimization for any type of grammar, regardless of its initial quality. This projects GRAEL as an interesting workbench for natural language grammar development, both for supervised, as well as unsupervised grammar optimization and induction tasks.

Acknowledgments. The research described in this paper was financed by the FWO (Fund for Scientific Research). The author would like to acknowledge Frederic Chappelier, for kindly making his parser available [18].

References

1. Daelemans, W., van den Bosch, A., Zavrel, J.: Forgetting exceptions is harmful in language learning. *Machine Learning, Special issue on Natural Language Learning* **34** (1999) 11–41
2. van Halteren, Hans, J.Z., Daelemans, W.: Improving accuracy in word class tagging through combination of machine learning systems. *Computational Linguistics* **27** (2) (2001) 199–230
3. Tjong Kim Sang, E., Daelemans, W., D'ejean, W., Koeling, H., Krymolowski, R., Punyakanok, Y., Roth, V.: Applying system combination to base noun phrase identification. In: *Proceedings of COLING 2000, Saarbruecken, Germany* (2000) 857–863
4. Breiman, L.: Bagging predictors. *Machine Learning* **24** (1996) 123–140
5. Abney, S., Schapire, R., Singer, Y.: Boosting applied to tagging and pp attachment. In: *Proceedings of the 1999 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*. (1999) 38–45
6. Henderson, J., Brill, E.: Bagging and boosting a treebank parser. In: *Proceedings of the 1st Meeting of the North American Chapter of the Association for Computational Linguistics (NAACL-2000)*. (2000) 34–41
7. Smith, T.C., Witten, I.H.: Learning language using genetic algorithms. In Wermter, S., Riloff, E., Scheler, G., eds.: *Connectionist, Statistical, and Symbolic Approaches to Learning for Natural Language Processing*. Volume 1040 of *LNAI*. Springer Verlag, Berlin (1996) 132–145
8. Wyard, P.: Context-free grammar induction using genetic algorithms. In Belew, R., Booker, L., eds.: *Proceedings of the Fourth International Conference on Genetic Algorithms*, San Mateo, ICGA, Morgan Kaufmann (1991) 514–518

9. Antonisse, H.J.: A grammar-based genetic algorithm. In Rawlings, G.J.E., ed.: *Foundations of genetic algorithms*. Morgan Kaufmann, San Mateo (1991) 193–204
10. Araujo, L.: A parallel evolutionary algorithm for stochastic natural language parsing. In: *Proceedings of The Seventh International Conference on Parallel Problem Solving From Nature*, Granada, Spain (2002) 700–709
11. Losee, R.: Learning syntactic rules and tags with genetic algorithms for information retrieval and filtering: An empirical basis for grammatical rules. *Information Processing and Management* **32** (1995) 185–197
12. De Pauw, G.: *An Agent-Based Evolutionary Computing Approach to Memory-Based Syntactic Parsing of Natural Language*. PhD thesis, University of Antwerp, Antwerp, Belgium (2002)
13. Marcus, M.P., Santorini, B., Marcinkiewicz, M.: Building a large annotated corpus of english: the penn treebank. *Computational linguistics* **19** (1993) 313–330 Reprinted in Susan Armstrong, ed. 1994, *Using large corpora*, Cambridge, MA: MIT Press, 273–290.
14. Bod, R.: *Beyond Grammar—An Experience-Based Theory of Language*. Cambridge University Press, Cambridge, England (1998)
15. Collins, M.: *Head-driven Statistical Models for Natural Language Parsing*. PhD thesis, University of Pennsylvania, Pennsylvania, USA (1999)
16. De Pauw, G.: Aspects of pattern-matching in dop. In: *Proceedings of the 18th International Conference on Computational Linguistics*. (2000) 236–242
17. Steels, L.: The origins of syntax in visually grounded robotic agents. *Artificial Intelligence* **103** (1998) 133–156
18. Chappelier, J.C., Rajman, M.: A generalized cyk algorithm for parsing stochastic cfg. In: *Proceedings of Tabulation in Parsing and Deduction (TAPD'98)*, Paris (FRANCE) (1998) 133–137
19. van Zaanen, M., Adriaans, P.: Alignment-Based Learning versus EMILE: A comparison. In: *Proceedings of the Belgian-Dutch Conference on Artificial Intelligence (BNAIC)*; Amsterdam, the Netherlands. (2001) 315–322
20. Clark, A.: Unsupervised induction of stochastic context-free grammars using distributional clustering. In Daelemans, W., Zajac, R., eds.: *Proceedings of CoNLL-2001*, Toulouse, France (2001) 105–112
21. Yuret, D.: *Discovery of Linguistic Relations Using Lexical Attraction*. PhD thesis, MIT, Cambridge, MA (1998)

Solving Distributed Asymmetric Constraint Satisfaction Problems Using an Evolutionary Society of Hill-Climbers

Gerry Dozier

Department of Computer Science and Software Engineering
Auburn University, Auburn AL 36849-5347, USA
gvdozier@eng.auburn.edu

Abstract. The distributed constraint satisfaction problem (DisCSP) can be viewed as a 4-tuple (X, D, C, A) , where X is a set of n variables, D is a set of n domains (one domain for each of the n variables), C is a set of constraints that constrain the values that can be assigned to the n variables, and A is a set of agents for which the variables and constraints are distributed. The objective in solving a DisCSP is to allow the agents in A to develop a consistent distributed solution by means of message passing. In this paper, we present an evolutionary society of hill-climbers (ESoHC) that outperforms a previously developed algorithm for solving randomly generated DisCSPs that are composed of asymmetric constraints on a test suite of 2,800 distributed asymmetric constraint satisfaction problems.

1 Introduction

A DisCSP [17] can be viewed as a 4-tuple (X, D, C, A) , where X is a set of n variables, D is a set of n domains (one domain for each of the n variables), C is a set of constraints that constrain the values that can be assigned to the n variables, and A is a set of agents for which the variables and constraints are distributed. Constraints between variables belonging to the same agent are referred to as intra-agent constraints while constraints between the variables of more than one agent are referred to as inter-agent constraints. The objective in solving a DisCSP is to allow the agents in A to develop a consistent distributed solution by means of message passing. The constraints are considered private and are not allowed to be communicated to fellow agents due to privacy, security, or representational reasons [17]. When comparing the effectiveness of DisCSP-solvers the number of communication cycles (through the distributed algorithm) needed to solve the DisCSP at hand is more important than the number of constraint checks [17].

Many real world problems have been modeled and solved using DisCSPs [1, 2, 3, 5, 6, 12]; however, many of these models use mirrored (symmetric) inter-agent constraints. Since these inter-agent constraints are known by the agents involved in the constraint, they cannot be regarded as private. If these constraints were

truly private then the inter-agent constraints of one agent would be unknown to the other agents involved in those constraints. In this case the DisCSP would be composed of asymmetric constraints. To date, with the exception of [4,5,12], little research has been done on distributed asymmetric CSPs (DisACSPs).

In this paper, we demonstrate how a distributed restricted form of uniform mutation can be used to improve the effectiveness of a previously developed evolutionary computation (EC) for solving DisACSPs known as a society of hill-climbers (SoHC) [4]. We refer to this new algorithm as an evolutionary SoHC (ESoHC). Our results show that ESoHC outperforms SoHC on a test suite of 2,800 DisACSPs.

The remainder of this paper is organized as follows. In Section 2, we present an overview of constraint processing which includes an introduction to the concept of asymmetric constraints and presents a formula for predicting where the most difficult randomly generated asymmetric CSPs are located, known as the phase transition [4,7,13]. In Section 3, we introduce the SoHC concept and explain how our ESoHC operates. In Section 4, we present the results of applying SoHC and ESoHC to 800 randomly generated distributed DisACSPs. In this section, we also compare SoHC and ESoHC on an additional 2,000 randomly generated DisACSPs in order to better visualize their performance across the phase transition. In Section 5, we present our conclusions and future work.

2 CSPs, Asymmetric Constraints, and the Phase Transition

A CSP [15] can be viewed as triple $\langle X, D, C \rangle$ where X is set of variables, D is set of domains where each $x_i \in X$ takes its value from the corresponding domain $d_i \in D$, and where C is a set of r constraints. Consider a binary constraint network (one where each constraint constrains the values of exactly two variables)¹ $\langle X, D, C \rangle$ where $X = \{E, F, G\}$, $D = \{d_E = \{e_1, e_2, e_3\}, d_F = \{f_1, f_2, f_3\}, d_G = \{g_1, g_2, g_3\}\}$, and $C = \{c_{EF}, c_{EG}, c_{FG}\}$. Suppose that the constraints c_{EF}, c_{EG}, c_{FG} are as follows:

$$\begin{aligned} c_{EF} &= \{ \langle \mathbf{e1}, \mathbf{f2} \rangle, \langle \mathbf{e1}, \mathbf{f3} \rangle, \langle \mathbf{e2}, \mathbf{f2} \rangle, \langle \mathbf{e3}, \mathbf{f2} \rangle \} \\ c_{EG} &= \{ \langle \mathbf{e2}, \mathbf{g3} \rangle, \langle \mathbf{e3}, \mathbf{g1} \rangle \} \\ c_{FG} &= \{ \langle \mathbf{f2}, \mathbf{g1} \rangle, \langle \mathbf{f2}, \mathbf{g3} \rangle \}. \end{aligned}$$

Constraint networks possess two additional attributes: tightness and density. The tightness of a constraint is the ratio of the number of tuples disallowed by the constraint to the total number of tuples in $d_i \times d_j$. The average constraint tightness of a binary constraint network is the sum of the tightness of each constraint divided by the number of constraints in the network. The density of a constraint network is the ratio of the number of constraints in the network to the total number of constraints possible.

¹ In this paper, we only consider binary constraint networks because any constraint that involves more than two variables can be transformed into a set of binary constraints [15].

2.1 Asymmetric Constraints

Constraints in a binary constraint network may also be represented as two directional constraints referred to as arcs [8,15]. For example, the symmetric constraint c_{EF} can be represented as $c_{EF} = \{c_{EF}^{\rightarrow}, c_{EF}^{\leftarrow}\}$, where $c_{EF}^{\rightarrow} = c_{EF}^{\leftarrow} = \{(\mathbf{e1}, \mathbf{f2}), (\mathbf{e1}, \mathbf{f3}), (\mathbf{e2}, \mathbf{f2}), (\mathbf{e3}, \mathbf{f2})\}$, where $c_{EF}^{\rightarrow}$ represents the directional constraint imposed on variable F by variable E, and where $c_{EF}^{\leftarrow}$ represents the directional constraint imposed on variable E by variable F. This view of a symmetric binary constraint admits the possibility of an asymmetric binary constraint between variables E and F as one where $c_{EF}^{\rightarrow} \neq c_{EF}^{\leftarrow}$.

2.2 Predicting the Phase Transition

Classes of randomly generated CSPs can be represent as a 4-tuple $(n, m, p1, p2)$ [13] where n is the number of variables in X , m is the number of values in each domain, $d_i \in D$, $p1$ represents the constraint density, the probability that a constraint exists between any two variables, and $p2$ represents the tightness of each constraint.

Smith [13] developed a formula for determining where the most difficult symmetric randomly generated CSPs can be found. This equation is as follows, where $\hat{p2}_{crit_S}$ is the critical tightness at the phase transition for $n, m, p1$.

$$\hat{p2}_{crit_S} = 1 - m^{\frac{-2}{p1(n-1)}} \quad (1)$$

Randomly generated symmetric CSPs of the form $(n, m, p1, \hat{p2}_{crit_S})$ have been shown to be the most difficult because they have on average only one solution. Problems of this type are at the border (phase transition) between those classes of CSPs that have solutions and those that have no solution. Classes of randomly generated symmetric CSPs for which $p2$ is relatively small compared to $\hat{p2}_{crit_S}$ are easy to solve because they contain a large number of solutions. Similarly, classes of CSPs where $p2$ is relatively large compared to $\hat{p2}_{crit_S}$, are easy to solve because the constraints are so tight that simple backtrack-based CSP-solvers [13] can quickly determine that no solution exists. Thus, for randomly generated CSPs, one will observe an easy-hard-easy transition as $p2$ is increased from 0 to 1.

Smith's equation can be modified [4] to predict the phase transition in randomly generated asymmetric CSPs as well. This equation is as follows where $p1_\alpha$ represents the probability that an arc exists between two variables and where $\hat{p2}_{crit_A}$ is the critical tightness at the phase transition for n, m , and $p1_\alpha$.

$$\hat{p2}_{crit_A} = 1 - m^{\frac{-1}{p1_\alpha(n-1)}}. \quad (2)$$

3 Society of Hill-Climbers

A society of hill-climbers (SoHC) [4,11] is a collection of hill-climbers that search in parallel and communicate promising (or futile) directions of search to one

another through some type of external collective structure. In the society of hill-climbers that we present in this paper, the external collective structure which records futile directions of search comes in the form of a distributed list of breakout elements, where each breakout element corresponds to a previously discovered nogood² of a local minimum [10]. Before presenting the society of hill-climbers concept, we must first discuss the distributed hill-climber that makes up the algorithm. In this section, we first introduce a modified version of Yokoo's distributed breakout algorithm with broadcasting [17] (mDBA) which is based on Morris' Breakout Algorithm [10]. After introducing mDBA we will describe the framework of a SoHC.

For the mDBA, each agent $a_i \in A$ is responsible for the value assignment of exactly one variable. Therefore agent a_i is responsible for variable $x_i \in X$, can assign variable x_i one value from domain $d_i \in D$, and has as constraints $C_{x_i x_j}^{\rightarrow}$ where $i \neq j$. The objective of agent a_i is to satisfy all of its constraints $C_{x_i x_j}^{\rightarrow}$. Each agent also maintains a breakout management mechanism (BMM) that records and updates the weights of all of the breakout elements corresponding to the nogoods of discovered local minima. This distributed hill-climber seeks to minimize the number of conflicts plus the sum of all of the weights of the violated breakout elements.

3.1 The mDBA

The mDBA used in our SoHCs is very similar to Yokoo's DBA+BC with the major exceptions being that each agent broadcasts to every other agent the number of conflicts that its current value assignment is involved in. This allows the agents to calculate the total number of conflicts (fitness) of the current best distributed candidate solution (dCS) and to know when a solution has been found (when the fitness is equal to zero). The mDBA, as outlined in Figure 1, is as follows.

Initially, each agent, a_i , randomly generates a value $v_i \in d_i$ and assigns it to variable x_i . Next, each agent broadcasts its assignment, $x_i = v_i$, to its neighbors $a_k \in Neighbor_i$ where $Neighbor_i$ ³ is the set of agents that a_i is connected with via some constraint. Each agent then receives the value assignments of every neighbor. This collection of value assignments is known as the **agent.view** of an agent a_i [17]. Given the agent.view, agent a_i computes the number of conflicts that the assignment ($x_i = v_i$) is involved in. This value is denoted as γ_i .

Once the number of conflicts, γ_i , has been calculated, each agent a_i randomly searches through its domain, d_i , for a value $b_i \in d_i$ that resolves the greatest number of conflicts (ties broken randomly). The number of conflicts that an agent can resolve by assigning $x_i = b_i$ is denoted as r_i . Once γ_i and r_i have been computed, agent a_i broadcasts these values to each of its neighbors.

When an agent receives the γ_j and r_j values from each of its neighbors, it sums up all γ_j including γ_i and assigns this sum to f_i where f_i represents the

² A nogood is a tuple that causes a conflict.

³ In this paper, $Neighbor_i = A - \{a_i\}$.

fitness of the current dCS. If agent a_i has the highest r_i value of its neighborhood then agent a_i sets $v_i = b_i$, otherwise agent a_i leaves v_i unchanged. Ties are broken randomly using a commonly seeded tie-breaker⁴ that works as follows: if $t(i) > t(j)$ then a_i is allowed to change otherwise a_j is allowed to change where $t(k) = (k + \text{rnd}()) \bmod |A|$, and where $\text{rnd}()$ is a commonly seeded random number generator used exclusively for breaking ties.

If r_i for each agent is equal to zero, i.e. if none of the agents can resolve any of their conflicts, then the current best solution is a local minimum and all agents a_i send the nogoods that violate their constraints to their BMM_i . An agent's BMM will create a breakout element for all nogoods that are sent to it. If a nogood has been encountered before in a previous local minimum then the weight of its corresponding breakout element is incremented by one. All weights of newly created breakout elements are assigned an initial value of one. Therefore the task for mDBA is to reduce the total number of conflicts plus the sum of all breakout elements violated.

After the agents have decided who will be allowed to change their value and invoked their BMMs (if necessary), the agents check their f_i value. If $f_i > 0$ the agents begin a new cycle by broadcasting their value assignments to each other. If $f_i = 0$ the algorithm terminates with a distributed solution.

3.2 The Simple and Evolutionary SoHCs

The SoHCs reported in this paper are based on mDBA. Each SoHC runs ρ mDBA hill-climbers in parallel, where ρ represents the society size. Each of the ρ hill-climbers communicate with each other indirectly through a distributed BMM. Figure 2 provides a simplified view of a simple SoHC. Notice in Figure 2 that each agent, a_i assigns values variables $x_{i1}, x_{i2}, \dots, x_{i\rho}$ where each variable x_{ij} represents the i^{th} variable for the j^{th} dCS. Each agent, a_i , has a local BMM (BMM_i) which manages the breakout elements that correspond to the nogoods of its constraints.

The ESoHC works exactly like the SoHC mentioned earlier except for on each cycle a distributed restricted uniform mutation operator is applied as follows. Each distributed candidate solution, dCS_j , that has an above average number of conflicts is replaced with an offspring that is a mutated version of the best individual, dCS_q , as follows. Given a distributed individual, dCS_k , that is involved in an above average number of conflicts, with probability μ , agent a_i will randomly assign v_{ik} a value from d_i and with probability $1 - \mu$ agent a_i will set $v_{ik} = v_{iq}$. Of course, μ is referred to as the mutation rate. We refer to this form of mutation as distributed restricted uniform mutation (dRUM- μ).

⁴ In case of a tie between two agents a_i and a_j , Yokoo's DBA+BC will allow the agent with the lower agent address to change its current value assignment. We refer to this as the deterministic tie-breaker (DTB) method

```

procedure MDBA(Agent  $a_i$ ) {
  Step 0: randomly assign  $v_i \in d_i$  to  $x_i$ ;
  do {
    Step 1: broadcast  $(x_i = v_i)$  to other agents;
    Step 2: receive assignments from other agents, agent_view $_i$ ;
    Step 3: assign  $\gamma_i$  the number conflicts that  $(x_i = v_i)$ 
             is involved in;
    Step 4: randomly search for value  $b_i \in d_i$  that minimizes the
             number of conflicts of  $x_i$  (ties broken randomly),
    Step 5: let  $r_i$  be the number of conflicts resolved
             by  $(x_i = b_i)$ ;
    Step 6: broadcast  $\gamma_i$  and  $r_i$  to other agents;
    Step 7: receive  $\gamma_j$  and  $r_j$  from other agents, let  $f_i = \sum \gamma_k$ ;
    Step 8: if  $(\max(r_k) == 0)$ 
             for each conflict,  $(x_i = v, x_j = w)$ 
             update_breakout_elements( $BMM_i, (x_i, v, x_j, w)$ );
    Step 9: if  $(r_i == \max(r_k))^{\dagger}$ 
              $v_i = b_i$ ;
  } while  $(f_i > 0)$ 
}

```

[†] Ties are broken with randomly with a synchronized tie-breaker.

Fig. 1. The mDBA Protocol

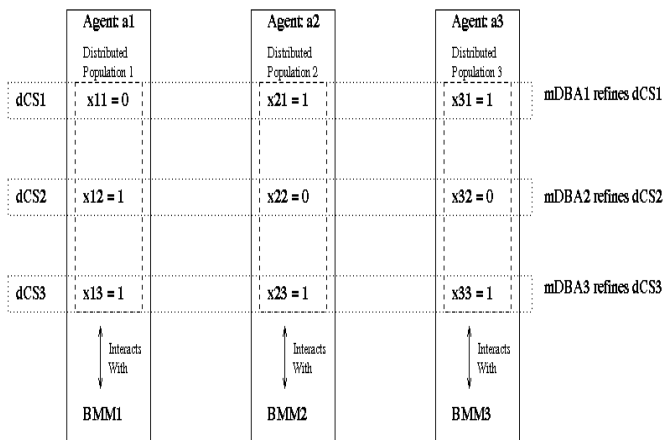


Fig. 2. An Simplified View of a SoHC with a Society Size of 3 dCSs

4 Results

4.1 Experiment I

In our first experiment, our test suite consisted of 800 instances of randomly generated DisACSPs of the form $\langle 30, 6, 1.0, p2 \rangle$ and $\langle 30, 6, p1_\alpha, 0.098 \rangle$. In this experiment, $p2$ took on values from the set, $\{0.03, 0.04, 0.05, 0.06\}$, for the 400 instances of $\langle 30, 6, 1.0, p2 \rangle$, where $\hat{p}_{crit_A}^2 \approx 0.06$ and $p1_\alpha$ took on values from the set, $\{0.3, 0.4, 0.5, 0.6\}$, for the 400 instances of $\langle 30, 6, p1_\alpha, 0.098 \rangle$, where instances of $\langle 30, 6, p1_\alpha = 0.6, 0.098 \rangle$ were at the phase transition. Each of the 30 agents randomly generated 29 arcs where each arc contained approximately 1.08, 1.44, 1.88, 2.16, and 3.53 nogoods respectively for $p2$ values of 0.03, 0.04, 0.05, 0.06, and 0.098. The arcs were generated according to a hybrid between Models A & B in [7]. This method of constraint generation is as follows. If each arc was to have 1.08 nogoods (which is the case when $p2 = 0.03$) then every arc received at least 1 nogood and was randomly assigned an additional nogood with probability 0.08. Similarly, if the average number of nogoods needed for each constraint was 2.16, (which is the case when $p2 = 0.06$) then every constraint received at least 2 nogoods and was randomly assigned an additional nogood with probability 0.16. The probability that an arc existed was determined with probability $p1_\alpha$.

Tables 1a-1d show the performance results of applying eight algorithms on each of the 100 instances of the $\langle 30, 6, 1.0, p2 \rangle$ classes of DisACSPs. The eight algorithms compared are mDBA-DTB, (the mDBA that uses Yokoo's deterministic tie-breaker algorithm to break ties between agents with the highest r_i value), six SoHCs with values of ρ taken from the set $\{1, 2, 4, 8, 16, 32\}$, and an ESoHC with $\rho = 32$ (ESoHC-32) that used dRUM-0.12.

In each table, the first column represents the algorithm. The second column represents the success rate (SR) that an algorithm had in finding a solution on the 100 problems when allowed a maximum of 2000 cycles. In the third column, the average number of cycles per run is recorded, and in the fourth column, the average number of constraint checks made by the algorithm is recorded.

When comparing the SoHCs, one can see that the larger the society size the better the performance is with respect to SR and average number of cycles. However using larger society sizes also results in an increased number of constraint checks and larger message sizes. In Table 1d, one can see that the classes of DisACSPs where $p2 = 0.06$ contains the most difficult problems.

It is important to realize that when comparing DisCSP-solvers that the most important criterion is success rate followed by communication cycles followed by the total number of constraint checks. For this reason, SoHC-32 has been selected as the overall best SoHC for the fully connected DisACSPs.

To make this point clearer, consider the communication medium used by the agents to be a network. With this being the case, we can compute the utilization of a link between any two agents given a normal 20 byte internet protocol (IP) packet header [16]. SoHC-01 will utilize $\frac{1}{20+1}$ of the bandwidth while SoHC-32

Table 1. Performances on the $\langle 30,6,1.0,0.03 \rangle$, $\langle 30,6,1.0,0.04 \rangle$, $\langle 30,6,1.0,0.05 \rangle$ and $\langle 30,6,1.0,0.06 \rangle$ DisACSPs

Alg.	SR	Cycles	Checks
mDBA-DTB	0.74	563.35	623849
SoHC-01	0.83	384.49	450684
SoHC-02	0.95	139.29	379128
SoHC-04	1.00	28.20	238510
SoHC-08	1.00	24.34	441775
SoHC-16	1.00	20.47	790013
SoHC-32	1.00	17.93	1448802
ESoHC-32	1.00	12.78	916539

(a) Performances on $\langle 30,6,1.0,0.03 \rangle$

Alg.	SR	Cycles	Checks
mDBA-DTB	0.60	989.30	1433954
SoHC-01	0.71	928.94	1422377
SoHC-02	0.93	444.35	1423055
SoHC-04	0.94	313.52	2122349
SoHC-08	1.00	124.22	1915596
SoHC-16	1.00	72.62	2456001
SoHC-32	1.00	54.28	3909942
ESoHC-32	1.00	26.09	1830178

(b) Performances on $\langle 30,6,1.0,0.04 \rangle$

Alg.	SR	Cycles	Checks
mDBA-DTB	0.07	1917.75	4218533
SoHC-01	0.10	1845.13	4149324
SoHC-02	0.11	1856.94	8466398
SoHC-04	0.17	1829.86	16838690
SoHC-08	0.22	1694.59	31591364
SoHC-16	0.36	1548.81	58666291
SoHC-32	0.52	1323.80	101607214
ESoHC-32	0.98	263.20	16101363

(c) Performances on $\langle 30,6,1.0,0.05 \rangle$

Alg.	SR	Cycles	Checks
mDBA-DTB	0.00	2000.00	5529200
SoHC-01	0.00	2000.00	5573943
SoHC-02	0.00	2000.00	11252052
SoHC-04	0.00	2000.00	22665737
SoHC-08	0.00	2000.00	45861472
SoHC-16	0.01	1982.02	91767502
SoHC-32	0.02	1981.06	184858509
ESoHC-32	0.11	1849.02	134172949

(d) Performances on $\langle 30,6,1.0,0.06 \rangle$

will utilize $\frac{32}{20+32}$ of the bandwidth. For this reason alone it is a welcomed result to see that larger society sizes lead to better performance results.

In Tables 1a-1d, the results also suggest that breaking ties randomly is more effective than breaking ties using Yokoo's deterministic tie-breaking method. This can be seen by comparing the success rates (SR) of mDBA-DTB and SoHC-01. Notice that in Tables 1a-1c that SoHC-01 has a slightly higher success rate.

When comparing SoHC-32 and ESoHC-32 in Tables 1a-1d, one can see that ESoHC-32 has the better performance on each of the four class of DisACSPs. As the tightness is increased from 0.03 to the critical value of 0.06, the difference in the performance as compared with SoHC-32 becomes more pronounced. At the phase transition, the SR of ESoHC-32 is $5\frac{1}{2}$ times better than the SR for SoHC-32.

Tables 2a-d show the performances of the eight algorithms on 100 instances of the $\langle 30,6,p1_\alpha,0.098 \rangle$ classes of DisACSPs. Notice that as $p1_\alpha$ is increased, in Tables 2a-2d, that the success rates of the algorithms decreases. Notice once again that the hardest DisACSPs seem to be located at the predicted phase transition, $\langle 30,6,0.6,0.098 \rangle$. Also in Table 2, one can see that mDBA-DTB outperforms SoHC-01 on the $\langle 30,6,0.3,0.098 \rangle$ class of DisACSPs but loses to SoHC-01 on the $\langle 30,6,0.4,0.098 \rangle$ and $\langle 30,6,0.5,0.098 \rangle$ classes. When compar-

Table 2. Performances on the $\langle 30,6,p1_{\alpha}.3,0.098 \rangle$ DisACSPs

Alg.	SR	Cycles	Checks
mDBA-DTB	0.64	758.98	896922
SoHC-01	0.63	790.50	904392
SoHC-02	0.97	307.26	735397
SoHC-04	1.00	36.37	280431
SoHC-08	1.00	30.57	490479
SoHC-16	1.00	21.53	791964
SoHC-32	1.00	18.20	1422854
ESoHC-32	1.00	12.77	894213

Alg.	SR	Cycles	Checks
mDBA-DTB	0.47	1197.23	1784095
SoHC-01	0.51	1163.29	1757059
SoHC-02	0.67	841.67	2534178
SoHC-04	0.89	450.68	2954802
SoHC-08	0.97	172.90	2461325
SoHC-16	1.00	129.86	4031357
SoHC-32	1.00	60.12	4184293
ESoHC-32	1.00	29.83	2009218

(a) Performances on $\langle 30,6,0.3,0.098 \rangle$ (b) Performances on $\langle 30,6,0.4,0.098 \rangle$

Alg.	SR	Cycles	Checks
mDBA-DTB	0.05	1960.59	4177278
SoHC-01	0.10	1864.31	3991908
SoHC-02	0.12	1859.26	8124593
SoHC-04	0.24	1700.12	14848626
SoHC-08	0.34	1600.72	28663041
SoHC-16	0.50	1432.35	52936951
SoHC-32	0.56	1154.75	85608964
ESoHC-32	0.96	279.36	16493167

Alg.	SR	Cycles	Checks
mDBA-DTB	0.00	2000.00	5329815
SoHC-01	0.00	2000.00	5459918
SoHC-02	0.01	1987.72	10844644
SoHC-04	0.00	2000.00	22344655
SoHC-08	0.02	1977.10	44391268
SoHC-16	0.01	1983.21	89557109
SoHC-32	0.03	1974.31	180966004
ESoHC-32	0.17	1755.67	126561756

(c) Performances on $\langle 30,6,0.5,0.098 \rangle$ (d) Performances on $\langle 30,6,0.6,0.098 \rangle$

ing SoHC-32 and ESoHC-32, the results are similar to what has been observed earlier. ESoHC-32 outperforms SoHC-32 on all classes of DisACSPs and has an SR that is $5\frac{2}{3}$ times greater than SoHC on the $\langle 30,6,0.6,0.098 \rangle$ class.

4.2 Experiment II

In the previous section, we presented the results of applying eight SoHCs to 800 randomly generated DisACSPs. In that presentation, we were only able to show the side of the phase transition where classes were likely to contain at least one solution [7,13]. This was done because the distributed hill-climbers presented in this paper are not complete; they cannot determine if the problem at hand has no solution at all. In order to visualize the phase transition in DisACSPs we used a technique introduced by Solmon in [14]. This technique is simple. When randomly generating a CSP make sure that it has at least one solution. Using the above approach will allow incomplete search algorithms to experience the easy-hard-easy behavior across the phase transition as tightness and/or density is increased from 0.0 to 1.0.

In order to visualize the phase transition for the $\langle 30,6,1.0,p2 \rangle$ classes of DisACSPs we randomly generated an additional 1,100 DisACSPs where $p2$ took values from the set, $\{0.03, 0.04, 0.05, 0.06, 0.065, 0.07, 0.08, 0.09, 0.1, 0.17, 0.25\}$.

For each value of p_2 , 100 DisACSPs were generated and guaranteed to have at least 1 solution. SoHC-32 and ESoHC-32 were then run on each of the problems with a maximum number of 2000 cycles in which to find a solution.

Figure 3 shows the performance results of SoHC-32 and ESoHC-32 on the 1,100 DisACSPs of the class $\langle 30,6,1.0,p_2 \rangle$. The range of values of p_2 (from 0.03 to 0.25) can be seen along the x-axis. Figure 3a shows the results in terms of the average number of cycles needed solve a DisACSP (shown on the y-axis) and Figure 3b shows the performance results in terms of failure rate (as shown on the y-axis).

In Figure 3a, one can see that the average number of cycles increases rapidly as p_2 is increased from 0.03 to about 0.065. For these problems, the actual phase transition seems to occur at $p_2 = 0.065$. As p_2 is increased beyond 0.065 one can see a rapid reduction in average number of cycles needed to find a solution. As the constraints become increasingly tighter, it becomes easier for SoHC-32 and ESoHC-32 to find the one and only solution. The shape of the curve in Figure 3b is very similar to the one shown in Figure 3a. Notice also that the performance of ESoHC-32 is superior to SoHC-32 over the total range of values for p_2 .

In order to visualize the phase transition as $p_{1\alpha}$ is increased, we created 900 randomly generated DisACSPs of the form $\langle 30,6,p_{1\alpha},0.098 \rangle$ where the arc density, $p_{1\alpha}$, took on values from the set $\{0.3, 0.4, 0.5, 0.6, 0.62, 0.7, 0.8, 0.9, 1.0\}$. Once again for each value of $p_{1\alpha}$ 100 DisACSPs were randomly generated, and the SoHCs were run on each problem with an allowed a maximum of 2000 cycles to find a solution.

Figure 4 shows the search behavior of SoHC-32 and ESoHC-32 as $p_{1\alpha}$ was increased from 0.3 to 1.0 in terms of the average number of cycles needed to find a solution as well as the failure rate. The results are similar to those shown in Figure 3; ESoHC-32 dramatically outperforms SoHC-32. However, Figures 3 and 4 differ in that the easy-hard-easy transition is less abrupt. The reason for this is that constraint tightness is a more sensitive predictor of the relative hardness of a CSP.

4.3 Discussion

The increased performance of ESoHC over SoHC is primarily due to the way in which the dRUM- μ operator intensifies search around the current best individual in the population. The basic assumption made by anyone applying an EC to a problem is that optimal (or near optimal) solutions are surrounded by good solutions. However, this assumption is not true for constrained problems. Even for problems where this is the case ECs typically employ local search in an effort to exploit promising regions. Thus, the EC will intensify search periodically in some region. Actually, the search behavior of ESoHC is no different. The individuals that are involved in a below average number of conflicts are allowed to continue to be refined by mDBA while individuals that are involved in an above average number of conflicts are replaced by offspring that more closely resemble the current best individual in the population. Upon closer inspection of the results in Tables 1 and 2 one can see that as ρ is increased in the SoHCs the

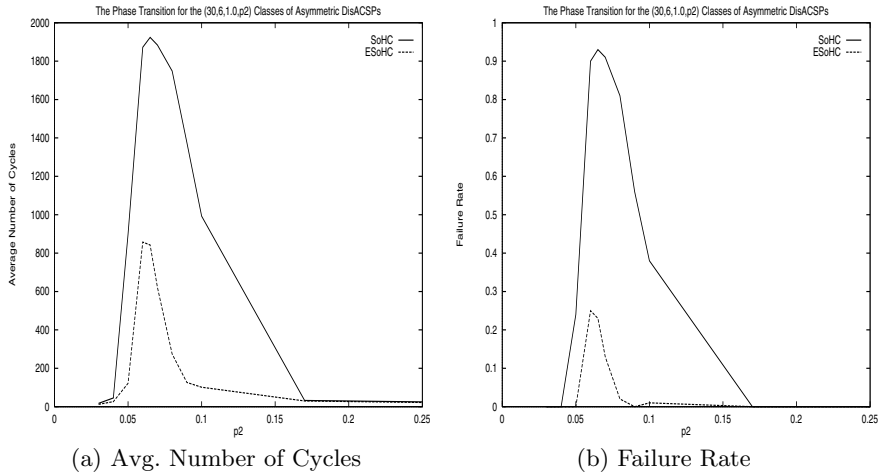


Fig. 3. Phase Transition Based on Avg. Number of Cycles and Failure Rate

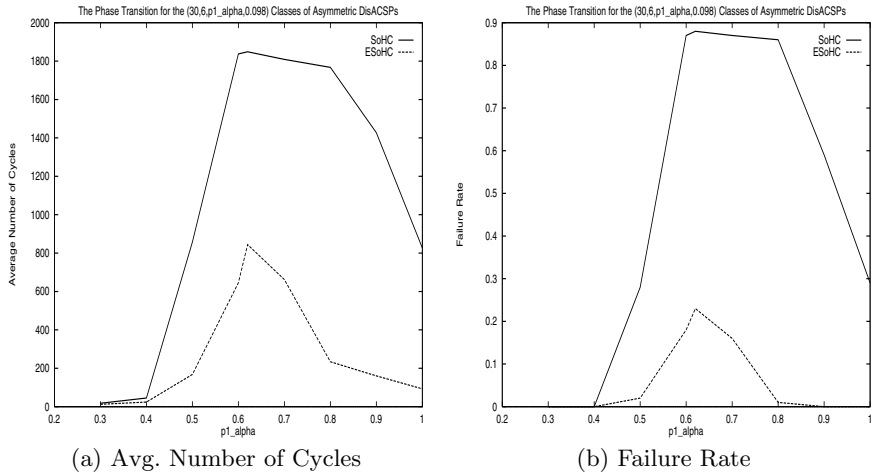


Fig. 4. Phase Transition Based on Avg. Number of Cycles and Failure Rate

performance gain diminishes. Therefore it seems reasonable, given a sufficiently large ρ , that half of the individuals can be used to intensify search without adversely affecting the convergence rate.

5 Conclusions and Future Work

In this paper, we have introduced the concept of DisACSPs and have demonstrated how distributed restricted uniform mutation can be used to improve the search of a society of hill-climbers on easy and difficult DisACSPs. We also provided a brief discussion of some of the reasons why the performance of ESoHC-32 is superior to SoHC-32. Our future work will include the development of other

distributed forms procreation that may increase the performance of ESoHC as well as the study of effect that different reallocation strategies have on the performance of ESoHC.

Acknowledgement. The author would like to thank the National Science Foundation for the support of this research under grant #IIS-9907377.

References

1. Bejar, R., Krishnamachari, B., Gomes, C., and Selman, B. (2001). "Distributed Constraint Satisfaction in a Wireless Sensor Tracking System", *Proc. of the IJCAI Workshop on Distributed Constraint Reasoning*, pp. 81–90.
2. Calisti, M., and Faltings, B. (2000). "Agent-Based Negotiations for Multi-Provider Interactions", *Proc. of the Intl. Sym. on Agent Sys. and Applications*, pp. 235–248.
3. Calisti, M., and Faltings, B. (2000). "Distributed Constrained Agents for Allocating Service Demands in Multi-Provider Networks", *Journal of the Italian Operational Society*, Special Issue on Constraint Problem Solving, vol. XXIX, no. 91.
4. Dozier, G. and Rupela, V. (2002). "Solving Distributed Asymmetric CSPs via a Society of Hill-Climbers", *Proc. of IC-AI'02*, pp. 949–953, CSREA Press.
5. Freuder, E. C., Minca, M., and Wallace, R. J. (2001). "Privacy/Efficiency Trade-offs in Distributed Meeting Scheduling by Constraint-Based Agents", *Proc. of the IJCAI Workshop on Distributed Constraint Reasoning*, pp. 63–71.
6. Krishnamachari, B., Bejar, R., and Wicker, S. (2002). "Distributed Problem Solving and the Boundaries of Self-Configuration in Multi-hop Wireless Networks", *Proc. of the Hawaii Intl. Conference on System Sciences*, HICSS-35.
7. MacIntyre, E., Prosser, P., Smith, B., and Walsh, T. (1998). "Random Constraint Satisfaction: Theory Meets Practice," *The Proc. of CP-98*, pp. 325–339.
8. Mackworth, A. K. (1977). "Consistency in networks of relations". *Artificial Intelligence*, 8 (1), pp. 99–118.
9. Modi, P. J., Jung, H., Tambe, M., Shen, W.-M., and Kulkarni, S. (2001). "Dynamic Distributed Resource Allocation: A Distributed Constraint Satisfaction Approach" *Proc. of the IJCAI Workshop on Distributed Constraint Reasoning*, pp. 73–79.
10. Morris, P. (1993). "The Breakout Method for Escaping From Local Minima," *Proc. of AAAI'93*, pp. 40–45.
11. Sebag, M. and Shoenauer, M. (1997). "A Society of Hill-Climbers," *The Proc. of ICEC-97*, pp. 319–324, IEEE Press.
12. Silaghi, M.-C., Sam-Haroud, D., Calisti, M., and Faltings, B. (2001). "Generalized English Auctions by Relaxation in Dynamic Distributed CSPs with Private Constraints", *Proc. of the IJCAI Workshop on Distributed Constraint Reasoning*, pp. 45–54.
13. Smith, B. (1994). "Phase Transition and the Mushy Region in Constraint Satisfaction Problems," *Proc. of ECAI-94*, pp. 100–104, John Wiley & Sons, Ltd.
14. Solnon, C. (2002). "Ants can solve constraint satisfaction problems", to appear in: *IEEE Transactions on Evolutionary Computation*, IEEE Press.
15. Tsang, E. (1993). *Foundations of Constraint Satisfaction*, Academic Press, Ltd.
16. Walrand, J. (1998). *Communication Networks: A First Course*, 2nd Edition, WCB/McGraw-Hill.
17. Yokoo, M. (2001). *Distributed Constraint Satisfaction*, Springer-Verlag.

Use of Multiobjective Optimization Concepts to Handle Constraints in Single-Objective Optimization

Arturo Hernández Aguirre¹, Salvador Botello Rionda¹, Carlos A. Coello Coello², and Giovanni Lizárraga Lizárraga¹

¹ Center for Research in Mathematics (CIMAT)

Department of Computer Science

Guanajuato, Gto. 36240, México

{artha,botello,giovanni}@cimat.mx

² CINVESTAV-IPN

Evolutionary Computation Group

Depto. de Ingeniería Eléctrica

Sección de Computación

Av. Instituto Politécnico Nacional No. 2508

Col. San Pedro Zacatenco

México, D. F. 07300

ccoello@cs.cinvestav.mx

Abstract. In this paper, we propose a new constraint-handling technique for evolutionary algorithms which is based on multiobjective optimization concepts. The approach uses Pareto dominance as its selection criterion, and it incorporates a secondary population. The new technique is compared with respect to an approach representative of the state-of-the-art in the area using a well-known benchmark for evolutionary constrained optimization. Results indicate that the proposed approach is able to match and even outperform the technique with respect to which it was compared at a lower computational cost.

1 Introduction

The success of Evolutionary Algorithms (EAs) in global optimization has triggered a considerable amount of research regarding the development of mechanisms able to incorporate information about the constraints of a problem into the fitness function of the EA used to optimize it [7]. So far, the most common approach adopted in the evolutionary optimization literature to deal with constrained search spaces is the use of penalty functions [10]. Despite the popularity of penalty functions, they have several drawbacks from which the main one is that they require a careful fine tuning of the penalty factors that indicates the degree of penalization to be applied [12].

Recently, some researchers have suggested the use of multiobjective optimization concepts to handle constraints in EAs. This paper introduces a new approach that is based on an evolution strategy that was originally proposed for multiobjective optimization: the Pareto Archived Evolution Strategy (PAES) [5]. Our approach (which is an extension of PAES) can be used to handle constraints in single-objective optimization problems and does not present the scalability problems of the original PAES. Besides using Pareto-based selection, our approach uses a secondary population (one of the most common

notions of elitism in evolutionary multiobjective optimization), and a mechanism that reduces the constrained search space so that our technique can approach the optimum more efficiently.

2 Problem Statement

We are interested in the general nonlinear programming problem in which we want to:

$$\text{Find } \mathbf{x} \text{ which optimizes } f(\mathbf{x}) \quad (1)$$

subject to:

$$g_i(\mathbf{x}) \leq 0, \quad i = 1, \dots, n \quad (2)$$

$$h_j(\mathbf{x}) = 0, \quad j = 1, \dots, p \quad (3)$$

where $\mathbf{x}$ is the vector of solutions $\mathbf{x} = [x_1, x_2, \dots, x_r]^T$, n is the number of inequality constraints and p is the number of equality constraints (in both cases, constraints could be linear or non-linear). For an inequality constraint that satisfies $g_i(\mathbf{x}) = 0$, then we will say that is active at $\mathbf{x}$. All equality constraints h_j (regardless of the value of $\mathbf{x}$ used) are considered active at all points of the feasible region $\mathcal{F}$.

3 Basic Concepts

A multiobjective optimization problem (MOP) has the following the form:

$$\text{Minimize } [f_1(\mathbf{x}), f_2(\mathbf{x}), \dots, f_k(\mathbf{x})] \quad (4)$$

subject to the m inequality constraints:

$$g_i(\mathbf{x}) \geq 0 \quad i = 1, 2, \dots, m \quad (5)$$

and the p equality constraints:

$$h_i(\mathbf{x}) = 0 \quad i = 1, 2, \dots, p \quad (6)$$

where k is the number of objective functions $f_i : \mathbb{R}^n \rightarrow \mathbb{R}$. We call $\mathbf{x} = [x_1, x_2, \dots, x_n]^T$ the vector of decision variables. We wish to determine from among the set $\mathcal{F}$ of all vectors which satisfy (5) and (6) the particular set of values $x_1^*, x_2^*, \dots, x_n^*$ which yield the optimum values of all the objective functions.

3.1 Pareto Optimality

A vector $\mathbf{u} = (u_1, \dots, u_k)$ is said to dominate $\mathbf{v} = (v_1, \dots, v_k)$ (denoted by $\mathbf{u} \preceq \mathbf{v}$) if and only if $\mathbf{u}$ is partially less than $\mathbf{v}$, i.e., $\forall i \in \{1, \dots, k\}, u_i \leq v_i \wedge \exists i \in \{1, \dots, k\} : u_i < v_i$. For a given multiobjective optimization problem, $\mathbf{f}(\mathbf{x})$, the Pareto optimal set ($\mathcal{P}^*$) is defined as:

$$\mathcal{P}^* := \{\mathbf{x} \in \mathcal{F} \mid \neg \exists \mathbf{x}' \in \mathcal{F} \quad \mathbf{f}(\mathbf{x}') \preceq \mathbf{f}(\mathbf{x})\}. \quad (7)$$

Thus, we say that a vector of decision variables $\mathbf{x}^* \in \mathcal{F}$ is *Pareto optimal* if there does not exist another $\mathbf{x} \in \mathcal{F}$ such that $f_i(\mathbf{x}) \leq f_i(\mathbf{x}^*)$ for all $i = 1, \dots, k$ and $f_j(\mathbf{x}) < f_j(\mathbf{x}^*)$ for at least one j . In words, this definition says that $\mathbf{x}^*$ is Pareto optimal if there exists no feasible vector of decision variables $\mathbf{x} \in \mathcal{F}$ which would decrease some criterion without causing a simultaneous increase in at least one other criterion. Unfortunately, this concept almost always gives not a single solution, but rather a set of solutions called the *Pareto optimal set*. The vectors $\mathbf{x}^*$ corresponding to the solutions included in the Pareto optimal set are called *nondominated*. The image of the Pareto optimal set under the objective functions is called *Pareto front*.

4 Related Work

The main idea of adopting multiobjective optimization concepts to handle constraints is to redefine the single-objective optimization of $f(\mathbf{x})$ as a multiobjective optimization problem in which we will have $m + 1$ objectives, where m is the total number of constraints. Then, we can apply any multiobjective optimization technique [3] to the new vector $\bar{v} = (f(\mathbf{x}), f_1(\mathbf{x}), \dots, f_m(\mathbf{x}))$, where $f_1(\mathbf{x}), \dots, f_m(\mathbf{x})$ are the original constraints of the problem. An ideal solution $\mathbf{x}$ would thus have $f_i(\mathbf{x})=0$ for $1 \leq i \leq m$ and $f(\mathbf{x}) \leq f(\mathbf{y})$ for all feasible $\mathbf{y}$ (assuming minimization).

These are the mechanisms taken from evolutionary multiobjective optimization that are more frequently incorporated into constraint-handling techniques:

1. Use of Pareto dominance as a selection criterion.
2. Use of Pareto ranking [4] to assign fitness in such a way that nondominated individuals (i.e., feasible individuals in this case) are assigned a higher fitness value.
3. Split the population in subpopulations that are evaluated either with respect to the objective function or with respect to a single constraint of the problem.

In order to sample the feasible region of the search space widely enough to reach the global optima it is necessary to maintain a balance between feasible and infeasible solutions. If this diversity is not reached, the search will focus only on one area of the feasible region. Thus, it will lead to a local optimum.

A multiobjective optimization technique aims to find a set of trade-off solutions which are considered good in all the objectives to be optimized. In global nonlinear optimization, the main goal is to find the global optimum. Therefore, some changes must be done to those approaches in order to adapt them to the new goal. Our main concern is that feasibility takes precedence, in this case, over nondominance. Therefore, good “trade-off” solutions that are not feasible cannot be considered as good as bad “trade-off” solutions that are feasible. Furthermore, a mechanism to maintain diversity must normally be added to any evolutionary multiobjective optimization technique. In our proposal, diversity is kept by using an adaptive grid, and by a selection process applied to the external file that maintains a mixture of both good “trade-off” and feasible individuals.

There are several approaches that have been developed using multiobjective optimization concepts to handle constraints, but due to space limitations we will not discuss them here (see for example [2,13,8,9]).

5 Description of IS-PAES

Our approach (called Inverted-Shrinkable Pareto Archived Evolution Strategy, or IS-PAES) has been implemented as an extension of the Pareto Archived Evolution Strategy (PAES) proposed by Knowles and Corne [5] for multiobjective optimization. PAES's main feature is the use of an adaptive grid on which objective function space is located using a coordinate system. Such a grid is the diversity maintenance mechanism of PAES and it constitutes the main feature of this algorithm. The grid is created by bisecting k times the function space of dimension $d = g + 1$. The control of 2^{kd} grid cells means the allocation of a large amount of physical memory for even small problems. For instance, 10 functions and 5 bisections of the space produce 2^{50} cells. Thus, the first feature introduced in IS-PAES is the "inverted" part of the algorithm that deals with this space usage problem. IS-PAES's fitness function is mainly driven by a feasibility criterion. Global information carried by the individuals surrounding the feasible region is used to concentrate the search effort on smaller areas as the evolutionary process takes place. In consequence, the search space being explored is "shrunk" over time. Eventually, upon termination, the size of the search space being inspected will be very small and will contain the solution desired. The main algorithm of IS-PAES is shown in Figure 1.

```

maxsize: max size of file
c: current parent  $\in X$  (decision variable space)
h: child of  $c \in X$ ,  $a_h$ : individual in file that dominates h
 $a_d$ : individual in file dominated by h
current: current number of individuals in file
cnew: number of individuals generated thus far
current = 1; cnew=0;
c = newindividual();
add(c);
While cnew  $\leq$  MaxNew do
    h = mutate(c); cnew++ = 1;
    if (c  $\preceq$  h) then Label A
    else if (h  $\preceq$  c) then { remove(c); add(g); c=h; }
    else if ( $\exists a_h \in \text{file} \mid a_h \preceq h$ ) then Label A
    else if ( $\exists a_d \in \text{file} \mid h \preceq a_d$ ) then {
        add(h);  $\forall a_d$  { remove( $a_d$ ); current-- = 1 }
    }
    else test(h,c,file)
Label A
    if (cnew % g==0) then c = individual in less densely populated region
    if (cnew % r==0) then shrinkspace(file)
End While

```

Fig. 1. Main algorithm of IS-PAES

The function **test(h,c,file)** determines if an individual can be added to the external memory or not. Here we introduce the following notation: $x_1 \square x_2$ means x_1 is located in

a less populated region of the grid than x_2 . The pseudo-code of this function is depicted in Figure 2.

```

if (current < maxsize) then {
    add(h);
    if (h  $\square$  c) then c=h }
else if ( $\exists a_p \in \text{file} \mid h \square a_p$ ) then {
    remove( $a_p$ ); add(h)
    if (h  $\square$  c) then c = h; }

```

Fig. 2. Pseudo-code of **test(h,c,file)**

5.1 Inverted “Ownership”

PAES keeps a list of individuals on every grid location, but in IS-PAES each individual knows its position on the grid. Therefore, building a sorted list of the most dense populated areas of the grid only requires to sort the k elements of the external memory. In PAES, this procedure needs to inspect all 2^{kd} locations in order to generate a list of the individuals sorted by the density of their current location in the grid. The advantage of the inverted relationship is clear when the optimization problem has many functions (more than 10), and/or the granularity of the grid is fine, for in this case only IS-PAES is able to deal with any number of functions and granularity level.

5.2 Shrinking the Objective Space

Shrinkspace(file) is the most important function of IS-PAES since its task is the reduction of the search space. The pseudo-code of **Shrinkspace(file)** is shown in Figure 3.

```

 $\underline{x}_{pob}$ : vector containing the smallest value of either  $x_i \in X$ 
 $\overline{x}_{pob}$ : vector containing the largest value of either  $x_i \in X$ 
select(file);
getMinMax( file,  $\underline{x}_{pob}$ ,  $\overline{x}_{pob}$ );
trim( $\underline{x}_{pob}$ ,  $\overline{x}_{pob}$  );
adjustparameters(file);

```

Fig. 3. Pseudo-code of **Shrinkspace(file)**

The function **select(file)** returns a list whose elements are the best individuals found in *file*. The size of the list is set to 15% of *maxsize*. Thus, the goal of **select(file)** is

```

m: number of constraints
i: constraint index
maxsize: max size of file
listsize: 15% of maxsize
constraintvalue(x,i): value of individual at constraint i
sortfile(file): sort file by objective function
worst(file,i): worst individual in file for constraint i
validconstraints={1,2,3,...,m};
i=firstin(validconstraints);
While (size(file) > listsize and size(validconstraints) > 0) {
    x=worst(file,i)
    if (x violates constraint i)
        file=delete(file,x)
    else validconstraints=removeindex(validconstraints,i)
if (size(validconstraints) > 0) i=nextin(validconstraints)
}
if (size(file) == listsize))
    list=file
else
    file=sort(file)
    list=copy(file,listsize) *pick the best listsize elements*

```

Fig. 4. Pseudo-code of **select(file)**

to create a list with: 1) only the best feasible individuals, 2) a combination of feasible and partially feasible individuals, or 3) the “best” infeasible individuals. The selection algorithm is shown in Figure 4. Note that *validconstraints* (a list of indexes to the problem constraints) indicates the order in which constraints are tested. One individual (the worst) is removed at a time in this loop of constraint testing till there is none to delete (all feasible), or 15% of the file is reached. The function **getMinMax(file)** finds the extreme values of the decision variables represented by the individuals of the list. Thus, the vectors $\underline{x}_{pob}$ and $\overline{x}_{pob}$ are found. Function **trim**($\underline{x}_{pob}, \overline{x}_{pob}$) shrinks the feasible space around the potential solutions enclosed in the hypervolume defined by the vectors $\underline{x}_{pob}$ and $\overline{x}_{pob}$. Thus, the function **trim**($\underline{x}_{pob}, \overline{x}_{pob}$) (see Figure 5) determines the new boundaries for the decision variables.

The value of β is the percentage by which the boundary values of either $x_i \in X$ must be reduced such that the resulting hypervolume H is a fraction α of its previous value. In IS-PAES all objective variables are reduced at the same rate β . Therefore, β can be deduced from α as discussed next. Since we need the new hypervolume to be a fraction α of the previous one, then

$$H_{\text{new}} \geq \alpha H_{\text{old}}$$

$$\prod_{i=1}^n (\overline{x}_i^{t+1} - \underline{x}_i^{t+1}) = \alpha \prod_{i=1}^n (\overline{x}_i^t - \underline{x}_i^t)$$


```

n: size of decision vector;
 $\bar{x}_i$ : actual upper bound of the  $i_{th}$  decision variable
 $\underline{x}_i$ : actual lower bound of the  $i_{th}$  decision variable
 $\bar{x}_{pob,i}$ : upper bound of  $i_{th}$  decision variable in population
 $\underline{x}_{pob,i}$ : lower bound of  $i_{th}$  decision variable in population
 $\forall i : i \in \{1, \dots, n\}$ 
   $slack_i = 0.05 \times (\bar{x}_{pob,i} - \underline{x}_{pob,i})$ 
   $width\_pob_i = \bar{x}_{pob,i} - \underline{x}_{pob,i}$ ;  $width_i^t = \bar{x}_i^t - \underline{x}_i^t$ 
   $deltaMin_i = \frac{\beta * width_i^t - width\_pob_i}{2}$ 
   $delta_i = \max(slack_i, deltaMin_i)$ ;
   $\bar{x}_i^{t+1} = \bar{x}_{pob,i} + delta_i$ ;  $\underline{x}_i^{t+1} = \underline{x}_{pob,i} - delta_i$ ;
  if  $(\bar{x}_i^{t+1} > \bar{x}_{original,i})$  then
     $\bar{x}_i^{t+1} = \bar{x}_{original,i}$ ;
  if  $(\underline{x}_i^{t+1} < \underline{x}_{original,i})$  then  $\underline{x}_i^{t+1} = \underline{x}_{original,i}$ ;
  if  $(\bar{x}_i^{t+1} > \bar{x}_{original,i})$  then  $\bar{x}_i^{t+1} = \bar{x}_{original,i}$ ;

```

Fig. 5. Pseudo-code of **trim**

Either x_i is reduced at the same rate β , thus

$$\begin{aligned}
 \prod_{i=1}^n \beta (\bar{x}_i^t - \underline{x}_i^t) &= \alpha \prod_{i=1}^n (\bar{x}_i^t - \underline{x}_i^t) \\
 \beta^n \prod_{i=1}^n (\bar{x}_i^t - \underline{x}_i^t) &= \alpha \prod_{i=1}^n (\bar{x}_i^t - \underline{x}_{i=1}^t) \\
 \beta^n &= \alpha \\
 \beta &= \alpha^{\frac{1}{n}}
 \end{aligned}$$

Summarizing, the search interval of each decision variable x_i is adjusted as follows (the complete algorithm is shown in Figure 3):

$$width_{new} \geq \beta \times width_{old}$$

In our experiments, $\alpha = 0.90$ worked well in all cases. Since α controls the shrinking speed, it can prevent the algorithm from finding the optimum if small values are chosen. In our experiments, values in the range [85%,95%] were tested with no effect on the performance. The last step of **shrinkspace()** is a call to **adjustparameters(file)**. The goal is to re-start the control variable σ using: $\sigma_i = (\bar{x}_i - \underline{x}_i) / \sqrt{n}$ $i \in (1, \dots, n)$. This expression is also used during the initial generation of the EA. In that case, the upper and lower bounds take the initial values of the search space indicated by the problem. The variation of the mutation probability follows the exponential behavior suggested by Bäck [1].

6 Comparison of Results

We have validated our approach with the benchmark for constrained optimization proposed in [7]. Our results are compared against the homomorphous maps [6], which is representative of the state-of-the-art in constrained evolutionary optimization. The following parameters were adopted for IS-PAES in all the experiments reported next: $maxsize = 200$, $bestindividuals = 15\%$, $slack = 0.05$, $r = 400$. The maximum number of fitness function evaluations was set to 350,000, whereas the results of Koziel and Michalewicz [6] were obtained with 1,400,000 fitness function evaluations.

1. **g01:** Min: $f(\mathbf{x}) = 5 \sum_{i=1}^4 x_i - 5 \sum_{i=1}^4 x_i^2 - \sum_{i=5}^{13} x_i$ subject to: $g_1(\mathbf{x}) = 2x_1 + 2x_2 + x_{10} + x_{11} - 10 \leq 0$, $g_2(\mathbf{x}) = 2x_1 + 2x_3 + x_{10} + x_{12} - 10 \leq 0$, $g_3(\mathbf{x}) = 2x_2 + 2x_3 + x_{11} + x_{12} - 10 \leq 0$, $g_4(\mathbf{x}) = -8x_1 + x_{10} \leq 0$, $g_5(\mathbf{x}) = -8x_2 + x_{11} \leq 0$, $g_6(\mathbf{x}) = -8x_3 + x_{12} \leq 0$, $g_7(\mathbf{x}) = -2x_4 - x_5 + x_{10} \leq 0$, $g_8(\mathbf{x}) = -2x_6 - x_7 + x_{11} \leq 0$, $g_9(\mathbf{x}) = -2x_8 - x_9 + x_{12} \leq 0$ where the bounds are $0 \leq x_i \leq 1$ ($i = 1, \dots, 9$), $0 \leq x_i \leq 100$ ($i = 10, 11, 12$) and $0 \leq x_{13} \leq 1$. The global optimum is at $x^* = (1, 1, 1, 1, 1, 1, 1, 1, 1, 3, 3, 3, 1)$ where $f(x^*) = -15$. Constraints g_1, g_2, g_3, g_4, g_5 and g_6 are active.

2. **g02:** Max: $f(\mathbf{x}) = \left| \frac{\sum_{i=1}^n \cos^4(x_i) - 2 \prod_{i=1}^n \cos^2(x_i)}{\sqrt{\sum_{i=1}^n x_i^2}} \right|$ subject to:

$$g_1(\mathbf{x}) = 0.75 - \prod_{i=1}^n x_i \leq 0$$

$$g_2(\mathbf{x}) = \sum_{i=1}^n x_i - 7.5n \leq 0 \quad (8)$$

where $n = 20$ and $0 \leq x_i \leq 10$ ($i = 1, \dots, n$). The global maximum is unknown; the best reported solution is [11] $f(x^*) = 0.803619$. Constraint g_1 is close to being active ($g_1 = -10^{-8}$).

3. **g03:** Max: $f(\mathbf{x}) = (\sqrt{n})^n \prod_{i=1}^n x_i$ subject to: $h(\mathbf{x}) = \sum_{i=1}^n x_i^2 - 1 = 0$ where $n = 10$ and $0 \leq x_i \leq 1$ ($i = 1, \dots, n$). The global maximum is at $x_i^* = 1/\sqrt{n}$ ($i = 1, \dots, n$) where $f(x^*) = 1$.
4. **g04:** Min: $f(\mathbf{x}) = 5.3578547x_3^2 + 0.8356891x_1x_5 + 37.293239x_1 - 40792.141$ subject to:

$$g_1(\mathbf{x}) = 85.334407 + 0.0056858x_2x_5 + 0.0006262x_1x_4 - 0.0022053x_3x_5 - 92 \leq 0$$

$$g_2(\mathbf{x}) = -85.334407 - 0.0056858x_2x_5 - 0.0006262x_1x_4 + 0.0022053x_3x_5 \leq 0$$

$$g_3(\mathbf{x}) = 80.51249 + 0.0071317x_2x_5 + 0.0029955x_1x_2 + 0.0021813x_3^2 - 110 \leq 0$$

$$g_4(\mathbf{x}) = -80.51249 - 0.0071317x_2x_5 - 0.0029955x_1x_2 - 0.0021813x_3^2 + 90 \leq 0$$

$$g_5(\mathbf{x}) = 9.300961 + 0.0047026x_3x_5 + 0.0012547x_1x_3 + 0.0019085x_3x_4 - 25 \leq 0$$

$$g_6(\mathbf{x}) = -9.300961 - 0.0047026x_3x_5 - 0.0012547x_1x_3 - 0.0019085x_3x_4 + 20 \leq 0 \quad (9)$$

where: $78 \leq x_1 \leq 102$, $33 \leq x_2 \leq 45$, $27 \leq x_i \leq 45$ ($i = 3, 4, 5$). The optimum solution is $x^* = (78, 33, 29.995256025682, 45, 36.775812905788)$ where $f(x^*) = -30665.539$. Constraints g_1 y g_6 are active.

5. **g05** Min: $f(\mathbf{x}) = 3x_1 + 0.000001x_1^3 + 2x_2 + (0.000002/3)x_2^3$ subject to:

$$g_1(\mathbf{x}) = -x_4 + x_3 - 0.55 \leq 0$$

$$g_2(\mathbf{x}) = -x_3 + x_4 - 0.55 \leq 0$$

$$h_3(\mathbf{x}) = 1000 \sin(-x_3 - 0.25) + 1000 \sin(-x_4 - 0.25) + 894.8 - x_1 = 0$$

$$h_4(\mathbf{x}) = 1000 \sin(-x_3 - 0.25) + 1000 \sin(x_3 - x_4 - 0.25) + 894.8 - x_2 = 0$$

$$h_5(\mathbf{x}) = 1000 \sin(-x_4 - 0.25) + 1000 \sin(x_4 - x_3 - 0.25) + 1294.8 = 0$$

(10)

where $0 \leq x_1 \leq 1200$, $0 \leq x_2 \leq 1200$, $-0.55 \leq x_3 \leq 0.55$, and $-0.55 \leq x_4 \leq 0.55$. The best known solution is $x^* = (679.9453, 1026.067, 0.1188764, -0.3962336)$ where $f(x^*) = 5126.4981$.

6. **g06** Min: $f(\mathbf{x}) = (x_1 - 10)^3 + (x_2 - 20)^3$ subject to:

$$g_1(\mathbf{x}) = -(x_1 - 5)^2 - (x_2 - 5)^2 + 100 \leq 0$$

$$g_2(\mathbf{x}) = (x_1 - 6)^2 + (x_2 - 5)^2 - 82.81 \leq 0$$

(11)

where $13 \leq x_1 \leq 100$ and $0 \leq x_2 \leq 100$. The optimum solution is $x^* = (14.095, 0.84296)$ where $f(x^*) = -6961.81388$. Both constraints are active.

7. **g07** Min:

$$\begin{aligned} f(\mathbf{x}) = & x_1^2 + x_2^2 + x_1x_2 - 14x_1 - 16x_2 + (x_3 - 10)^2 + 4(x_4 - 5)^2 \\ & + (x_5 - 3)^2 + 2(x_6 - 1)^2 + 5x_7^2 + 7(x_8 - 11)^2 + 2(x_9 - 10)^2 \\ & + (x_{10} - 7)^2 + 45 \end{aligned}$$

(12)

subject to:

$$g_1(\mathbf{x}) = -105 + 4x_1 + 5x_2 - 3x_7 + 9x_8 \leq 0$$

$$g_2(\mathbf{x}) = 10x_1 - 8x_2 - 17x_7 + 2x_8 \leq 0$$

$$g_3(\mathbf{x}) = -8x_1 + 2x_2 + 5x_9 - 2x_{10} - 12 \leq 0$$

$$g_4(\mathbf{x}) = 3(x_1 - 2)^2 + 4(x_2 - 3)^2 + 2x_3^2 - 7x_4 - 120 \leq 0$$

$$g_5(\mathbf{x}) = 5x_1^2 + 8x_2 + (x_3 - 6)^2 - 2x_4 - 40 \leq 0$$

$$g_6(\mathbf{x}) = x_1^2 + 2(x_2 - 2)^2 - 2x_1x_2 + 14x_5 - 6x_6 \leq 0$$

$$g_7(\mathbf{x}) = 0.5(x_1 - 8)^2 + 2(x_2 - 4)^2 + 3x_5^2 - x_6 - 30 \leq 0$$

$$g_8(\mathbf{x}) = -3x_1 + 6x_2 + 12(x_9 - 8)^2 - 7x_{10} \leq 0$$

(13)

where $-10 \leq x_i \leq 10$ ($i = 1, \dots, 10$). The global optimum is $x^* = (2.171996, 2.363683, 8.773926, 5.095984, 0.9906548, 1.430574, 1.321644, 9.828726, 8.280092, 8.375927)$ where $f(x^*) = 24.3062091$. Constraints g_1 , g_2 , g_3 , g_4 , g_5 and g_6 are active.

8. **g08** Max: $f(\mathbf{x}) = \frac{\sin^3(2\pi x_1) \sin(2\pi x_2)}{x_1^3(x_1 + x_2)}$ subject to: $g_1(\mathbf{x}) = x_1^2 - x_2 + 1 \leq 0$, $g_2(\mathbf{x}) = 1 - x_1 + (x_2 - 4)^2 \leq 0$ where $0 \leq x_1 \leq 10$ and $0 \leq x_2 \leq 10$. The optimum solution is located at $x^* = (1.2279713, 4.2453733)$ where $f(x^*) = 0.095825$. The solutions is located within the feasible region.

9. **g09** Min:

$$f(\mathbf{x}) = (x_1 - 10)^2 + 5(x_2 - 12)^2 + x_3^4 + 3(x_4 - 11)^2 + 10x_5^6 + 7x_6^2 + x_7^4 - 4x_6x_7 - 10x_6 - 8x_7 \quad (14)$$

subject to:

$$\begin{aligned} g_1(\mathbf{x}) &= -127 + 2x_1^2 + 3x_2^4 + x_3 + 4x_4^2 + 5x_5 \leq 0 \\ g_2(\mathbf{x}) &= -282 + 7x_1 + 3x_2 + 10x_3^2 + x_4 - x_5 \leq 0 \\ g_3(\mathbf{x}) &= -196 + 23x_1 + x_2^2 + 6x_6^2 - 8x_7 \leq 0 \\ g_4(\mathbf{x}) &= 4x_1^2 + x_2^2 - 3x_1x_2 + 2x_3^2 + 5x_6 - 11x_7 \leq 0 \end{aligned} \quad (15)$$

10. **g10** Min: $f(\mathbf{x}) = x_1 + x_2 + x_3$ subject to:

$$\begin{aligned} g_1(\mathbf{x}) &= -1 + 0.0025(x_4 + x_6) \leq 0 \\ g_2(\mathbf{x}) &= -1 + 0.0025(x_5 + x_7 - x_4) \leq 0 \\ g_3(\mathbf{x}) &= -1 + 0.01(x_8 - x_5) \leq 0 \\ g_4(\mathbf{x}) &= -x_1x_6 + 833.33252x_4 + 100x_1 - 83333.333 \leq 0 \\ g_5(\mathbf{x}) &= -x_2x_7 + 1250x_5 + x_2x_4 - 1250x_4 \leq 0 \\ g_6(\mathbf{x}) &= -x_3x_8 + 1250000 + x_3x_5 - 2500x_5 \leq 0 \end{aligned} \quad (16)$$

where $100 \leq x_1 \leq 10000$, $1000 \leq x_i \leq 10000$, ($i = 2, 3$), $10 \leq x_i \leq 1000$, ($i = 4, \dots, 8$). The global optimum is: $x^* = (579.3167, 1359.943, 5110.071, 182.0174, 295.5985, 217.9799, 286.4162, 395.5979)$, where $f(x^*) = 7049.3307$. g_1, g_2 and g_3 are active.

11. **g11** Min: $f(\mathbf{x}) = x_1^2 + (x_2 - 1)^2$ subject to: $h(\mathbf{x}) = x_2 - x_1^2 = 0$ where: $-1 \leq x_1 \leq 1$, $-1 \leq x_2 \leq 1$. The optimum solution is $x^* = (\pm 1/\sqrt{2}, 1/2)$ where $f(x^*) = 0.75$.

The comparison of results is summarized in Table 1. It is worth indicating that IS-PAES converged to a feasible solution in all of the 30 independent runs performed. The discussion of results for each test function is provided next (HM stands for homomorphous maps):

For **g01** both the best and the mean results found by IS-PAES are better than the results found by HM, although the difference between the worst and the best result is higher for IS-PAES. For **g02**, again both the best and mean results found by IS-PAES are better than the results found by HM, but IS-PAES has a (slightly) higher difference between its worst and best results. In the case of **g03**, IS-PAES obtained slightly better results than HM, but in this case it also has a lower variability. It can be clearly seen that for **g04**, IS-PAES had a better performance with respect to all the statistical measures evaluated. The same applies to **g05** in which HM was not able to find any feasible solutions (the best result found by IS-PAES was very close to the global optimum). For **g06**, again IS-PAES found better values than HM (IS-PAES practically converges to the optimum in all cases). For **g07** both the best and the mean values produced by IS-PAES were better than those produced by HM, but the difference between the worst and best

Table 1. Comparison of the results for the test functions from [7]. Our approach is called IS-PAES and the homomorphous maps approach [6] is denoted by HM. N.A. = Not Available.

TF	OPTIMAL	BEST RESULT		MEAN RESULT		WORST RESULT	
		IS-PAES	HM	IS-PAES	HM	IS-PAES	HM
g01	-15.0	-14.995	-14.7864	-14.909	-14.7082	-12.4476	-14.6154
g02	-0.803619	-0.8035376	-0.79953	-0.798789	-0.79671	-0.7855539	-0.79199
g03	-1.0	-1.00050019	-0.9997	-1.00049986	-0.9989	-1.00049952	-0.9978
g04	-30665.539	-30665.539	-30664.5	-30665.539	-30655.3	-30665.539	-30645.9
g05	5126.498	5126.99795	N.A.	5210.22628	N.A.	5497.40441	N.A.
g06	-6961.814	-6961.81388	-6952.1	-6961.81387	-6342.6	-6961.81385	-5473.9
g07	24.306	24.3410221	24.620	24.7051034	24.826	25.9449662	25.069
g08	-0.095825	-0.09582504	-0.0958250	-0.09582504	-0.0891568	-0.09582504	-0.0291438
g09	680.630	680.638363	680.91	680.675002	681.16	680.727904	683.18
g10	7049.331	7055.11415	7147.9	7681.59187	8163.6	9264.35787	9659.3
g11	0.750	0.75002984	0.75	0.74992803	0.75	0.74990001	0.75

result is slightly lower for HM. For **g08** the best result found by the two approaches is the optimum of the problem, but IS-PAES found this same solution in all the runs performed, whereas HM presented a much higher variability of results. In **g09**, IS-PAES had a better performance than HM with respect to all the statistical measures adopted. For **g10** none of the two approaches converged to the optimum, but IS-PAES was much closer to the optimum and presented better statistical measures than HM. Finally, for **g11**, HM presented slightly better results than IS-PAES, but the difference is practically negligible.

Summarizing, we can see that IS-PAES either outperformed or was very close to the results produced by HM even when it only performed 25% of the fitness functions evaluations of HM. IS-PAES was also able to approach the global optimum of **g05**, for which HM did not find any feasible solutions.

7 Conclusions and Future Work

We have presented a new constraint-handling approach that combines multiobjective optimization concepts with an efficient reduction mechanism of the search space and a secondary population. We have shown how our approach overcomes the scalability problem of the original PAES (which was proposed exclusively for multiobjective optimization) from which it was derived, and we also showed that the approach is highly competitive with respect to a constraint-handling approach that is representative of the state-of-the-art in the area.

The proposed approach illustrates the usefulness of multiobjective optimization concepts to handle constraints in evolutionary algorithms used for single-objective optimization. Note however, that this mechanism can also be used for multiobjective optimization and that is in fact part of our future work.

Another aspect that we want to explore in the future is the elimination of all of the parameters of our approach using online or self-adaptation. This task, however, requires

a careful analysis of the algorithm because any online or self-adaptation mechanism may interfere with the mechanism used by the approach to reduce the search space.

Acknowledgments. The first and second authors acknowledge support from CONACyT project No. P-40721-Y. The third author acknowledges support from NSF-CONACyT project No. 32999-A.

References

1. Thomas Bäck. *Evolutionary Algorithms in Theory and Practice*. Oxford University Press, New York, 1996.
2. Eduardo Camponogara and Sarosh N. Talukdar. A Genetic Algorithm for Constrained and Multiobjective Optimization. In Jarmo T. Alander, editor, *3rd Nordic Workshop on Genetic Algorithms and Their Applications (3NWGA)*, pages 49–62, Vaasa, Finland, August 1997. University of Vaasa.
3. Carlos A. Coello Coello, David A. Van Veldhuizen, and Gary B. Lamont. *Evolutionary Algorithms for Solving Multi-Objective Problems*. Kluwer Academic Publishers, New York, May 2002. ISBN 0-3064-6762-3.
4. David E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Publishing Company, Reading, Massachusetts, 1989.
5. Joshua D. Knowles and David W. Corne. Approximating the Nondominated Front Using the Pareto Archived Evolution Strategy. *Evolutionary Computation*, 8(2):149–172, 2000.
6. Slawomir Koziel and Zbigniew Michalewicz. Evolutionary Algorithms, Homomorphous Mappings, and Constrained Parameter Optimization. *Evolutionary Computation*, 7(1):19–44, 1999.
7. Zbigniew Michalewicz and Marc Schoenauer. Evolutionary Algorithms for Constrained Parameter Optimization Problems. *Evolutionary Computation*, 4(1):1–32, 1996.
8. I. C. Parmee and G. Purchase. The development of a directed genetic search technique for heavily constrained design spaces. In I. C. Parmee, editor, *Adaptive Computing in Engineering Design and Control-'94*, pages 97–102, Plymouth, UK, 1994. University of Plymouth, University of Plymouth.
9. Tapabrata Ray, Tai Kang, and Seow Kian Chye. An Evolutionary Algorithm for Constrained Optimization. In Darrell Whitley et al., editor, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2000)*, pages 771–777, San Francisco, California, 2000. Morgan Kaufmann.
10. Jon T. Richardson, Mark R. Palmer, Gunar Liepins, and Mike Hilliard. Some Guidelines for Genetic Algorithms with Penalty Functions. In J. David Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms (ICGA-89)*, pages 191–197, San Mateo, California, June 1989. George Mason University, Morgan Kaufmann Publishers.
11. T.P. Runarsson and X. Yao. Stochastic Ranking for Constrained Evolutionary Optimization. *IEEE Transactions on Evolutionary Computation*, 4(3):284–294, September 2000.
12. Alice E. Smith and David W. Coit. Constraint Handling Techniques—Penalty Functions. In Thomas Bäck, David B. Fogel, and Zbigniew Michalewicz, editors, *Handbook of Evolutionary Computation*, chapter C 5.2. Oxford University Press and Institute of Physics Publishing, 1997.
13. Patrick D. Surry and Nicholas J. Radcliffe. The COMOGA Method: Constrained Optimisation by Multiobjective Genetic Algorithms. *Control and Cybernetics*, 26(3):391–412, 1997.

Evolution Strategies with Exclusion-Based Selection Operators and a Fourier Series Auxiliary Function

Kwong-Sak Leung and Yong Liang

Department of Computer Science & Engineering,
The Chinese University of Hong Kong, Shatin, N.T., Hong Kong
{ksleung, yliang}@cse.cuhk.edu.hk

Abstract. To improve the efficiency of the currently known evolutionary algorithms, we have proposed two complementary efficiency speed-up strategies in our previous research work respectively: the exclusion-based selection operators and the Fourier series auxiliary function. In this paper, we combine these two strategies together to search the global optima in parallel, one for optima in large attraction basins and the other for optima in very narrow attraction basins respectively. They can compliment each other to improve evolutionary algorithms (EAs) on efficiency and safety. In a case study, the two strategies have been incorporated into evolution strategies (ES), yielding a new type of accelerated exclusion and Fourier series auxiliary function ES: the EFES. The EFES is experimentally tested with a test suite containing 10 complex multimodal function optimization problems and compared against the standard ES (SES). The experiments all demonstrate that the EFES consistently and significantly outperforms the SES in efficiency and solution quality.

1 Introduction

Evolutionary algorithms (EAs) are global search procedures based on the evolution of a set of solutions viewed as a population of interacting individuals. They have been successfully used for optimization problems. But for solving large scale and complex optimization problems, EAs have not demonstrated themselves to be very efficient [4] [5]. We believe the main factor which causes low efficiency of the current EAs is the convergence towards undesired attractors. This phenomenon occurs when the objective function has some local optima with large attraction basins or its global optimum is located in a small attraction basin in a minimization case. The relationship between the convergence to a global minimum and the geometry (landscape) of the difficult function problems is very important. If the population of EAs gets trapped into suboptimal states, which locate in comparative large attraction basins, then it is difficult for the variation operators to produce an offspring which outperforms its parents. In the second case, if global optima are located in relatively small attraction basins, and the individuals of EAs have not found these basins yet, the probability of the variation operators to produce offspring which locate in these small attraction basins

is quite low. In both cases, the stochastic mechanism of EAs yields unavoidable resampling, which increases the algorithm's complexity and decelerates the search efficiency.

To overcome these two limitations that cause low efficiency of the currently known EAs, we have proposed two complementary efficiency speed-up strategies in our previous research work respectively [6]: the exclusion-based selection operators and the Fourier series auxiliary function. The exclusion-based selection operators could efficiently prevent the individuals of EAs from getting into the attraction basins of local optima. While the Fourier series auxiliary function could guide an algorithm to search for optima with small attraction basins efficiently. Moreover, this strategy can compensate the deficiency of the exclusion-based selection operators on the algorithm's safety, i.e. the avoidance of excluding a global optimum contained in a narrow attraction basin. In this paper, we developed a new algorithm, the EFES, which incorporate the exclusion and Fourier series auxiliary function into the evolution strategies (ES) [2]. We expect that EFES will have the advantages of both two strategies — efficiency and safety.

This paper is organized as follows: we will explain the two novel efficiency speed-up strategies for EAs implementation in Sections 2 and 3 respectively. Particularly, a set of “exclusion-based” selection operators is proposed in Section 2 to simulate the “survival of the fittest” principle more powerfully and a Fourier series auxiliary function is introduced in Section 3. In Section 4, we will demonstrate how to embed the exclusion-based selection operators and the Fourier series auxiliary function into ES to generate EFES. In Section 5, the EFES is experimentally examined, analyzed and compared with a set of typical, multi-modal function optimization problems. The last section is the conclusion.

2 Exclusion-Based Selection Operators

This section defines and explores a somewhat different selection mechanism — the exclusion-based selection operators.

Any EAs solve an optimization problem, say,

$$(\mathbf{P}) \quad \min\{f(x) : x \in \Omega\},$$

where $f : \Omega \subset R^n \rightarrow R$ is a function. For simplicity, consider the problem $(\mathbf{P})$ with the domain Ω specified by $\Omega = [u_1, v_1] * [u_2, v_2] * \cdots * [u_n, v_n]$. The new scheme is based on the cellular partition methodology. Given an integer d , let $h_i = \frac{v_i - u_i}{d}$, we define

$$\sigma(j_1, j_2, \dots, j_n) =$$

$$\{x = (x_1, x_2, \dots, x_n) \in \Omega : (j_i - 1) \times h_i \leq x_i - u_i \leq j_i \times h_i, 1 \leq j_i \leq d\}$$

and let the subregion $\sigma(j_1, j_2, \dots, j_n)$, be a cell, and the cell collection $\Gamma(d)$ is called a cellular partition of Ω . The vector representing the center of the cell is defined by

$$m(\sigma(j_1, \dots, j_n)) = ((u_1 + (j_1 + \frac{1}{2}) \times h_1), \dots, (u_n + (j_n + \frac{1}{2}) \times h_n)).$$

Let us first formalize the exclusion operators with reference to the cellular partition $\Gamma(d)$ of Ω .

Definition 1: An exclusion operator is a computationally verifiable test for nonexistence of solution to problem $(\mathbf{P})$ which can be implemented on every cell of $\Gamma(d)$.

For example, assume that f satisfies the Lipschitz condition: $|f(x) - f(y)| \leq \alpha \|x - y\|$. Then, for any $\sigma \in \Gamma(d)$, there is a global minimizer x^* in σ only if $|f(m(\sigma)) - f(x^*)| \leq \alpha \|m(\sigma) - x^*\| \leq \frac{\alpha}{2} \omega(\sigma)$, where $\omega(\sigma)$ is the mesh size of $\Gamma(d)$, defined by

$$\omega(\sigma) = \max_{1 \leq i \leq n} \left\{ \frac{v_i - u_i}{d} \right\}$$

This implies that $f(m(\sigma)) - \frac{\alpha}{2} \omega(\sigma) \leq f(x^*)$ is a necessary condition for the existence of the global optimum of $(\mathbf{P})$. Thus $f(m(\sigma)) - \frac{\alpha}{2} \omega(\sigma) > f(x^*)$ gives a nonexistence test for the solution. This test is an exclusion operator because it can be computationally verified in each cells of $\Gamma(d)$.

By Definition 1, any exclusion operator can serve to computationally test the nonexistence of the solution of $(\mathbf{P})$ in any given cells of $\Gamma(d)$. It therefore can be used to check if a given cell in $\Gamma(d)$ is prospective or not as a global optimum (or, as a portion of a attraction basin of a global optimum) of $(\mathbf{P})$. Accordingly, the non-global optimum, or, more loosely, less prospective cells can be identified and can be deleted from further consideration. This is the key mechanism that is adopted in the present work to accelerate EAs. Hereafter any selection mechanism based on such exclusion principle will be called an *exclusion-based selection operator*.

Let σ be an arbitrary cell in $\Gamma(d)$ with its center $m(\sigma)$ and the mesh size $\omega(\sigma)$. The following provides us with a series of exclusion operators for minimization problems. Please note that these operators (tests) can be equally applied to maximization problems by reversing the inequality signs.

Example 1: Concavity Test Operator

Suppose f is continuous, then a necessary condition for a point x to be a global minimizer is that f is convex at the neighborhood of x . Therefore, when d is sufficiently large, the following tests $(\mathbf{E}_1)$ and $(\mathbf{E}_2)$ are exclusion operators.

$(\mathbf{E}_1)$: There is a direction $e \in R^n$ such that

$$f(m(\sigma)) > \frac{1}{2}[f(x^+) + f(x^-)]$$

where $x^- = x - \eta e$, $x^+ = x + \eta e$ and $0 < \eta \leq \frac{1}{2}\omega(\sigma)$

$(\mathbf{E}_2)$: There is a direction $e \in R^n$ such that $\max\{f(x^+), f(x^-)\} > f_{best}$ and

$$[f(x^+) - f(m(\sigma))][f(x^-) - f(m(\sigma))] < 0$$

where x^+ and x^- are the same as in $(\mathbf{E}_1)$, and f_{best} is the current known best fitness value.

The tests $(\mathbf{E}_1)$ and $(\mathbf{E}_2)$ can immediately follow from the observations that every $m(\sigma)$ is in the interior of Ω , the $(\mathbf{E}_1)$ features the concave property of f

on cell σ , and $(\mathbf{E}_2)$ characterizes the convex and concave overlapped property in which no global optimum of f exists.

Example 2: Lipschitz Test Operator

Let $\mathcal{L}(f, \alpha)$ denote the family of all continuous functions that satisfies the Lipschitz conditions: $|f(x) - f(y)| \leq \alpha(\Omega) \|x - y\|, x, y \in \Omega \in \Gamma(d)$. Then the following tests $(\mathbf{E}_3)$ and $(\mathbf{E}_4)$ are exclusion operators.

$$(\mathbf{E}_3): f_{best} < f(m(\sigma)) - \frac{\alpha(\sigma)}{2} \omega(\sigma), \text{ if } f \in \mathcal{L}(f, \alpha)$$

$$(\mathbf{E}_4): f_{best} < f(m(\sigma)) - \frac{\alpha(\sigma)}{8} \omega^2(\sigma), \text{ if } f' \in \mathcal{L}(f, \alpha)$$

where f_{best} is again the “best-so-far” fitness value, and f' is the derivative of f .

Example 3: Formal Series Test Operator

Let $\mathfrak{F}$ be the class of all functions that can be expressed as a finite number of superpositions of formal series and their absolute values $A(f)$ of f is defined by

$$A(f) = A(f^{(1)}) + \sum_{j=2}^k A(f^{(j)})(x).$$

For any $f \in \mathfrak{F}$ and $g \succ A(f)$, it is known [8] that the following basic inequality holds:

$$|f(x) - f(y)| \leq A(g)(|y| + |x - y|) - A(g)(|y|), \forall x, y \in R^n$$

This implies, similar to test $(\mathbf{E}_3)$, that the following test $(\mathbf{E}_5)$ is an exclusion operator.

$(\mathbf{E}_5)$: There is a formal series $g \succ A(f)$ such that

$$f_{best} \leq f(m(\sigma)) - [A(g)(|m(\sigma)| + \frac{1}{2} \omega(\sigma)) - A(g)(|m(\sigma)|)]$$

Various other exclusion operators may also be constructed by virtue of other delicate mathematical tools such as interval arithmetic [1] and the cell mapping methods [7] [8].

Remark 1:

(i) The exclusion-based selection is the main component and contributor to the accelerated evolutionary algorithms (the fast-EAs). Aiming at suppressing the resampling effect of EAs, such type of selection provides a smart guidance to EAs search towards promising areas through eliminating non-promising areas. Different in principle from the conventional selection operators, the construction of which is based on sufficient condition for the existence of solution to $(\mathbf{P})$, the exclusion-based selection operators can be constructed based on necessary condition. This presents a general methodology of constructing exclusion operators. The above listed tests $(\mathbf{E}_1)$ - $(\mathbf{E}_5)$ show such examples of the construction.

(ii) The application of exclusion-based operator has another advantage: It can very naturally incorporate some useful properties of the objective function into the EAs search, providing other acceleration means whenever possible. Indeed, Examples 1-3 all have taken advantage of properties of f in certain ways (say, continuity in Examples 1, Lipschitz conditions in Example 2 and analyticity in

Example 3). As a general rule, the more exclusive properties of f are utilized, the more accurate a test could be deduced (For example, $(\mathbf{E}_3)$ is more accurate than $(\mathbf{E}_4)$ when the Lipschitz condition was applied for the derivative f' instead of for f). These different tests deduced from different properties by no means have to be applied for a problem in the same time. They can, for instance, be applied either independently, or with several others together, or totally simultaneously, depending on the available information on f that can actually be made use of.

3 A Fourier Series Auxiliary Function

The exclusion-based selection operators are the efficient accelerating operators based on the interval arithmetic and the cell mapping methods. However, if the function optimization problem is very complex, say there are many sharp basins, the optima may be excluded by mistake using the above operators. Meanwhile, searching an optimum with a small attraction basin is difficult for standard evolutionary algorithms. To solve the above problem, a Fourier series auxiliary function is introduced in this section.

As we know, if $f(x)$ is continuous or merely piecewise continuous (continuous except for finitely many finite jumps in the interval of integration), then the Fourier series of $f(x)$ is convergent. Its sum is $f(x)$, except at a point x_0 at which $f(x)$ is discontinuous and the sum of the series is the average of the left- and right-hand limits of $f(x)$ at x_0 [3]. We define the finite partial sum of the Fourier series called $F_\ell^k(x)$ (e.g. for one dimension),

$$F_\ell^k(x) = \sum_{n=\ell}^k \left(a_n \cos \frac{2n\pi}{u-v} x + b_n \sin \frac{2n\pi}{u-v} x \right), x \in [u, v].$$

The infinite Fourier series $F_1^\infty(x)$ converges to $f(x)$ at any point, but the convergent speed of the finite partial sum $F_1^\ell(x)$ ($\ell < \infty$) is different at each point. For numerical function optimization, the finite partial sum $F_1^\ell(x^*)$ ($\ell < \infty$) converges to $f(x^*)$ much slower for optimum x^* with small attraction basin in $f(x)$ than that of optimum x^* with a large attraction basin. This indicates the partial sum $|F_\ell^\infty(x^*)| = |f(x^*) - F_1^\ell(x^*)|$ at x^* with a small attraction basin is larger than at x^* with a large attraction basin. Because when an integer $k \rightarrow \infty$, the coefficients of the k^{th} Fourier series term a_k and $b_k \rightarrow 0$, the infinite partial sum $F_k^\infty(x) \rightarrow 0$. So we consider the finite partial sum $F_\ell^k(x)$ ($\ell < k$) instead of $F_\ell^\infty(x)$. The proposition $|F_\ell^\infty(x^*)| > |F_\ell^\infty(x)|$ equals to $|F_\ell^k(x^*)| > |F_\ell^k(x)|$ ($\ell < k$) when x^* locates in a small attraction basin.

The features of the finite partial sum $F_\ell^k(x)$ ($\ell < k$) include enlarging small attraction basins, and smoothing large attraction basins of $f(x)$ as shown in Fig.1. We have designed three strategies: the region partition strategy, the one-element strategy and the double integral strategy to construct auxiliary function $g(x)$. The first strategy is designed for representing all optima with small attraction basins by the $F_\ell^k(x)$ ($\ell < k$) with a small number of terms. The second one is for significantly reducing the computational complexity. The last one is

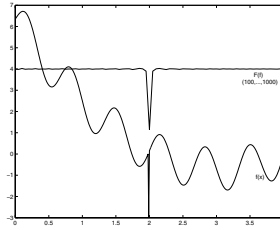


Fig. 1. A schematic illustration for the feature of the Fourier finite partial sum F_ℓ^k ($\ell = 100$, $k = 1000$).

for expanding the dummy optima out of the original feasible region, and keeping the original position of the optimum unchanged (Fig.2). Consequently, we could construct the auxiliary function $g(x)$ using the finite partial sum of the one element of the Fourier trigonometric system

$$g(x) = \sum_{m=\ell}^k a_m \prod_{i=1}^n \cos mx_i,$$

where n is dimensions, $\ell = 100$ and $k = 200$;

$$a_m = \frac{1}{u - 2v} \int_u^{2v} f(x) \cos \frac{2m\pi}{2v - u} x dx;$$

to locate the optima of the origin function $f(x)$.

Since the auxiliary function $g(x)$ can enlarge the small attraction basins of the optima and flatten the large attraction basins, the $g(x)$ can guide an algorithm to search the optima with small attraction basins more efficient, and these optima are difficult to find in the original objective function by EAs. Furthermore, this strategy runs in parallel with first strategy and compensates the deficiency of the exclusion-based selection operators on the algorithm's risk of missing optima with many sharp attraction basins.

4 EFES: The Evolution Strategies with Exclusion-Based Selection Operators and a Fourier Series Auxiliary Function

In this section, we demonstrate how all the strategies developed in the previous sections could be embedded into the known evolutionary algorithms, to yield new versions of the algorithms. We will particularly take evolution strategies (ES) as an example. Consequently, a new type of ES the EFES will be developed. The incorporation of the developed strategies with other known evolutionary algorithms might be straightforward, and could be done similarly as that in the example presented below.

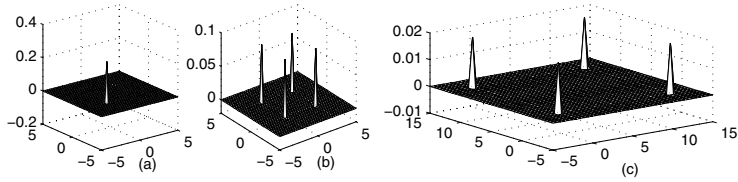


Fig. 2. A graphical illustration of the $g(x)$. (a) shows the complete Fourier system representation; (b) shows one element Fourier representation with parameters determined by $[u, v]$; (c) shows one element Fourier representation with parameters are determined by $[u, 2v]$.

The EFES Algorithm Is Given as Follows:

I. Initialization Step

- I.1.** Set $k = 0$, $\Omega^{(0)} = \Omega$, $n(0) = 1$ and $f_{best}^0 = 10^8$; set the pre-determined number of cellular partition d , and stopping criteria ε_1 and ε_2 , where ε_1 is the solution precision requirement and ε_2 is the space discretization precision tolerance.
- I.2.** Initialize all the ES parameters including:
 - N — the population size;
 - M — the maximum number of ES evolution.

II. Iteration Step (Epoch)

II.1. ES Search in the auxiliary function $g(x)$:

- (a) randomly select N individuals from $\Omega^{(0)}$ to form an initial population;
- (b) perform M steps of ES search, yielding the currently best minimum $g(x^*)$ in the region $\Omega^{(0)}$;
- (c) if generation = 1, we put these points into population of ES for $f(x)$; if generation > 1, we will compare the fitness of $f(x)$ at these points with the current optimum of $f(x)$, and only put the points which values of $f(x)$ smaller than the current optimum of $f(x)$.

The two steps **II.2** and **II.3** below will use the cellular partition method, assuming $\Omega^{(k)}$ consists of $n(k)$ subregions, say, $\Omega^{(k)} = \Omega_1^{(k)} \cup \Omega_2^{(k)} \cup \dots \cup \Omega_{n(k)}^{(k)}$.

For each subregion $\Omega_i^{(k)}$, do the following:

II.2. ES Search in the objective function $f(x)$:

- (a) use the fixed points by step II.1 and some random selected points from $\Omega_i^{(k)}$ to form initial population;
- (b) perform M steps of ES search, yielding the currently best minimum $f_i^{(k)}$ in the subregion $\Omega_i^{(k)}$;
- (c) let $f_{best}^{(k)} := \min\{f_i^{(k)}, f_{best}^{(k-1)}\}$.

II.3. Exclusion-based Selection:

With the “best-so-far” fitness value $f_{best}^{(k)}$ guided, eliminate the “less prospective” individuals (cells) from $\Omega_i^{(k)}$ by employing appropriate exclusion operator(s) and a specific exclusion scheme. The remaining cells are denoted by $\Pi_i^{(k)}$.

II.4. Space Shrinking:

- (a) generate a cell $\Omega_i^{(k+1)}$ such that $\Pi_i^{(k)} \subset \Omega_i^{(k+1)}$ and $\omega(\Omega_i^{(k+1)}) < \omega(\Omega_i^{(k)})$ whenever possible; in this case, set $\Omega_{i1}^{(k+1)} = \Omega_i^{(k+1)}$ and $\Omega_{i2}^{(k+1)} = \emptyset$;
 (b) bisect $\Pi_i^{(k)}$ and construct two large cells $\Omega_{i1}^{(k+1)}$ and $\Omega_{i2}^{(k+1)}$ such that

$$\Pi_i^{(k)} \subset \Omega_{i1}^{(k+1)} \cup \Omega_{i2}^{(k+1)} \text{ and } \max\{\omega(\Omega_{i1}^{(k+1)}), \omega(\Omega_{i2}^{(k+1)})\} < \omega(\Pi_i^{(k)}).$$

III. Termination Test Step

If $\omega(\Omega^{(k)}) \leq \varepsilon_2$ and $|f^{(k)} - f^{(k-1)}| \leq \varepsilon_1$ hold for three consecutive iteration steps, then stop; Otherwise, go to step II with $k := k + 1$, and

$$\Omega^{(k+1)} = \Omega_{i1}^{(k+1)} \cup \Omega_{i2}^{(k+1)} \dots \cup \Omega_{n(k)}^{(k)}.$$

Detailed remarks on each step of the above algorithm are given in [6].

The iteration step (step II) is the core of the algorithm. Step II.1 performs the ES for M generations to search in the auxiliary function $g(x)$, the best individuals are potential global optima, which are difficult to find in $f(x)$. Adopting the comparison criterion could efficiently eliminate interference caused by other kinds of points (like discontinuous point). In order to ensure the safety of the algorithm, we do not apply the exclusion-based selection operators in the search process of $g(x)$.

Table 1. The test suite used in our experiments.

$f_1 = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4, n = 2, x \in [-5, 5];$
$f_2 = x_2^2 + 2x_1^2 - 0.3\cos(3\pi x_2) - 0.4\cos(3\pi x_1) + 0.7, n = 2, x \in [-5.12, 5.12];$
$f_3 = \sum_{i=1}^{n/4} (x_{4i-3} + 10x_{4i-2})^2 + 5(x_{4i-1} - x_{4i})^2 + (x_{4i-2} - 2x_{4i-1})^2 + 10(x_{4i-3} - x_{4i})^4, n = 40, x \in [-100, 100];$
$f_4 = \sum_{i=1}^{n/4} 3[\exp(x_{4i-3} - x_{4i-2})^2 + 100(x_{4i-2} - x_{4i-1})^6 + [\tan(x_{4i-1} - x_{4i})^4 + x_{4i-3}^8], n = 40, x \in [-4, 4];$
$f_5 = \sum_{i=1}^{n/4} \{100[x_{4i-3}^2 - x_{4i-2}]^2 + (x_{4i-3} - 1)^2 + 90(x_{4i-1}^2 - x_{4i})^2 + 10.1[(x_{4i-2} - 1)^2 + (x_{4i} - 1)^2] + 19.8(x_{4i-2} - 1)(x_{4i} - 1)\}, n = 40, x \in [-50, 50];$
$f_6 = \frac{1}{4000} \sum_{i=1}^n x_i^2 - \prod_{i=1}^n \cos(\frac{x_i}{\sqrt{i}}) + 1, n = 40, x \in [-600, 600];$

5 Simulations and Comparisons

We will experimentally evaluate the performance of the EFES, and compare it with the standard evolution strategies (SES). The EFES was implemented with $k = 10, N = 1000$ and $\varepsilon_1 = \varepsilon_2 = 10^{-8}$. The maximal number M of ES evolution was taken uniformly to be 500 on $f(x)$ and $g(x)$ respectively, called a epoch. All these parameters and schemes were kept invariant unless noted otherwise. For fairness of comparison, we also implemented the SES with the same parameter settings and the same initial population. The maximum number of SES evolution

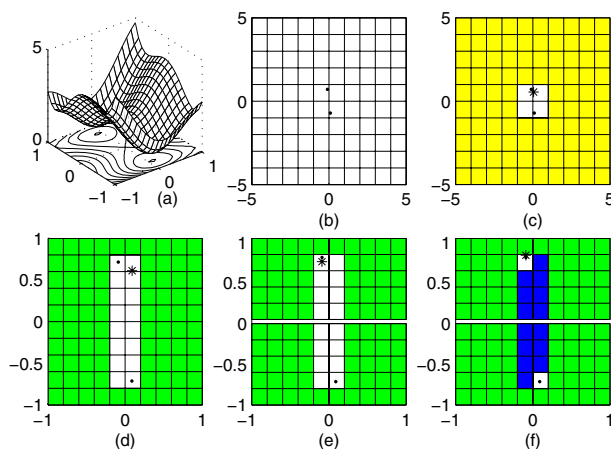


Fig. 3. The search space (the white cells) shrinking process when the EFES applied to f_1 . (b) shows the locations of the global optima of f_1 (denoted by “•”); (c) and (d) show the cells excluded by the algorithm in epochs 1 and 2 (current best search point is denoted by “*”). In epoch 3, the search space is forked into two subregions in (e) because the two global optima all lie in the husk layer of the search space. After forking, one more epoch yields the two global optima with precision 10^{-8} .

is 10^5 . All experiments were run for ten times with random initial populations, and the averages of the ten runs were taken as the final result.

The test suite used in our experiments include those minimization problems listed in Table 1. The suite mainly contains some representative, complex, multimodal functions with many local optima and being highly nonseparable in features. Our experiments were divided into two groups with different purposes. We report the results of each group below.

Explanatory Experiments: This group of experiments aims to exhibit the evolution processes of the EFES in detail. To clearly demonstrate the running process of the exclusion-based selection operators, this simulation first studies the ES with exclusion-based selection operators and without the Fourier series auxiliary function to solve the problem f_1 . Fig.3-(a) shows the function f_1 that has only two global minimizers. The evolution details (particularly, the search space shrinking details) of the EFES when applied to minimization of this function is presented in Figs.3 (b)-(f), which demonstrate clearly how the remaining cells are accumulated around the currently acceptable search point (denoted by “*”) for the global minimum, and how local minima are successively excluded. This demonstrates the common features of the EFES. The experiments also demonstrate that the number of subregions, $n(k)$, contained in search space $\Omega^{(k)}$ in each step are uniformly bounded. These bounds are seen to be very small in each case, but vary with the problems under consideration. Particularly we observed in the experiments that the bounds $n(k)$ are generally related (ac-

tually, proportional) to the number of the global optima of the function to be optimized.

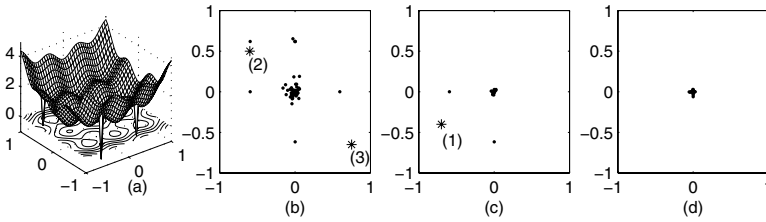


Fig. 4. A schematic illustration that the EFES to search on f_2 . (a) shows random version of f_2 . (b) and (c) show the results after a first and second epoches respectively. The points (1), (2) and (3) show the optima $(-1, 10^{-5})$, $(-0.5, 10^{-3})$ and $(-0.5, 10^{-3})$, respectively.

We designed the test function based on benchmark multimodal functions f_2 to demonstrate the feature of the auxiliary function $g(x)$. Three optima with small attraction basins are generated randomly in the feasible solution space of $f(x)$ and they have the following properties respectively: $(-0.5, 10^{-3})$, $(-0.5, 10^{-3})$ and $(-1, 10^{-5})$, representing the value of the optima and the width of its attraction basin in the bracket, respectively. The locations of these optima are decided in random.

Fig.4-(a) shows random version of f_2 . Figs.4-(b) and (c) show the results of the EFES within the first and second epoches respectively, the points * are obtained from the EFES in the search in $g(x)$. Fig.4-(b) demonstrates that the EFES can find the two optima $(-0.5, 10^{-3})$ in the first epoch(1000 generations), however the optimum $(-1, 10^{-5})$ is not represented in $g(x)$ at this time. In the second epoch, after bisecting the whole space between these two optima $(-0.5, 10^{-3})$, the optimum $(-1, 10^{-5})$ is represented by $g(x)$ and identified by the EFES(c.f. Fig.4-(c)). These results confirm that the EFES can find these three optima with small attraction basins in $g(x)$. Fig.4-(d) shows SES is converged to the optimum $f_1(0, 0) = 0$ after 10000 generations, which is the global optimum of the original version of function f_2 . However, this point is not the global optimum of random version of f_2 .

Comparisons: To assess the effectiveness and efficiency of the EFES, its performance is compared with the standard ES (SES). We define that $f_7 - f_{10}$ are the random versions of $f_3 - f_6$. Functions $f_7 - f_{10}$ have one optimum with a small attraction basin $(-1, 10^{-5})$, representing the value of the optimum and the width of its attraction basin in the bracket, respectively. The location of the optimum are decided in random. The comparisons are made in terms of the solution quality and computational efficiency and on the basis of applications of the algorithms to the test functions $f_3 - f_{10}$ in the test suite. As each algorithm

has its associated overhead, a time measurement was taken as a fair indication of how effectively and efficiently each algorithm could solve the problems. The solution quality and computational efficiency are therefore respectively measured by the solution precision and the fitness attained by each algorithm within an equal period of fixed time. Unless mentioned otherwise, the time is measured in minutes as measured on the computer.

Table 2 and Fig.5 present the solution quality comparison results in terms of $f_{best}^{(t)}$ when the EFES and SES are applied to the test functions $f_3 - f_{10}$. We can observe that while the EFES consistently converges to the global optima for all test functions, SES is unable to find the global solutions for $f_3 - f_6$ and their random versions, $f_7 - f_{10}$. On the other hand, Table 2 and Fig.5 also show that the EFES can always locate the global optimum with higher solution precision. That is, the EFES outperforms SES in solution effectiveness.

The computational efficiency comparison results are shown in Fig.5. It is clear from these figures that the EFES significantly outperform the SES for all test functions. In addition, we could see from Figs.5 (e)-(h) that the efficiency increases because the auxiliary function $g(x)$ efficiently guide the EFES to find the global optimum with a small attraction basin. Even with such efficiency speed-up, the guaranteed monotonic convergence of the EFES is still clearly observed in all these experiments. All these comparisons show the superior performance of the EFES in efficacy and efficiency.

Table 2. The results of the EFES and SES when applied to test functions

Function	Epochs	The running time (minute)		The solution precision attained	
		SES	EFES	SES	EFES
f_3	5	17.2	17.2	10^{-4}	10^{-8}
f_4	5	19.0	19.0	10^{-3}	10^{-8}
f_5	5	21.6	21.6	10^{-3}	10^{-8}
f_6	5	16.8	16.8	10^{-3}	10^{-8}
f_7	3	17.2	7.3	10^{-4}	10^{-8}
f_8	2	19.0	5.8	10^{-3}	10^{-8}
f_9	3	21.6	15.8	10^{-3}	10^{-8}
f_{10}	3	16.8	9.1	10^{-3}	10^{-8}

6 Conclusion

In this paper, we have developed a new evolutionary algorithm— EFES, which incorporates two strategies, exclusion-based selection operators and the Fourier series auxiliary function into ES, to solve global optimization problems.

The EFES has been experimentally tested with a difficult test suite consisted of two groups of complex multimodal function optimization examples. The performance of the EFES is compared against the standard evolution strategies

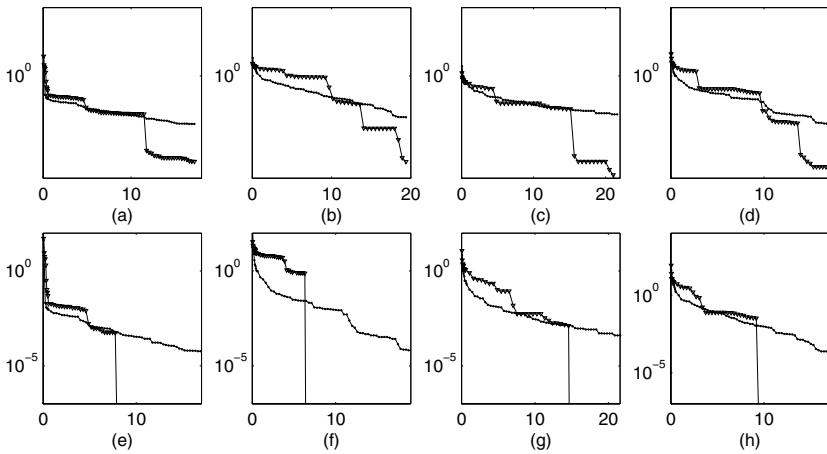


Fig. 5. The solution quality and computational efficiency comparisons of the EFES and SES when applied to problems $f_3(a)$, $f_4(b)$, $f_5(c)$, $f_6(d)$, $f_7(e)$, $f_8(f)$, $f_9(g)$, $f_{10}(h)$, where the abscissa is the time (minutes), and the ordinate is the solution precision $|f_{best}^{(t)} - f^*|$ (absolute error) (Keys: v – EFES, . – SES).

(SES). All experiments have demonstrated that the EFES consistently and significantly outperforms the SES in efficiency and solution quality, particularly towards problems whose global optima are located in small attraction basins. Since the Fourier series auxiliary function could be used on discontinuous function optimization problems and prevent accidental deletion of the global optima with very narrow attraction basins by the exclusion-based selection operators, EFES has wider application, both for continuous and discontinuous problems.

Acknowledgment. This research was partially supported by RGC Earmarked Grant 4212/01E of Hong Kong SAR and RGC Research Grant Direct Allocation of the Chinese University of Hong Kong.

References

1. E.Hansen, *Global optimization Using Interval Analysis*, Marcel Dekker, INC, 1992
2. H.P.Schwefel, *Evolution and Optimum Seeking*, Chichester, UK: John Wiley, 1995
3. R.E.Edwards, Fourier Series, Second Edition, *Springer-Verlag New York Heidelberg Berlin*, 1999
4. Thomas Bäck, *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms*, Oxford, New York: Oxford University Press, 1996.
5. Y.Xin, *Evolutionary computation: theory and applications*, Singapore; New Jersey: World Scientific, 1999
6. Y.Liang & K.S.Leung, Fast-GA: A Genetic Algorithm with Exclusion-based Selections, *Proceedings of 2001 WSES International Conference on: Evolutionary Computations (EC'01)*, pp. 638(1)–638(6), Spain.

7. Z.B.Xu, J.S.Zhang & W.Wang, A cell exclusion algorithm for finding all the solutions of a nonlinear system of equation, *Applied Mathematics and Computation*, 80, 1996, pp. 181–208
8. Z.B.Xu, J.S.Zhang, & Y.W.Leung, A general CDC formulation for specializing the cell exclusion algorithms of finding all zeros of vector functions, *Applied Mathematics and Computation*, 86(2 & 3), 1997, pp. 235–259

Ruin and Recreate Principle Based Approach for the Quadratic Assignment Problem

Alfonas Misevicius

Kaunas University of Technology, Department of Practical Informatics,
Studentu St. 50-400a, LT-3031 Kaunas, Lithuania
misevi@soften.ktu.lt

Abstract. In this paper, we propose an algorithm based on so-called ruin and recreate (R&R) principle. The R&R approach is conceptual simple but at the same time powerful meta-heuristic for combinatorial optimization problems. The main components of this method are a ruin (mutation) procedure and a recreate (improvement) procedure. We have applied the R&R principle based algorithm for a well-known combinatorial optimization problem, the quadratic assignment problem (QAP). We tested this algorithm on a number of instances from the library of the QAP instances – QAPLIB. The results obtained from the experiments show that the proposed approach appears to be significantly superior to a "pure" tabu search on real-life and real-life like QAP instances.

1 Introduction

The quadratic assignment problem (QAP) is formulated as follows. Let two matrices $A = (a_{ij})_{n \times n}$ and $B = (b_{kl})_{n \times n}$ and the set Π of permutations of the integers from 1 to n be given. Find a permutation $\pi = (\pi(1), \pi(2), \dots, \pi(n)) \in \Pi$ that minimizes

$$z(\pi) = \sum_{i=1}^n \sum_{j=1}^n a_{ij} b_{\pi(i)\pi(j)} \cdot \quad (1)$$

The context in which Koopmans and Beckmann [1] first formulated this problem was the facility location problem. In this problem, one is concerned with locating n units on n sites with some physical products flowing between the facilities, and with distances between the sites. The element a_{ij} is the "flow" from unit i to unit j , and the element b_{kl} represents the distance between the sites k and l . The permutation $\pi = (\pi(1), \pi(2), \dots, \pi(n))$ can be interpreted as an assignment of units to sites ($\pi(i)$ denotes the site what unit i is assigned to). Solving the QAP means searching for an assignment π that minimizes the "transportation cost" z .

An important area of the application of the QAP is computer-aided design (CAD), more precisely, the placement of electronic components into locations (positions) on a board (chip) [2,3,4]. Other applications of the QAP are: campus planning [5], hospital layout [6], image processing (design of grey patterns) [7], testing of electronic devices [8], typewriter keyboard design [9], etc (see [10,11] for a more detailed description of the QAP applications).

The quadratic assignment problem is one of the most complex combinatorial optimization problems. It has been proved that the QAP is NP-hard [12]. To this date problems of size n , where $n > 30$, are not practically solvable in terms of obtaining exact solutions. Therefore, heuristic approaches have to be used for solving medium- and large-scale QAPs: ant algorithms [13,14], genetic algorithms [15,16,17], greedy randomized search (GRASP) [18], iterated local search [19], simulated annealing [20,21], tabu search [22,23,24]. An extended survey of heuristics for the QAP one can find in [11,25].

This paper is organized as follows. In Section 2, basic definitions are given. Section 3 outlines the ruin and recreate approach for combinatorial optimization problems. Section 4 describes the R&R principle based algorithm for the quadratic assignment problem. The results of the computational experiments on the QAP instances are presented in Section 5. Section 6 completes the paper with conclusions.

2 Preliminaries

Let S be a set of solutions (solution space) of a combinatorial optimization problem with objective function $f: S \rightarrow R$. Furthermore, let $N: S \rightarrow 2^S$ be a neighbourhood function which defines for each $s \in S$ a set $N(s) \subseteq S$ – a set of neighbouring solutions of s . Each solution $s' \in N(s)$ can be reached from s by an operation called a move.

Suppose, $S = \{s \mid s = (s(1), s(2), \dots, s(n))\}$, where n is the cardinality of the set, i.e. the problem size. Given a solution s from S , a λ -exchange neighbourhood function $N_\lambda(s)$ is defined as follows:

$$N_\lambda(s) = \{s' \mid s' \in S, d(s, s') \leq \lambda\} \quad (2 \leq \lambda \leq n), \quad (2)$$

where $d(s, s')$ is the "distance" between solutions s and s' :

$$d(s, s') = \sum_{i=1}^n \text{sgn} |s(i) - s'(i)|.$$

For the QAP, the commonly used case is $\lambda = 2$, i.e. the 2-exchange function N_2 . In this case, a transformation from the current permutation (solution) π to the neighbouring permutation π' can formally be defined by using a special move (operator) – 2-way perturbation $p_{ij}: \Pi \rightarrow \Pi$ ($i, j = 1, 2, \dots, n$), which exchanges i th and j th elements in the current permutation. The notation $\pi' = \pi \oplus p_{ij}$ means that π' is obtained from π by applying p_{ij} . The difference in the objective function values (when moving from the current permutation to the neighbouring one) can be calculated in $O(n)$ operations:

$$\Delta_z(\pi, i, j) = (a_{ij} - a_{ji})(b_{\pi(j)\pi(i)} - b_{\pi(i)\pi(j)}) + \sum_{k=1, k \neq i, j}^n [(a_{ik} - a_{jk})(b_{\pi(j)\pi(k)} - b_{\pi(i)\pi(k)}) + (a_{ki} - a_{kj})(b_{\pi(k)\pi(j)} - b_{\pi(k)\pi(i)})], \quad (3)$$

where a_{ii} (or b_{ii}) = const, $\forall i \in \{1, 2, \dots, n\}$.

If the matrix A and/or matrix B are symmetric formula (3) becomes much more simpler. For example, if the matrix B is symmetric, one can transform the matrix A to the symmetric one A' by adding up corresponding entries of A .

Formula (3) then reduces to the following formula

$$\Delta_z(\pi, i, j) = \sum_{k=1, k \neq i, j}^n (a'_{ik} - a'_{jk})(b_{\pi(j)\pi(k)} - b_{\pi(i)\pi(k)}), \quad (4)$$

where $a'_{ik} = a_{ik} + a_{ki}$, $\forall i, k \in \{1, 2, \dots, n\}$, $i \neq k$. Moreover, for two consecutive solutions π and $\pi' = \pi \oplus p_{uv}$, if all values $\Delta_z(\pi, i, j)$ have been stored, the values $\Delta_z(\pi', i, j)$ ($i \neq u, v$ and $j \neq u, v$) can be computed in time $O(1)$ [24]:

$$\begin{aligned} \Delta_z(\pi', i, j) = \Delta_z(\pi, i, j) + (a_{iu} - a_{iv} + a_{ju} - a_{jv})(b_{\pi(i)\pi(u)} - b_{\pi(i)\pi(v)} + b_{\pi(j)\pi(v)} - b_{\pi(j)\pi(u)}) + \\ (a_{ui} - a_{vi} + a_{vj} - a_{uj})(b_{\pi(u)\pi(i)} - b_{\pi(v)\pi(i)} + b_{\pi(v)\pi(j)} - b_{\pi(u)\pi(j)}). \end{aligned} \quad (5)$$

(If $i = u$ or $i = v$ or $j = u$ or $j = v$, then formula (3) is applied.)

3 Ruin and Recreate Principle

The ruin and recreate (R&R) principle was formulated by Schrimpf et al. [26]. Note also that this principle has some similarities with other approaches, among them: combined local search heuristic (chained local optimization) [27], iterated local search [19], large step Markov chains [28], variable neighbourhood search [29]. As mentioned in [26], the basic element of the R&R principle is to obtain better optimization results by a reconstruction (destruction) of an existing solution and a following improvement (rebuilding) procedure. By applying this type of process frequently, i.e. in an iterative way, one seeks for high quality solutions. In the first phase of the process, one reconstructs a significant part of the existing solution, roughly speaking, one "ruins" the current solution. (That is a relatively easy part of the method.) In the second phase, one tries to rebuild the solution just "ruined" as best as one can. Hopefully, the new solution is better than the solution(s) obtained in the previous phase(s) of the improvement. (Naturally, the improvement is the harder part of the method.) There are a lot of different ways to reconstruct (ruin) and, especially, to improve the solutions. So, we think of the ruin and recreate approach as a meta-heuristic – not a pure heuristic.

It should be noted that the approach we have outlined differs a little from the original R&R version. The main distinguishing feature is that, in our version of R&R, there is considered rather an iterative process of some kind of mutations (as reconstructions) and local searches (as improvements) applied to solutions. In the original R&R, in contrast, one speaks rather of a sequence of disintegrations and constructions of solutions (see Schrimpf et al. [26]).

The advantage of the R&R approach – which is very similar to the iterated local search [19] – over the well-known random multistart method (that is based on multiple starts of local improvements applied to randomly generated solutions) is that, instead of generating (constructing) new solutions from scratch, a better idea is to reconstruct (a part of) the current solution: continuing search from this reconstructed solution may allow to escape from a local optimum and to find better solutions. (So, improvements of the reconstructions of local optima are considered rather than improvements of the purely random solutions.) For the same computation time, many more reconstruction improvements can be done than when starting from randomly

generated solutions, because the reconstruction improvement procedure requires only a few steps to reach the next local optimum.

The R&R principle can be thought of as a method with so-called "large" moves (moves consisting of a large number of perturbations) [26], instead of "small" moves (e.g. pair-wise interchanges) that typically take place in "classical" algorithms. For simple problems, there is no need to use "large" moves – "small" moves are enough. The "large" moves – that are also referred to as "kick" moves [27] – are very important when dealing with complex problem (QAP among them). These problems can often be seen as "discontinuous": if one walks in the solution space, the qualities of the solutions can be significantly different, i.e. the "landscapes" of these problems can be very rugged.

Two aspects regarding to the "kick" moves are of high importance (see also [19]). Firstly, the "kick" move should be large (strong) enough to allow to leave the current locally optimal solution and to enable the improvement procedure to find, probably, better solution. However, if the "kick" move is too large, the resulting algorithm might be quite similar to random multistart, which is known to be not a very efficient algorithm. Secondly, the "kick" move should be small (weak) enough to keep features (characteristics) of the current local minimum since parts of the solution may be close to the ones of the globally optimal solution. However, if the "kick" move is too small, the improvement procedure may return to the solution to which the "kick" move has been applied, i.e. a cycling may occur.

In the simplest case, it is sufficient to use for the "kick" move a certain number of random perturbations (further, we shall refer those perturbations to as mutation). Doing so can be interpreted as a random search of higher order neighbourhoods N_λ ($\lambda > 2$), i.e. random variable neighbourhood search [29].

Additional component of the R&R method is an acceptance criterion that is used to decide which solution is to be chosen for the "ruin". Here, two main alternatives are so-called intensification (exploitation) and diversification (exploration). Intensification is achieved by choosing only the best local optimum as a candidate for the "ruin". On the other hand, diversification takes place if every new local optimum is accepted for the reconstruction.

The paradigm of the ruin and recreate approach, which is conceptually surprisingly simple, is presented in Fig. 1.

4 Ruin and Recreate Principle Based Algorithm for the QAP

All we need by creating the ruin and recreate principle based algorithm for a specific problem is to design four components: 1) an initial solution generation (construction) procedure, 2) a solution improvement procedure (we shall also refer this procedure to as local search procedure), 3) a solution reconstruction (mutation) procedure, and 4) a candidate acceptance (choosing) rule.

Now we present details of the ruin and recreate principle based algorithm for the QAP, which is entitled R&R-QAP.

```

procedure R&R { ruin and recreate procedure }
  generate (or construct) initial solution  $s^\circ$ 
   $s^\bullet := \text{recreate}(s^\circ)$  { recreate (improve) the initial solution }
   $s^* := s^\bullet, s := s^\bullet$ 
  repeat { main loop }
     $s := \text{choose\_candidate\_for\_ruin}(s, s^\bullet)$ 
     $s^\sim := \text{ruin}(s)$  { ruin (reconstruct) the current solution }
     $s^\bullet := \text{recreate}(\sim)$  { recreate (improve) the ruined solution }
    if  $s^\bullet$  is better than  $s^*$  then  $s^* := s^\bullet$ 
  until termination criterion is satisfied
  return  $s^*$ 
end { R&R }

```

Fig. 1. The paradigm of the ruin and recreate (R&R) principle based procedure

4.1 Initial Solution Generation

We use randomly generated permutations as initial permutations for the algorithm R&R-QAP. These permutations can be generated by a very simple procedure.

4.2 Local Search

In principle, any local search concept based algorithm can be applied at the improvement phase of R&R method. In the simplest case, a greedy descent ("first improvement") algorithm or steepest descent ("best improvement") algorithm (also known as a "hill climbing") can be used. Yet, it is possible to apply more sophisticated algorithms, like limited simulated annealing or tabu search. In our algorithm, we use the tabu search algorithm, more precisely, a modified version of the robust tabu search algorithm due to Taillard [24]. The framework of the algorithm can briefly be described as follows. Initialize tabu list, $T = (t_{ij})_{n \times n}$, and start from an initial solution π . Continue the following process until a termination criterion is satisfied (a predetermined number of trials is executed): a) find a neighbour π' of the current solution π in such a way that $\pi'' = \arg \min_{\pi' \in N_2(\pi)} z(\pi')$, where

$$N'_2(\pi) = \{\pi' \mid \pi' \in N_2(\pi), (\pi' = \pi \oplus p_{ij} \text{ and } p_{ij} \text{ is not tabu}) \text{ or } z(\pi') < z(\pi''')\} \quad (\pi''' \text{ is}$$

the best so far solution); b) replace the current solution π by the neighbour π' (even if $z(\pi') - z(\pi) > 0$), and use as a starting point for the next trials; c) update the tabu list T . The last found solution, which is locally optimal, is declared as a solution of the tabu search algorithm. The detailed template of the tabu search based local search algorithm for the QAP is presented in Fig. 2.


```

function local_search( $\pi, n, \tau$ ) { local search (based on tabu search) for the QAP }
    {  $\pi$  – the current permutation,  $n$  – problem size,  $\tau$  – number of iterations }
    set lower and higher tabu sizes  $h_{min}$  and  $h_{max}$ 
     $\pi^* := \pi$  {  $\pi^*$  denotes the best permutation found }
    calculate the objective function differences  $\delta_{ij} = \Delta_z(\pi, i, j)$ :  $i = \overline{1, n-1}$ ;  $j = \overline{i+1, n}$ 
     $T := 0$ 
    choose  $h$  randomly between  $h_{min}$  and  $h_{max}$ 
     $i := 1, j := 1, k := 1$ 
    improved := FALSE
    while ( $k \leq \tau$ ) or improved do begin { main loop }
         $\delta_{min} := \infty$  {  $\delta_{min}$  – minimum difference in the objective function values }
        for  $l := 1$  to  $|N_2|$  do begin
             $i := \text{if}(j < n, i, \text{if}(i < n-1, i+1, 1))$ ,  $j := \text{if}(j < n, j+1, i+1)$ 
             $\text{tabu} := \text{if}(t_{ij} \geq k, \text{TRUE}, \text{FALSE})$ ,  $\text{aspired} := \text{if}(z(\pi) + \delta_{ij} < z(\pi^*), \text{TRUE}, \text{FALSE})$ 
            if ( $(\delta_{ij} < \delta_{min})$  and not  $\text{tabu}$ ) or  $\text{aspired}$  then begin
                 $\delta_{min} := \delta_{ij}$ ,  $u := i$ ,  $v := j$ 
            end { if }
        end { for }
        if  $\delta_{min} < \infty$  then begin
             $\pi := \pi \oplus p_{uv}$  { replace the current permutation by the new one }
            if  $z(\pi) < z(\pi^*)$  then  $\pi^* := \pi$ 
             $\text{improved} := \text{if}(\delta_{uv} < 0, \text{TRUE}, \text{FALSE})$ 
            update the differences  $\delta_{lm}$ :  $l = \overline{1, n-1}$ ;  $m = \overline{l+1, n}$ 
             $t_{uv} := k + h$ 
            if  $k \bmod 2h_{max} = 0$  then choose new  $h$  randomly between  $h_{min}$  and  $h_{max}$ 
        end { if }
    end { while }
    return  $\pi = \pi^*$ 
end { local_search }
    
```

Fig. 2. Template of the tabu search based local search algorithm for the QAP.

- Notes. 1. The objective function difference δ_{ij} is calculated according to (3) (or 4) formula.
 2. The difference δ_{lm} is updated according to (5) formula.
 3. The function $\text{if}(x, y_1, y_2)$ returns y_1 if $x = \text{TRUE}$, otherwise it returns y_2

4.3 Mutation

As mentioned in Section 3, a mutation of the current solution is achieved by performing a complex move, i.e. a move in the neighbourhood N_λ , where $\lambda > 2$. This can be modelled by generating $\mu = \lfloor (\lambda + 1) / 2 \rfloor$ sequential elementary perturbations, like p_{ij} ; here, the parameter μ ($\mu \geq 2$) is referred to as a mutation level.

```

function mutation( $\pi, n, \mu$ ) { mutation procedure for the QAP }
{  $\pi$  – the current permutation,  $n$  – problem size,  $\mu$  – mutation level ( $2 \leq \mu \leq n$ ) }
 $\pi^* := \pi$ 
for  $k := 1$  to  $\mu$  do begin { main loop }
   $i := \text{randint}(1, n), j := \text{randint}(1, n-1)$ 
  if  $i \leq j$  then  $j := j + 1$ 
   $\pi := \pi \oplus p_{ij}$  { replace the current permutation by the new one }
  if  $z(\pi) < z(\pi^*)$  then break
end { for  $k$  }
return  $\pi$ 
end { mutation }

```

Fig. 3. Template of the mutation procedure for the QAP.

Note. The function $\text{randint}(x, y)$ returns integer random, uniformly distributed number between x and y

```

procedure R&R-QAP { ruin and recreate principle based algorithm for the QAP }
{ input:  $A, B$  – flow and distance matrices,  $n$  – problem size }
{  $Q$  – the total number of iterations }
{  $\alpha, \beta_{\min}, \beta_{\max}$  – the control parameters ( $\alpha > 0, 0 < \beta_{\min} \leq \beta_{\max} \leq 1$ ) }
{ output:  $\pi^*$  – the best permutation found }
 $\tau := \max(1, \lfloor \alpha n \rfloor), \mu_{\min} := \max(2, \lfloor \beta_{\min} n \rfloor), \mu_{\max} := \max(2, \lfloor \beta_{\max} n \rfloor)$ 
generate random initial permutation  $\pi^0$ 
 $\pi^* := \text{local\_search}(\pi^0, n, \tau)$ 
 $\pi^* := \pi^*, \pi := \pi^*$ 
 $\mu := \mu_{\min} - 1$  {  $\mu$  is the current value of the mutation level }
for  $q := 1$  to  $Q$  do begin { main loop of R&R-QAP }
   $\pi := \text{if}(z(\pi^*) < z(\pi), \pi^*, \pi)$  {  $\pi$  is the candidate for the mutation }
   $\mu := \text{if}(\mu < \mu_{\max}, \mu + 1, \mu_{\min})$ 
   $\tilde{\pi} := \text{mutation}(\pi, n, \mu)$  { mutate (reconstruct) the permutation  $\pi$  }
   $\pi^* := \text{local\_search}(\tilde{\pi}, n, \tau)$  { try to improve the mutated permutation }
  if  $z(\pi^*) < z(\pi^*)$  then begin
     $\pi^* := \pi^*, \mu := \mu_{\min} - 1$  { save the best permutation, reset the mutation level }
  end { if }
end { for  $q$  }
return  $\pi^*$ 
end { R&R-QAP }

```

Fig. 4. Template of the ruin and recreate (R&R) principle based algorithm for the QAP

We can add more robustness to the R&R method if we let vary the parameter μ in some interval, say $[\mu_{\min}, \mu_{\max}] \subseteq [2, n]$ (n is the problem size), during the execution of the algorithm. Two ways are possible when varying μ : random and deterministic. In the first case, "a coin is tossed"; in the second one, μ is varied sequentially within the interval $[\mu_{\min}, \mu_{\max}]$ starting from $\mu_{\min}$. Once maximum value $\mu_{\max}$ has been reached (or a better locally optimum solution has been found), the value of μ is dropped to the minimum value $\mu_{\min}$, and so on. The last case is used in our implementation. The detailed template of the mutation procedure is presented in Fig. 3.

4.4 Candidate Acceptance

In our algorithm, we accept only the best locally optimal solution as a candidate for the mutation. The candidate solution at q th iteration is defined according to the following formula $\pi^{(q)} = \text{if}(z(\pi^*) < z(\pi^{(q-1)}), \pi^*, \pi^{(q-1)})$, where $\pi^{(q)}$ is the candidate solution at the current iteration, $\pi^{(q-1)}$ is the solution before the last local search execution (the best so far solution), and π^* is the solution after the current local search execution.

The template of the resulting R&R approach based algorithm is presented in Fig. 4. Note that, for convenience, the parameters $\alpha > 0$, $\beta_{\min}, \beta_{\max} \in (0, 1]$ are used instead of $\tau, \mu_{\min}, \mu_{\max}$, respectively.

5 Computational Experiments and Results

We have carried out a number of computational experiments in order to test the performance of the algorithm R&R-QAP. The following types of the QAP instances taken from the quadratic assignment problem library QAPLIB [30] were used:

a) real-life instances (instances of this class are real world instances from practical applications of the QAP, among them: chr25a, els19, esc32a, esc64a, kra30a, ste36a, tai64c (also known as grey_8_8_13 [7]));

b) real-life like instances (they are generated in such a way that the entries of the matrices A and B resemble a distribution from real-life problems; the instances are denoted by tai20b, tai25b, tai30b, tai35b, tai40b, tai50b, tai60b, tai80b, tai100b (tai*b)).

For the comparison, we used the tabu search algorithm due to Taillard [24] entitled as the robust tabu search algorithm – RTS-QAP, because it is among the best algorithms for the quadratic assignment problem. As a performance measure, the average deviation from the best known solution is chosen. The average deviation, θ_{avg} , is defined according to the formula $\theta_{\text{avg}} = 100(z_{\text{avg}} - z_b)/z_b$ [%], where z_{avg} is the average objective function value over W restarts of the algorithm, and z_b is the best known value (BKV) of the objective function. (Note: BKVs are from [30].)

All the experiments were carried out on x86 Family 6 processor. The computations were organized in such a way that both algorithms use identical initial assignments, and require the similar CPU times. The execution of the algorithms is controlled by a fixed a priori number of iterations. Some differences of the run time for RTS-QAP

and R&R-QAP are due to non-deterministic behaviour of the local search procedure used in R&R-QAP. The results of the comparison, i.e. the average deviations from BKV for both RTS-QAP and R&R-QAP, as well as approximated CPU times per restart (in seconds) are presented in Table 1. The deviations are averaged over 100 restarts. The best values are printed in bold face.

Table 1. Comparison of the algorithms on real-life and real-life like instances ($Q=50$, $\alpha=0.1$)

Instance name	n	BKV	θ_{avg}				t_{avg}
			RTS-QAP	R&R-QAP ^a	R&R-QAP ^b	R&R-QAP ^c	
chr25a	25	3796	21.198	16.596	18.891	20.874	0.05
els19	19	17212548	22.765	0.035	0.123	0.018	0.03
esc32a	32	130	9.692	7.123	7.862	8.477	0.09
esc64a	64	116	0.966	0.000	0.000	0.000	0.50
kra30a	30	88900	2.692	2.422	2.272	2.247	0.09
ste36a	36	9526	3.562	2.561	2.456	2.771	0.16
tai20b	20	122455319	14.258	0.261	0.263	0.226	0.03
tai25b	25	344355646	14.251	0.744	0.419	0.341	0.06
tai30b	30	637117113	13.519	1.112	1.093	0.949	0.10
tai35b	35	283315445	8.014	0.799	0.715	0.563	0.16
tai40b	40	637250948	10.031	1.459	1.492	1.440	0.27
tai50b	50	458821517	6.741	0.655	0.719	0.631	0.50
tai60b	60	608215054	7.026	1.143	1.050	1.080	0.90
tai80b	80	818415043	5.681	1.360	1.434	1.589	2.3
tai100b	100	1185996137	4.696	0.910	0.853	0.747	5.0
tai64c	64	1855928	0.410	0.002	0.000	0.003	0.55

^a $\beta_{\min}=0.35$, $\beta_{\max}=0.45$; ^b $\beta_{\min}=0.40$, $\beta_{\max}=0.50$; ^c $\beta_{\min}=0.45$, $\beta_{\max}=0.55$.

Looking at Table 1, it turns out that the efficiency of the algorithms depends on the type of the instances (problems) being solved. For some real-life instances, R&R-QAP produces only slightly better results than RTS-QAP. For the real-life like problems, the situation is different: for these problems, R&R-QAP is much more better than RTS-QAP. The results for the particular instances differ dramatically (see, for example, the results for the instances tai20b, tai25b, tai35b).

Two directions that may lead to the improvement of the results of R&R-QAP are as follows: the parameter-improvement-based direction, and the algorithm's-idea-improvement-based direction. First of all, the results can be improved by a more accurate tuning the control parameters α , $\beta_{\min}$ and $\beta_{\max}$. This can be seen from Table 1: by the proper choosing the values $\beta_{\min}$, $\beta_{\max}$ the average deviation can be lowered noticeably (see, for example, the results for chr25a, els19, tai25b, tai35b). The results can also be improved by increasing the value of the parameter Q , but at the cost of a longer computation time. Some results for the instances tai*b are shown in Table 2.

Table 2. Additional computational results of R&R-QAP ($\alpha=0.01$, $\beta_{\min}=0.45$, $\beta_{\max}=0.55$). The following notations are used: W – the total number of restarts, $W^{1\%}$ – the number of restarts at which the solution that is within 1% optimality was found, W^* – the number of restarts at which best known solution was found, t_{avg} – the average CPU time per restart (in seconds)

Instance name	R&R-QAP ($Q=1000$)			R&R-QAP ($Q=10000$)			R&R-QAP ($Q=100000$)		
	θ_{avg}	$W/W^{1\%}/W^*$	t_{avg}	θ_{avg}	$W/W^{1\%}/W^*$	t_{avg}	θ_{avg}	$W/W^{1\%}/W^*$	t_{avg}
tai20b	0.000	30/30/300.17							
tai25b	0.000	30/30/300.36							
tai30b	0.000	30/30/300.67							
tai35b	0.083	30/30/19 1.1		0.037	30/30/24 10.7		0.000	30/30/30 107	
tai40b	0.134	30/28/27 1.7		0.000	30/30/30 17.0				
tai50b	0.224	30/30/8 3.5		0.169	30/30/6 34.9		0.027	30/30/28 349	
tai60b	0.587	30/20/5 6.2		0.000	30/30/30 62.4				
tai80b	0.598	30/25/0 16.4		0.432	30/29/5 164		0.198	30/30/19 1640	
tai100b	0.195	30/30/0 35.6		0.121	30/30/5 356		0.086	30/30/10 3560	

It should be stressed that R&R-QAP was successful in finding a large number of new best solutions for so-called problems of grey densities. Problems of this type are described in [7,31] under the name `grey_n1_n2_m`, where m is the density of the grey ($0 \leq m \leq n = n_1 \cdot n_2$), and $n_1 \cdot n_2$ is the size of a frame. New solutions for the instance family `grey_16_16_*` are presented in Table 3.

Table 3. New best known solutions for grey density problems

Instance name	Previous best known value ^a	New best known value	Instance name	Previous best known value ^a	New best known value
grey_16_16_25	2216160	2215714	grey_16_16_57	14798834	14793682
grey_16_16_43	7795062	7794422	grey_16_16_58	15395948	15363628
grey_16_16_44	8219428	8217264	grey_16_16_61	17202312	17194812
grey_16_16_45	8677670	8676802	grey_16_16_62	17824828	17822806
grey_16_16_46	9134172	9130426	grey_16_16_65	19859448	19848790
grey_16_16_49	10523016	10518838	grey_16_16_66	20655396	20648754
grey_16_16_51	11517060	11516840	grey_16_16_67	21461130	21439396
grey_16_16_52	12019082	12018388	grey_16_16_68	22271676	22234020
grey_16_16_54	13098850	13096646	grey_16_16_69	23086332	23049732
grey_16_16_55	13668486	13661614	grey_16_16_70	23898390	23852796
grey_16_16_56	14238840	14229492			

^a comes from [31].

6 Conclusions

The quadratic assignment problem is a very difficult combinatorial optimization problem. In order to obtain satisfactory results in a reasonable time, heuristic algorithms are to be applied. One of them, a ruin and recreate principle based algorithm (R&R-QAP) is described in this paper. The proposed algorithm applies a reconstruction (mutation) to the best solution found so far and subsequently applies an improvement (local search) procedure. This scheme – despite its conceptual simplicity, easy programming and practical realization – allowed to achieve very good results with small amount of the computation time. The results from the comparison with the robust tabu search algorithm ("pure" tabu search algorithm) show that R&R-QAP produces much more better results than this algorithm (one of the most efficient heuristics for the QAP), as far as real-life and real-life like problems are concerned. The power of R&R-QAP is also corroborated by the fact that many new best known solutions were found for the largest available QAP instances, so-called grey density problems.

There are several possible directions to try to improve the performance of the proposed algorithm: a) implementing a faster local search procedure; b) investigating new reconstruction (mutation) operators; c) introducing a memory based mechanism when selecting the solutions (candidates) for the reconstruction; d) incorporating R&R-QAP into hybrid genetic algorithms as an initial population generation and/or local improvement procedure.

References

1. Koopmans, T., Beckmann, M.: Assignment Problems and the Location of Economic Activities. *Econometrica* 25 (1957) 53–76
2. Hanan, M., Kurtzberg, J.M.: Placement Techniques. In: Breuer, M.A. (ed.): *Design Automation of Digital Systems: Theory and Techniques*, Vol.1. Prentice-Hall (1972) 213–282
3. Hu, T.C., Kuh, E.S. (ed.): *VLSI Circuit Layout: Theory and Design*. IEEE Press, New York (1985)
4. Steinberg, L.: The Backboard Wiring Problem: A Placement Algorithm. *SIAM Review* 3 (1961) 37–50
5. Dickey, J.W., Hopkins, J.W.: Campus Building Arrangement Using TOPAZ. *Transportation Research* 6 (1972) 59–68
6. Elshafei, A.N.: Hospital Layout as a Quadratic Assignment Problem. *Operations Research Quarterly* 28 (1977) 167–179
7. Taillard, E.: Comparison of Iterative Searches for the Quadratic Assignment Problem. *Location Science* 3 (1995) 87–105
8. Eschermann, B., Wunderlich, H.J.: Optimized Synthesis of Self-testable Finite State Machines. 20th International Symposium on Fault-Tolerant Computing (FFTC20) (1990)
9. Burkard, R.E., Offermann, J.: Entwurf von Schreibmaschinentastaturen mittels Quadratischer Zuordnungsprobleme. *Zeitschrift fuer Operations Research* 21 (1977) 121–132
10. Burkard, R.E.: Locations with Spatial Interactions: The Quadratic Assignment Problem. In: Mirchandani, P.B., Francis, R.L. (eds.): *Discrete Location Theory*. Wiley, New York (1991) 387–437

11. Burkard, R.E., Çela, E., Pardalos, P.M., Pitsoulis, L.: The Quadratic Assignment Problem. In: Du, D.Z., Pardalos, P.M. (eds.): *Handbook of Combinatorial Optimization*, Vol.3. Kluwer (1998) 241–337
12. Sahni, S., Gonzalez, T.: P-complete Approximation Problems. *J. of ACM* 23 (1976) 555–565
13. Gambardella, L.M., Taillard, E., Dorigo, M.: Ant Colonies for the Quadratic Assignment Problems. *J. of the Operational Research Society* 50 (1999) 167–176
14. Stuetzle, T., Dorigo, M.: ACO Algorithms for the Quadratic Assignment Problem. In: Corne, D., Dorigo, M., Glover, F. (eds.): *New Ideas in Optimization*. McGraw-Hill (1999) 33–50
15. Ahuja, R.K., Orlin, J.B., Tiwari, A.: A Greedy Genetic Algorithm for the Quadratic Assignment Problem. *Computers & Operations Research* 27 (2000) 917–934
16. Fleurent, C., Ferland, J.A.: Genetic Hybrids for the Quadratic Assignment Problem. In: Pardalos, P.M., Wolkowicz, H. (eds.): *Quadratic Assignment and Related Problems*. DIMACS Series in Discrete Mathematics and Theoretical Computer Science, Vol.16. AMS, Providence (1994) 173–188
17. Merz, P., Freisleben, B.: A Genetic Local Search Approach to the Quadratic Assignment Problem. *Proceedings of the Seventh International Conference on Genetic Algorithms (ICGA'97)* (1997)
18. Li, Y., Pardalos, P.M., Resende, M.G.C.: A Greedy Randomized Adaptive Search Procedure for the Quadratic Assignment Problem. In: Pardalos, P.M., Wolkowicz, H. (eds.): *Quadratic Assignment and Related Problems*. DIMACS Series in Discrete Mathematics and Theoretical Computer Science, Vol.16. AMS, Providence (1994) 237–261
19. Stuetzle, T.: Iterated Local Search for the Quadratic Assignment Problem. Tech. Report AIDA-99-03, Darmstadt, Germany (1999)
20. Boelte, A., Thonemann, U.W.: Optimizing Simulated Annealing Schedules with Genetic Programming. *European J. of Operational Research* 92 (1996) 402–416
21. Connolly, D.T.: An Improved Annealing Scheme for the QAP. *European J. of Operational Research* 46 (1990) 93–100
22. Battiti, R., Tecchioli, G.: The Reactive Tabu Search. *ORSA J. on Computing* 6 (1994) 126–140
23. Skorin-Kapov, J.: Tabu Search Applied to the Quadratic Assignment Problem. *ORSA J. on Computing* 2 (1990) 33–45
24. Taillard, E.: Robust Taboo Search for the QAP. *Parallel Computing* 17 (1991) 443–455
25. Çela, E.: *The Quadratic Assignment Problem: Theory and Algorithms*. Kluwer, Dordrecht (1998)
26. Schrimpf, G., Schneider, K., Stamm-Wilbrandt, H., Dueck, V.: Record Breaking Optimization Results Using the Ruin and Recreate Principle. *J. of Computational Physics* 159 (2000) 139–171
27. Martin, O., Otto, S.W.: Combining Simulated Annealing with Local Search Heuristics. *Annals of Operations Research* 63 (1996) 57–75
28. Martin, O., Otto, S.W., Felten, E.W.: Large-step Markov Chains for the Traveling Salesman Problem. *Complex Systems* 5 (1991) 299–326
29. Mladenović, N., Hansen, P.: Variable Neighbourhood Search. *Computers & Operations Research* 24 (1997) 1097–1100
30. Burkard, R.E., Karisch, S., Rendl, F.: QAPLIB – A Quadratic Assignment Problem Library. *J. of Global Optimization* 10 (1997) 391–403
31. Taillard, E., Gambardella, L.M.: Adaptive Memories for the Quadratic Assignment Problem. Tech. Report IDSIA-87-97, Lugano, Switzerland (1997)

Model-Assisted Steady-State Evolution Strategies

Holger Ulmer, Felix Streichert, and Andreas Zell

Center for Bioinformatics Tübingen (ZBIT), University of Tübingen,
Sand 1, 72074 Tübingen, Germany,
{ulmerh,streiche,zell}@informatik.uni-tuebingen.de,
<http://www-ra.informatik.uni-tuebingen.de>

Abstract. The task of speeding up the optimization process on problems with very time consuming fitness functions is a central point in evolutionary computation. Applying models as a surrogate of the real fitness function is a quite popular idea. The performance of this approach is highly dependent on the frequency of how often the model is updated with data from new fitness evaluations. However, in generation based algorithms this is only done every λ -th fitness evaluation. To overcome this problem we use a steady-state strategy, which updates the model immediately after each fitness evaluation. We present a new model assisted steady-state Evolution Strategy (ES), which uses Radial-Basis-Function networks as a model. To support self-adaption in the steady-state algorithm a median selection scheme is applied. The convergence behavior of the new algorithm is examined with numerical results from extensive simulations on several high dimensional test functions. It achieves better results than standard ES, steady-state ES or model assisted ES.

1 INTRODUCTION

Evolution Strategies (ES) were developed in the late 60s by Rechenberg and Schwefel [10] [11]. They are known as excellent optimization tools for complex high dimensional multimodal real valued problems. However, they require a very high number of fitness function evaluations. In many real world applications, like high throughput material science or design optimization, the fitness evaluation is very expensive and time consuming. Therefore standard ES methods are not practical for such applications.

One approach to overcome this problem is the application of modeling techniques: To decrease the number of expensive fitness evaluations, approximation models, also known as metamodels, are used instead of the exact fitness function. These models are orders of magnitude cheaper to evaluate than the real fitness function.

The use of metamodeling techniques in evolutionary computation receives increasing attention [9] [7] [3] [4]. A survey on this research field can be found in [8].

The selection of an appropriate model to approximate the fitness function is a central point. Neural networks are widely used for function approximation and are therefore used for modeling in evolutionary optimization [7] [6]. Gaussian processing and kriging are statistical modeling techniques which are also used for modeling [9] [2] [3].

The coupling of the metamodel with the evolutionary optimization process controls how the optimization process is affected by replacing the expensive real fitness evaluation by the approximation of the model. The adaptive evolution control concept [6] [7] controls the impact of the model on the evolutionary optimization process. The Metamodel-Assisted Evolution Strategy (MA-ES) [4] uses the estimation of the model to preselect the most promising individuals before applying the expensive real fitness function. Another approach is to use the criterion confidence given by statistical models like kriging [4] or gaussian processing [3] to control the interaction of the metamodel with the evolutionary optimization process.

Our work aims at improving the coupling between the metamodel and the ES by using a steady-state strategy. The remainder of this paper is organized as follows: We first describe the modeling technique of Radial-Basis-Function (RBF) networks, which is used in this work as a surrogate for the expensive real fitness function, in section 2. Section 3 introduces the synthesis of the metamodel with a standard generation based ES. We propose the new Model Assisted Steady-State ES (MASS-ES) with median selection scheme in section 4.

Numerical results from extensive simulations on several high dimensional test functions are presented in section 5. The paper closes with a brief conclusion and outlook on future work.

2 RBF NETWORK MODEL

Consider a d -dimensional real valued problem with fitness function $f(\mathbf{x})$, which has to be minimized. As stated in the introduction, a model is needed to predict the fitness $\hat{f}(\mathbf{x})$ of an individual to save time consuming and expensive evaluations of the real fitness function $f(\mathbf{x})$.

We chose in our study Radial-Basis-Function (RBF) networks, which are known to be general and proper real valued function approximators. A comprehensive description of RBF-networks is given in [1].

The RBF network maps the objective variable $\mathbf{x}$ of an individual to its corresponding fitness value $\hat{f}(\mathbf{x})$. The network consists of an input layer of d units (one for each input dimension), a single hidden layer of h nonlinear processing units and an output layer of linear weights ω_i (see figure 1).

The output $\hat{f}$ of the RBF network is given as a linear combination of a set of radial basis functions:

$$\hat{f}(\mathbf{x}) = \omega_0 + \sum_{i=1}^h \omega_i \phi_i(\|\mathbf{x} - \mathbf{c}_i\|) \quad (1)$$

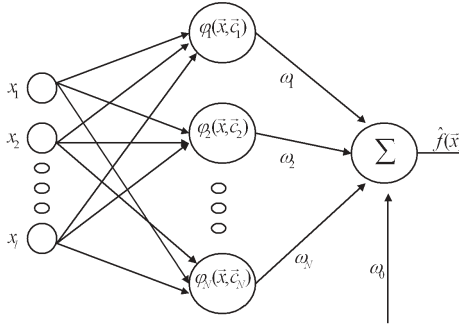


Fig. 1. Structure of a RBF network. Left side input layer of d units (one for each input dimension), hidden layer and output unit.

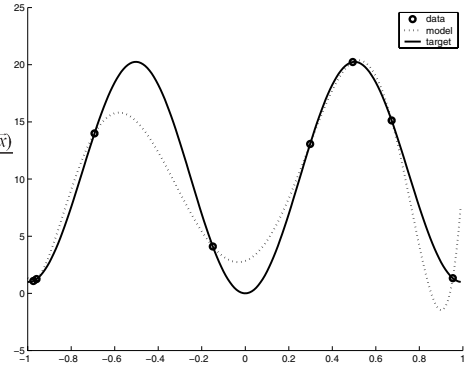


Fig. 2. Model output of RBF network for an one dimensional Rastrigin's test function (see A.5).

The term $\phi_i(\|\mathbf{x} - \mathbf{c}_i\|)$ represents the i -th radial basis function which evaluates the distance between the input $\mathbf{x}$ and the center $\mathbf{c}_i$. For ϕ_i we use the gaussian kernel, which is the most common used in practice.

$$\phi_i(\|\mathbf{x} - \mathbf{c}_i\|) = \exp\left(-\frac{\|\mathbf{x} - \mathbf{c}_i\|^2}{2\sigma^2}\right) \quad (2)$$

The centers $\mathbf{c}_i$ are determined by a simple k-means clustering algorithm. The parameter σ represents the width of the radial basis functions, which is given here by the mean euclidian distance of all nearest neighbor pairs on the data inputs $\mathbf{x}_j$.

Inserting the fitness cases $(\mathbf{x}_j, f(\mathbf{x}_j))$ from previous fitness evaluations into equation 1 leads to a linear equation system from which the weight parameter vector $\boldsymbol{\omega}$ can be computed by least squares.

We used the RBF network without regularisation. In figure 2 a RBF network output can be seen for a model which is trained with 10 data points from a one dimensional test function (see A.5).

3 MODEL ASSISTED STANDARD EVOLUTION STRATEGY

The start of our research is based on a standard (μ, λ) ES which will be later coupled with the model.

3.1 STANDARD (μ, λ) EVOLUTION STRATEGY

An ES works on a population of potential solutions $\mathbf{x}$ (individuals) by manipulating these individuals with evolutionary operators. λ offspring individuals are

generated from μ parents by applying the evolutionary operators reproduction, recombination and mutation (see pseudocode in figure 3). After evaluating the fitness of the offspring, μ individuals with the best fitness are selected to build the parent population for the next generation.

The most important evolutionary operator for an ES is the mutation of the objective variables representing the solution of the problem, which is responsible for the self-adaptation capability of the ES. Throughout our study we use Covariance Matrix Adaption (CMA) developed by Hansen et al. [5], which is still the most powerful method, without recombination. The algorithm terminates when a maximum number of fitness function evaluations have been performed.

Procedure ES

```

Begin
  eval=0;
  Pop=CreateInitialPop();
  Pop.EvaluateRealFitness();

  while (eval<maxeval);
    Offspring=Pop.Reproduce( $\lambda$ );
    Offspring.Mutate();

    Offspring.EvaluateRealFitness();

    Pop=Offspring.SelectBest( $\mu$ );
    eval=eval+lambda;
  end while
End

```

Fig. 3. Standard (μ, λ) Evolution Strategy (ES).

Procedure MA-ES

```

Begin
  eval=0;
  Pop=CreateInitialPop();
  Pop.EvaluateRealFitness();
  RBFModel.update(Pop);
  while (eval<maxeval);
    PrePop=Pop.Reproduce( $\lambda_{Pre}$ );
    PrePop.Mutate();
    PrePop.EvaluateWithRBFModel();
    Offspring=PrePop.SelectBest( $\lambda$ );
    Offspring.EvaluateRealFitness();
    RBFModel.update(Offspring);
    Pop=Offspring.SelectBest( $\mu$ );
    eval=eval+lambda;
  end while
End

```

Fig. 4. Model Assisted Evolution Strategy (MA-ES).

3.2 MODEL ASSISTED EVOLUTION STRATEGY

To couple the ES with the model we use a preselection concept similar to the one described by Emmerich et al. [4] (see pseudocode in figure 4).

Compared to the standard ES $\lambda_{Pre} > \lambda$ new offspring individuals are reproduced and mutated from μ parents. These offspring individuals are evaluated by using the prediction $\hat{f}(\mathbf{x})$ of the model. Out of these λ_{Pre} individuals λ individuals with the best predicted fitness are preselected to build the new offspring population. Finally the offspring were evaluated with the real fitness function. The model is updated after each generation step with λ new fitness cases. The idea behind this preselection approach is that only the most promising individuals with a good fitness prediction are evaluated with the real fitness function.

We use a comma selection strategy which selects the μ best individuals out of the λ offspring individuals instead of a plus selection strategy which selects the best μ out of the λ offspring individuals plus its μ parents. The comma selection strategy has the advantage of a better support of the self-adaption mechanism for the mutation step size control [12], which is necessary to solve complex problems.

The size of the preselected population λ_{Pre} controls the impact of the model on the evolutionary optimization process. For $\lambda_{Pre} = \lambda$ the algorithm performs like a standard (μ, λ) ES. Increasing λ_{Pre} results in a stronger impact of the model on the convergence behavior of the optimization process.

4 MODEL ASSISTED STEADY-STATE EVOLUTION STRATEGY

It is obvious that the performance of a metamodeling approach for an evolutionary optimization process is highly dependent on the quality of the model used. The more often the model is updated with already evaluated fitness cases the better is the quality of the model.

The MA-ES described in the last section has hereby a problem. It is generation based and its model is updated only after λ fitness evaluations. This problem could be fixed by dividing the preselection of the λ offspring individuals into λ steps. Within each step the model should be updated by only one preselected and evaluated individual. But this method leads to a much more complicated algorithm. Therefore we discuss an alternative approach of applying steady-state strategies:

A standard steady-state ES is equivalent to a $(\mu + 1)$ ES [10]. Only one individual is generated and evaluated at each step and gets immediately integrated into the population. Compared to generation based algorithms the information of new evaluated individuals can be integrated directly into the optimization process. But standard steady-state strategies have one big disadvantage. They are missing the ability of self-adaption, which is the most important property of ES. As a result of this the application of standard steady-state ES is not very promising. However, this problem can be solved by the median selection scheme, suggested by Wakunda [13].

The idea is to approximate the selection mechanism of a standard (μ, λ) ES, by using a fitness buffer containing fitness values of the last n_P evaluations. Given a relative rate of acceptance $r_P \equiv \frac{\mu}{\lambda}$. A newly evaluated individual substitutes the worst individual of the population, if it has a better fitness than the $r_P \cdot n_P$ best individuals in the buffer. The pseudocode for the steady-state ES is given in figure 5.

The model is coupled with the steady-state ES in the same way as MA-ES, which results in the new Model Assisted Steady-State ES (MASS-ES) (see pseudocode figure 6). The median selection scheme is again used. The main difference as mentioned before is, that the model will be updated after each fitness evaluation.

Procedure SS-ES

```

Begin
  eval=0;
  Pop=CreateInitialPop();
  Pop.EvaluateRealFitness();

  while (eval<maxeval);
    Individual=Pop.Reproduce(1);
    Individual.Mutate();

    Individual.EvaluateRealFitness();

    Pop = Pop.SelectMedian(Individual);
    eval=eval+1;
  end while
End

```

Fig. 5. Steady-State Evolution Strategy (SS-ES) with median selection scheme.

Procedure MASS-ES

```

Begin
  eval=0;
  Pop=CreateInitialPop();
  Pop.EvaluateRealFitness();
  RBFModel.update(Pop);
  while (eval<maxeval);
    PrePop=Pop.Reproduce( $\lambda_{Pre}$ );
    PrePop.Mutate();
    PrePop.EvaluateWithRBFModel();
    Individual=PrePop.SelectBest(1);
    Individual.EvaluateRealFitness();
    RBFModel.update(Individual);
    Pop=Pop.SelectMedian(Individual);
    eval=eval+1;
  end while
End

```

Fig. 6. Model Assisted Steady-State Evolution Strategy (MASS-ES) with median selection scheme.

5 EXPERIMENTAL RESULTS AND DISCUSSION

To analyze and compare the performance of the algorithms extensive simulations were performed for several well-known-real valued 10-dimensional test functions. For each test function the following strategies were compared:

- Standard (μ, λ) -ES ($\mu = 2, \lambda = 8$),
- RBF network model assisted ES ($\mu = 2, \lambda = 8, \lambda_{Pre} = 30$),
- Steady-state ES with median selection ($n_P = 10, r_P = 0.15$),
- RBF network model assisted steady-state ES with median selection ($n_P = 10, r_P = 0.15, \lambda_{Pre} = 10$).

All four strategies use covariance matrix adaptation (CMA) for the adaptation of the strategy parameters. For each test function two figures are presented, first the best individual for each generation, and second the mean squared error (MSE) of all N fitness predictions done by the model, both as a function of the number of evaluations of the real fitness function.

$$MSE = \frac{1}{N} \cdot \sum_{i=1}^N \left(f(x) - \hat{f}(x) \right)^2 \quad (3)$$

The MSE represents the quality of the RBF network model during the optimization process. The values are always evaluated as the mean from 100 repeated runs with different seed values for random number generation.

The model was built in all cases by the RBF network with 10 hidden neurons,

in section 2 described. The training data for the network consists of the 30 most recently performed fitness evaluations. For this reason the RBF-network is a local model of the individual's neighborhood in object space. Using more training data would improve the performance only slightly but comes along with much higher computational costs for model training.

5.1 UNIMODAL TEST FUNCTIONS

The sphere function (A.1) and the weighted sphere function (A.2) are continuously, convex, smooth functions, which are appropriate tests for the self-adaptation mechanism of ES.

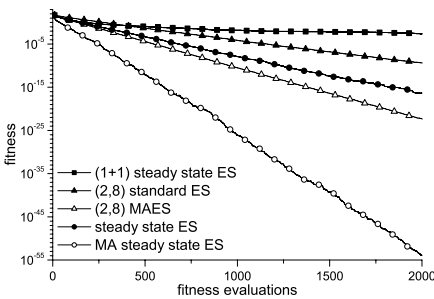


Fig. 7. Result for 10-dim. sphere function: fitness of best individual.

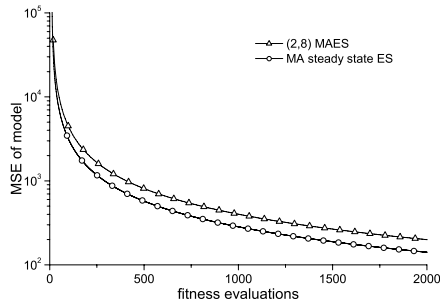


Fig. 8. MSE of model, for MA-ES vs MASS-ES.

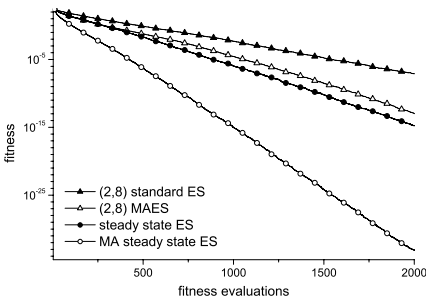


Fig. 9. Result for 10-dim. weighted sphere function: fitness of best individual.

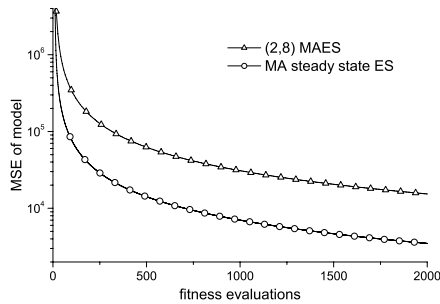


Fig. 10. MSE of model, for MA-ES vs MASS-ES.

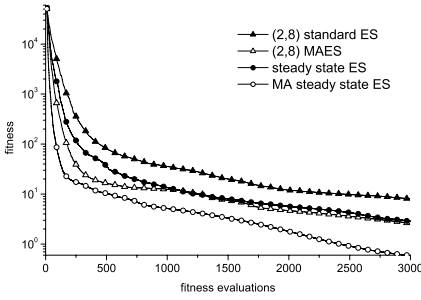


Fig. 11. Result for 10-dim. Rosenbrock function: fitness of best individual.

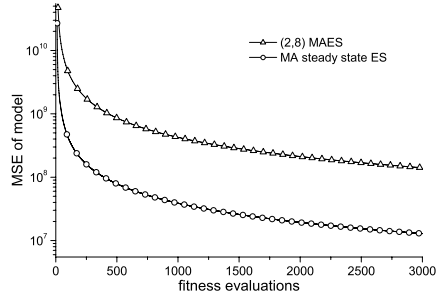


Fig. 12. MSE of model, for MA-ES vs MASS-ES.

In Figure 7 the standard (1+1) steady-state ES without median selection has the worst performance. In contrast steady-state ES ($\mu = 1$) with median selection performs much better and also better than the generation based standard ES due to the median selection.

This proves the effectiveness of median selection scheme by providing a self-adaption mechanism comparable to the one of a standard generation based (μ, λ) ES. For this reason standard (1+1) steady-state ES results are not compared later.

The support by the model results in an improvement of convergence both for standard ES and steady-state ES. But the relative enhancement due to the model is in case of the steady-state strategy much higher than in the standard ES case. MASS-ES yields orders of magnitude better solutions than all other strategies (see figure 7 and 9).

This amazing result can be explained by the smaller model error (MSE) of the model for the steady-state ES compared to standard generation based ES (see figure 8, 10), due to the higher model update frequency of the steady-state strategy. The MSE has a hyperbolic appearance, because of the decreasing absolute prediction error of the model with increasing N .

The Rosenbrock function (A.3) is nonlinear, continuous and not symmetric. It is a very popular test function and has a very hard to find global optimum. Figure 11 shows again the superiority of the MASS-ES to the other strategies. It reaches the best fitness value and has a remarkably higher convergence speed at the beginning of the optimization process. Also the MSE of the steady-state strategy is smaller than the standard ES one and is responsible for the better performance of MASS-ES.

5.2 MULTIMODAL TEST FUNCTIONS

Multimodal functions evoke hills and valleys which are misleading local optima. A simple optimization algorithm like hill-climbing would get stuck in a local minimum. For multimodal functions a population size of $\mu = 10$ for steady state

is selected, in order not to converge too fast into a local optimum.

The Step function (A.4) consists of flat plateaus with slope=0 in an underlying continuous function. It is hard to find the global optimum because minor changes of the object variables do not affect the fitness.

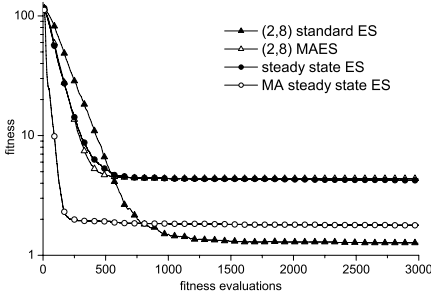


Fig. 13. Result for 10-dim. Step function: fitness of best individual.

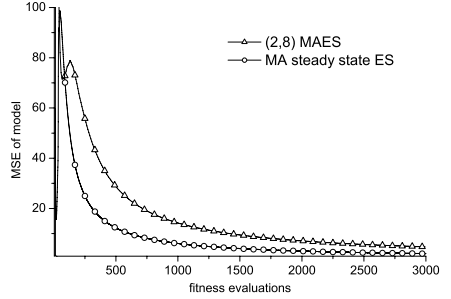


Fig. 14. MSE of model, for MA-ES vs MASS-ES.

Here the application of modeling results in an acceleration of the convergence velocity at the beginning (first 500 fitness evaluations) of the optimization process (figure 13). Again the benefit for the steady-state case is bigger and the MSE lower (figure 14). However, both modeling approaches stall in local optima and standard ES achieves a little better fitness values than MASS-ES. Rastrigin's (A.5) and Ackley's (A.6) are very bumpy symmetric test functions. Ackley's function has a global optimum with very strong local features. It is the only here analyzed test function, whose MSE increases temporary and is for MASS-ES higher than for MA-ES (figure 18). This observation could be explained by the very high complexity of Ackley's function. However, MASS-ES reaches the best fitness values (figure 15 and 17), although it has a lower convergence speed at the beginning than the standard ES strategy for Ackley's function. This observation is in contrast to results of Keane [3], whose modeling approach stalls in situations where the optimum has strong local features.

6 CONCLUSIONS

We applied metamodeling with RBF networks to standard generation based ES and a steady-state ES with median selection scheme. The strategies were tested in extensive simulations on 6 high-dimensional test functions. Metamodeling enhances the performance in both cases, due to the effective preselection of only very promising individuals by fitness value prediction.

However, the improvement for steady-state is much better than for the standard

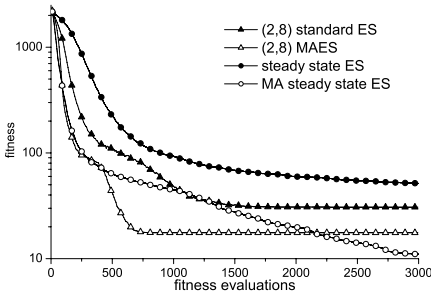


Fig. 15. Result for 10-dim. Rastrigin function: fitness of best individual.

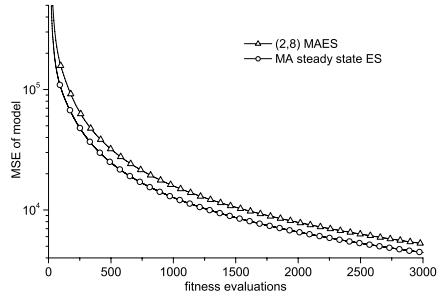


Fig. 16. MSE of model, for MA-ES vs MASS-ES.

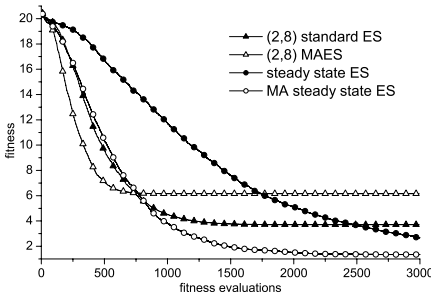


Fig. 17. Result for 10-dim Ackley's function: fitness of best individual.

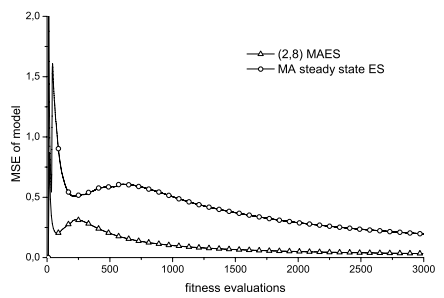


Fig. 18. MSE of model, for MA-ES vs MASS-ES.

ES strategies. This can be explained by its superior model quality due to its higher model update frequency. Therefore steady-state strategies are more suitable for metamodeling assistance than generation based strategies.

The new model assisted steady-state ES with median selection achieves equally good or much better results compared to standard ES, MA-ES or standard steady-state ES. It outperforms other strategies especially at unimodal smooth test functions, whereby it yields orders of magnitude better solutions. For multimodal test functions it converges faster at the beginning and achieves equally good or better fitness values compared to the other strategies.

For further work it is planned to develop a mechanism which controls the impact of the model on the ES. The optimization time can be shortened by asynchronous parallel fitness evaluation in a multiprocessor environment. Moreover a comparison of different modeling techniques would be interesting.

A TEST FUNCTIONS

A.1 Sphere function

$$f_{Sphere}(\mathbf{x}) = \sum_{i=1}^n x_i^2 \quad (4)$$

$$-5.12 \leq x_i \leq 5.12 ; n = 10 ; \min(f_{Sphere}) = f_{Sphere}(0, \dots, 0) = 0$$

A.2 Weighted Sphere function

$$f_{WSphere}(\mathbf{x}) = \sum_{i=1}^n i \cdot x_i^2 \quad (5)$$

$$-5.12 \leq x_i \leq 5.12 ; n = 10 ; \min(f_{WSphere}) = f_{WSphere}(0, \dots, 0) = 0$$

A.3 Generalized Rosenbrock's function

$$f_{Rosen}(\mathbf{x}) = \sum_{i=1}^n (100 \cdot (x_{i+1} - x_i)^2 + (x_i - 1)^2) \quad (6)$$

$$-5.12 \leq x_i \leq 5.12 ; n = 10 ; \min(f_{Rosen}) = f_{Rosen}(1, \dots, 1) = 0$$

A.4 Step function

$$f_{Step}(\mathbf{x}) = \sum_{i=1}^n \lfloor x_i \rfloor \quad (7)$$

$$-5.12 \leq x_i \leq 5.12 ; n = 10 ; \min(f_{Step}) = f_{Step}(0, \dots, 0) = 0$$

A.5 Rastrigin's function

$$f_{Rastrigin}(\mathbf{x}) = 10 \cdot n + \sum_{i=1}^n (x_i^2 - \cos(2\pi x_i)) \quad (8)$$

$$-32.768 \leq x_i \leq 32.768 ; n = 10 ; \min(f_{Rastrigin}) = f_{Rastrigin}(0, \dots, 0) = 0$$

A.6 Ackley's function

$$f_{Ackley}(\mathbf{x}) = 20 + e - 20 \exp \left(-0.2 \cdot \sqrt{\frac{1}{n} \cdot \sum_{i=1}^n x_i^2} \right) - \exp \left(\frac{1}{n} \sum_{i=1}^n \cos(2\pi x_i) \right) \quad (9)$$

$$-32.768 \leq x_i \leq 32.768 ; n = 10 ; \min(f_{Ackley}) = f_{Ackley}(0, \dots, 0) = 0$$

References

1. C.M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, 1995.
2. M. EL-Beltagy, P. Nair, and A.Keane. Metamodeling techniques for evolutionary optimization of computationally expensive problems: promises and limitations. In *GECCO 2000 Proceedings of Genetic and Evolutionary Computation Conference*, pages 196–203, 1999.
3. M. A. El-Beltagy and A. J. Keane. Evolutionary optimization for computationally expensive problems using gaussian processes. In CSREA Press Hamid Arabnia, editor, *Proc. Int. Conf. on Artificial Intelligence IC-AI'2001*, pages 708–714, 2001.
4. M. Emmerich, A. Giotis, M.Multlu Özdemir, T. Bäck, and K. Giannakoglou. Metamodel-assisted evolution strategies. In *Parallel Problem Solving from Nature VII*, pages 362–370, 2002.
5. N. Hansen and A. Ostermeier. Convergence properties of evolution strategies with the derandomized covariance matrix adaptation: The $(\mu/\mu_i, \lambda)$ -cma-es. In *5th European Congress on Intelligent Techniques and Soft Computing*, pages 650–654, 1997.
6. Y. Jin, M. Olhofer, and B. Sendhoff. On evolutionary optimisation with approximate fitness functions. In *GECCO 2000 Proceedings of Genetic and Evolutionary Computation Conference*, pages 786–793, 2000.
7. Y. Jin, M. Olhofer, and B. Sendhoff. A framework for evolutionary optimization with approximate fitness functions. *IEEE Transactions on Evolutionary Computation*. March 2002 (in press). (ISSN: 1089-778X), 2002.
8. Y. Jin and B. Sendhoff. Fitness approximation in evolutionary computing - a survey. In *GECCO 2002 Proceedings of Genetic and Evolutionary Computation Conference*, pages 1105–1111, 2002.
9. A. Ratle. Accelerating the convergence of evolutionary algorithms by fitness landscape approximation. In A. Eiben et al, editor, *Parallel Problem Solving from Nature V*, pages 87–96, 1998.
10. I. Rechenberg. *Evolutionsstrategie '94*. frommann-holzboog, Stuttgart, 1994.
11. H.-P. Schwefel. *Numerische Optimierung von Computer-Modellen mittels der Evolutionsstrategie*. Birkhäuser, Basel, 1977.
12. H.-P. Schwefel. Natural evolution and collective optimum-seeking. In PA. Sydow, editor, *Computational Systems Analysis: Topics and Trends*, pages 5–14, 1992.
13. J. Wakunda and A. Zell. A new selection scheme for steady-state evolution strategies. In Darell Whitley, David Goldberg, Erick Cantú-Paz, Lee Spector, Ian Parmee, and Hans-Georg Beyer, editors, *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2000*, pages 794–801, Las Vegas, Nevada, July 10-12 2000. Morgan Kaufmann Publishers, San Francisco, California.

On the Optimization of Monotone Polynomials by the (1+1) EA and Randomized Local Search

Ingo Wegener* and Carsten Witt*

FB Informatik, LS 2
Univ. Dortmund
44221 Dortmund, Germany
{wegener, witt}@ls2.cs.uni-dortmund.de

Abstract. Randomized search heuristics like evolutionary algorithms and simulated annealing find many applications, especially in situations where no full information on the problem instance is available. In order to understand how these heuristics work, it is necessary to analyze their behavior on classes of functions. Such an analysis is performed here for the class of monotone pseudo-boolean polynomials. Results depending on the degree and the number of terms of the polynomial are obtained. The class of monotone polynomials is of special interest since simple functions of this kind can have an image set of exponential size, improvements can increase the Hamming distance to the optimum and, in order to find a better search point, it can be necessary to search within a large plateau of search points with the same fitness value.

1 Introduction

Randomized search heuristics like random local search, simulated annealing, and all variants of evolutionary algorithms have many applications and practitioners report surprisingly good results. However, there are few theoretical papers on the design and analysis of randomized search heuristics.

In this paper, we investigate general randomized search heuristics, namely a random local search algorithm and a mutation-based evolutionary algorithm. It should be obvious that they do not improve heuristics with well-chosen problem-specific modules. Our motivation to investigate these algorithms is that such algorithms are used in many applications and that only an analysis will provide us with some knowledge to understand these algorithms better. This will give us the chance to improve these heuristics, to decide when to apply them, and also to teach them. The idea is to analyze randomized search heuristics for complexity-theoretical easy scenarios. One may hope that the heuristics behave similarly also on those functions which are “close” to the considered ones.

Each pseudo-boolean function $f: \{0, 1\}^n \rightarrow \mathbb{R}$ can be written uniquely as a polynomial

* Supported in part by the Deutsche Forschungsgemeinschaft as a part of the Collaborative Research Center “Computational Intelligence” (SFB 531).

$$f(x) = \sum_{A \subseteq \{1, \dots, n\}} w_A \cdot \prod_{i \in A} x_i .$$

The degree $d := \max\{|A| \mid w_A \neq 0\}$ and the number N of non-vanishing terms $w_A \neq 0$ are parameters describing properties of f . Note that the value of N can vary if we exchange the meanings of ones and zeros for some variables, i. e., replace some x_i by their negations, $1 - x_i$. For instance, the product of all $(1 - x_i)$ has the maximal number of 2^n non-vanishing terms but only one non-vanishing term if we replace x_i by $y_i := 1 - x_i$. The parameter N will be relevant in some upper bounds presented in this paper. However, all search heuristics that we will consider treat zeros and ones in the same way. Therefore, we may silently assume that in the polynomial representation of some monotone polynomial f , variables x_i have possibly been replaced by their negations $1 - x_i$ in such a way that N takes its minimum value. Droste, Jansen and Wegener (2) have analyzed evolutionary algorithms on polynomials of degree $d = 1$ and Wegener and Witt (14) have investigated polynomials of degree $d = 2$. The last case is known to be NP-hard in general. A simpler subcase is the case of monotone polynomials where f can be written as a polynomial with non-negative weights on some variable set $z_1, \dots, z_n$, where $z_i = x_i$ or $z_i = 1 - x_i$. In the first case, the function is monotone increasing with respect to x_i and, in the second case, monotone decreasing. In this paper, we investigate randomized search heuristics for the maximization of monotone polynomials of degree bounded by some parameter d . Since all considered heuristics treat zeros and ones in the same way, we can restrict our analysis to monotone increasing polynomials where $z_i = x_i$ for all i . The results hold for all monotone polynomials.

The investigation of polynomials of small degree is well motivated since many problems lead to polynomials of bounded degree. Monotonicity is a restriction that simplifies the problem. However, in the general setting it is unknown whether the function is monotone increasing or decreasing with respect to x_i .

Evolutionary algorithms are general problem solvers that eventually optimize each $f: \{0, 1\}^n \rightarrow \mathbb{R}$. We conjecture that our arguments would not lead to better upper bounds when we allow large populations and/or crossover. Indeed, it seems to be the case that they increase the optimization time moderately. Therefore, we investigate a simple standard evolutionary algorithm (EA), which is mutation-based and works with population size 1. This so-called (1+1) EA consists of an initialization step and an infinite loop.

(1+1) EA

Initialization: Choose $a \in \{0, 1\}^n$ randomly.

Loop: The loop consists of a mutation and a selection step.

Mutation: For each position i , decide independently whether a_i should be flipped (replaced by $1 - a_i$). The flipping probability equals $1/n$.

Selection: Replace a by a' iff $f(a') \geq f(a)$.

The advantage of the (1+1) EA is that each point can be created from each point with positive probability but steps flipping only a few bits are preferred. Therefore, it is not necessary (as in simulated annealing) to accept worsenings.

Random local search (RLS) flips only one bit per step. The algorithm is also called *random mutation hill-climbing* (Mitchell, Holland and Forrest 10).

RLS works like the (1+1) EA with a different mutation operator.

Mutation: Choose $i \in \{1, \dots, n\}$ randomly and flip a_i .

RLS cannot escape from local optima, where the local neighborhood is the Hamming ball with distance 1. However, it can optimize monotone polynomials (by our assumption the optimum is 1^n) since, for each a , there exists a sequence $a_0 = a, a_1, \dots, a_m = 1^n$ such that $m \leq n$, the Hamming distance of a_i and a_{i+1} equals 1, and $f(a_0) \leq f(a_1) \leq \dots \leq f(a_m)$. The analysis of RLS will be much easier than the analysis of the (1+1) EA; however, only the (1+1) EA is a general problem solver. The difficulty is that accepted steps can increase the number of zeros and the Hamming distance to the optimum. The problem for all heuristics is that it can be necessary to change many bits (up to d) until one finds a search point with larger fitness.

We have to discuss how we analyze RLS and the (1+1) EA, which are defined as infinite loops. In applications, we need a stopping criterion; however, this is not the essential problem. Hence, we are interested in the random optimization time X_f , defined as the minimum time step t where an optimal search point is created. Its mean value $E(X_f)$ is called the expected optimization time and $\text{Prob}(X_f \leq t)$ describes the success probability within t steps.

We present monotone polynomials of degree d where the expected optimization time equals $\Theta((n/d) \cdot \log(n/d + 1) \cdot 2^d)$ for RLS and the (1+1) EA, and we believe that the upper bound holds for all monotone polynomials. This can be proved for RLS, but our best bound for the (1+1) EA is worse and depends on N . For this reason, we also investigate a class of algorithms that bridge the difference between RLS and the (1+1) EA. The first idea is to reduce the mutation probability $1/n$ of the (1+1) EA. However, then we increase the probability of useless steps flipping no bit. Hence, we guarantee that at least one bit is flipped. We call the new algorithm RLS_p since it is a modification of RLS.

RLS_p works like the (1+1) EA and RLS with a different mutation operator.

Mutation: Choose $i \in \{1, \dots, n\}$ randomly and flip a_i . For each $j \neq i$, flip a_j independently of the other positions with probability p .

Obviously, RLS_p equals RLS for $p = 0$. For $p = 1/n$, RLS_p is close to the (1+1) EA, but omits steps without flipped bit. Hence, we investigate RLS_p only for $0 \leq p \leq 1/n$ and try to maximize p such that we can prove the upper bound $O((n/d) \cdot \log(n/d + 1) \cdot 2^d)$ on the expected optimization time of RLS_p on monotone polynomials.

The paper is structured as follows. Search heuristics with population size 1 lead to a Markov chain on $\{0, 1\}^n$. Therefore, we have developed some results on such Markov chains. The results are presented without proof in an appendix. In Sect. 2, we investigate the very special case of monomials, i. e., monotone polynomials where $N = 1$. These results are crucial since we later consider how long it takes to maximize a special monomial in the presence of many other monomials in the polynomial. In the Sections 3, 5, and 6, we prove upper bounds on the

expected optimization time of the algorithms RLS, RLS_p and the (1+1) EA on monotone polynomials. In Sect. 4, we present a worst-case monotone polynomial for RLS, which is conjectured to also be a worst-case monotone polynomial for RLS_p and the (1+1) EA. We finish with some conclusions. Some preliminary ideas of this paper have been given in a survey (Wegener 13).

2 The Optimization of Monomials

Because of the symmetry of all considered algorithms with respect to the bit positions and the role of zeros and ones, we can investigate w.l.o.g. the monomial $m(x) = x_1 \cdots x_d$. The following result has been proved by Garnier, Kallel and Schoenauer (4) for the special case $d = n$ and for the algorithms RLS_0 and (1+1) EA. We omit the proof of our generalizations.

Theorem 1. *The algorithms RLS_p , $p \leq 1/n$, and (1+1) EA in their pure form and under the condition of omitting all steps flipping more than two bits optimize monomials of degree d in an expected time of $\Theta((n/d) \cdot 2^d)$. The upper bounds also hold if the initialization is replaced by the deterministic choice of any $a \in \{0, 1\}^n$.*

3 On the Analysis of Random Local Search

The random local search algorithm, RLS, is easy to analyze since it flips one bit per step. This implies that activated monomials, i. e., monomials where all bits are 1, never get passive again.

Theorem 2. *The expected optimization time of RLS on a monotone polynomial of degree d is bounded by $O((n/d) \cdot \log(n/d + 1) \cdot 2^d)$.*

Sketch of proof. First, we investigate the case of polynomials with pairwise non-overlapping monomials, i. e., monomials that do not share variables. For each monomial of degree i , the probability of activating it in $O(2^i)$ steps is at least $1/2$ (see Theorem 1) if we count only steps flipping bits of the monomial. Now the arguments for proving the famous Coupon Collector's Theorem (see Motwani and Raghavan 11) can be applied to obtain the result.

In the general case, we choose a maximal set M_1 of pairwise non-overlapping monomials and consider the time T_1 until all monomials of M_1 are activated. The existence of further monotone monomials can only decrease the time for activating the monomials of M_1 . Here the property of monotonicity is essential. Hence, by the considerations above, $E(T_1)$ is bounded by $O((n/d) \cdot \log(n/d + 1) \cdot 2^d)$. The key observation is that afterwards each passive monomial contains at least one variable that is shared by an active monomial and, therefore, is fixed to 1. Hence, we are essentially in the situation of monomials whose degree is bounded by $d - 1$. This argument can be iterated and we obtain an upper bound on the expected optimization time, which is the sum of all $O((n/i) \cdot \log(n/i + 1) \cdot 2^i)$, $1 \leq i \leq d$. Simple calculations show that this sum is only by a constant factor larger than the term for $i = d$. This proves the theorem. $\square$

4 Royal Roads as a Worst-Case Example

It is interesting that the *royal road functions* RR_d introduced by Mitchell, Forrest and Holland (9) are the most difficult monotone polynomials for RLS and presumably also for RLS_p and the (1+1) EA. The function RR_d is defined for $n = kd$ by

$$\text{RR}_d(x) = \sum_{i=0}^{k-1} x_{id+1} \cdots x_{id+d} ,$$

or the number of blocks of length d containing ones only. Theorem 2 contains an upper bound of $O((n/d) \cdot \log(n/d + 1) \cdot 2^d)$ for the expected optimization time of RLS on RR_d , and this can be easily generalized to RLS_p , where $p \leq 1/n$, and the (1+1) EA. The result for RLS was also shown by Mitchell, Holland and Forrest (10). The mentioned upper bound disproved the conjecture that RR_d are royal roads for the crossover operator. Real royal roads have been presented only recently by Jansen and Wegener (7,8). Here we prove matching lower bounds on the expected optimization time for RR_d . First, we investigate RLS and, afterwards, we transfer the results to RLS_p and the (1+1) EA.

Theorem 3. *The probability that RLS has optimized the function RR_d within $(n/d) \cdot \log(n/d) \cdot 2^{d-5}$ steps is $o(1)$ (convergent to 0) if $d = o(n)$. The expected optimization time of RLS on RR_d equals $\Theta((n/d) \cdot \log(n/d + 1) \cdot 2^d)$.*

Sketch of proof. We only have to prove the lower bounds. For $d = \Theta(n)$, the result has been proved by Droste, Jansen, Tinnefeld and Wegener (1) for all considered algorithms. For $d = O(1)$, the bounds follow easily by considering the time until each bit initialized as 0 has flipped once (again the Coupon Collector's Theorem).

In the following, we assume that $d = \omega(1)$ and $d = o(n)$. First, we investigate the essential steps for a single monomial m , i.e., those steps flipping a bit of m . Let τ be the random number of essential steps until m is activated. Garnier, Kallel and Schoenauer (4) have proved that this process is essentially memoryless. More precisely

$$|\text{Prob}(\tau \geq t) - \text{Prob}(\tau \geq t + t' \mid \tau \geq t')| = O(1/d) ,$$

and $\text{Prob}(\tau \geq t)$ is approximately $1 - e^{-t}$. Hence, since $d = \omega(1)$, we have $\text{Prob}(\tau \leq 2^{d-1} + t' \mid \tau \geq t') \leq 1/2$ for all t' .

The next idea is that all monomials are affected by essentially the same number of steps. However, many steps for one monomial imply less steps for the other monomials. We partition the $k \cdot (\log k) \cdot 2^{d-5}$ steps into $(\log k)/4$ phases of length $k \cdot 2^{d-3}$ each. Let p_i be the random number of passive monomials after phase i . We claim that the following events all have an exponentially small probability with respect to $k^{1/4}$: the event $p_0 < k/2$ and the events $p_i < p_{i-1}/8$. Hence, the probability that none of these events happens is still $1 - o(1)$. This implies the existence of at least $p_0 \cdot (1/8)^{(\log k)/4} \geq k^{1/4}/2$ passive monomials at the end of the last phase implying that RR_d is not optimized.

The expected value of p_0 equals $k \cdot (1 - 2^{-d})$, and, therefore, the probability of the event $p_0 < k/2$ can be estimated by Chernoff bounds. If $p_j \geq p_{j-1}/8$ for all $j < i$, there are at least $k^{1/4}/2$ passive monomials at the end of phase $i - 1$. The expected number of steps essential for one of the passive monomials in phase i equals $p_{i-1} \cdot 2^{d-3}$, and the probability that this number is less than $p_{i-1} \cdot 2^{d-2}$ is exponentially close to 1. By the pigeon-hole principle, there are at most $p_{i-1}/2$ monomials with at least 2^{d-1} essential steps each. Pessimistically, we assume all these monomials to become active in phase i . We have proved before that each other monomial activates with probability at most $1/2$. By Chernoff bounds, the probability of activating at least $3/4$ of these and altogether at least $7/8$ of the passive monomials is exponentially small. This proves the theorem. $\square$

Theorem 4. *For each $\varepsilon > 0$, the probability that the (1+1) EA is on RR_d by a factor of $1 + \varepsilon$ faster than RLS is $O(1/n)$. The same holds for RLS_p , $p \leq 1/n$, and the factor $2 + \varepsilon$.*

Sketch of proof. We prove the result on the (1+1) EA by replacing the (1+1) EA by a faster algorithm $(1+1)^*$ EA and comparing the faster algorithm with RLS. A step of the $(1+1)^*$ EA works as follows. First, the number k of flipped bits is chosen according to the same distribution as for the (1+1) EA. Then the $(1+1)^*$ EA flips a random subset of k bits. This can be realized as follows. In each step, one random bit is flipped until one obtains a point of Hamming distance k to the given one. Now the new search point of the $(1+1)^*$ EA is obtained as follows. The selection procedure of RLS is applied after each step. This implies by the properties of the royal road functions that we obtain a search point a^* compared to the search point a of the (1+1) EA such that $a \leq a^*$ according to the componentwise partial order. This implies that the $(1+1)^*$ EA reaches the optimal string 1^n no later than the (1+1) EA.

However, the $(1+1)^*$ EA chooses flipped bits as RLS, and it uses the same selection procedure. The difference is that the $(1+1)^*$ EA sometimes simulates many steps of RLS in one step, while the (1+1) EA flips on average one bit per step. It is easy to see that we have to consider $t = \Omega(n)$ steps. Then it is for each $\gamma > 0$ very likely that the $(1+1)^*$ EA flips not more than $(1 + \gamma)t$ bits within t steps. Moreover, with high probability the number of flipped bits is bounded by δn in each step, $\delta > 0$ a constant. Let a be the starting point of the simulation of one step. The probability of increasing the Hamming distance to a with the next flipped bit is at least $1 - \delta$. Hence, with large probability we have among t steps an overhead of $(1 - 3\delta)t$ distance-increasing steps. Hence, the probability that $(1 + \gamma) \cdot t / (1 - 3\delta)$ steps of RLS do not suffice to simulate t steps of the $(1+1)^*$ EA is exponentially small. Choosing γ and δ such that $(1 + \gamma)/(1 - 3\delta) = 1 + \varepsilon$, we are done. The statement on RLS_p follows in the same way taking into account that RLS_p , $p \leq 1/n$, flips on average not more than two bits per step. $\square$

5 On the Analysis of RLS_p

In contrast to RLS, RLS_p with $p > 0$ can deactivate monomials by simultaneously activating other monomials. Even the analysis of the time until a single

monomial is activated becomes much more difficult. Steps where two bits of the monomial flip from 0 to 1 and only one bit flips from 1 to 0 may decrease the fitness and be rejected. Hence, we do not obtain simple Markov chains as in the case of RLS or in the case of single monomials.

We can rule out the event of three or more flipped bits contained in the same monomial if its degree is not too large, more precisely $d = O(\log n)$. This makes sense since Theorems 3 and 4 have shown that we cannot obtain polynomial upper bounds otherwise. To analyze the optimization process of RLS_p on a monotone polynomial, we first consider some fixed, passive monomial and estimate the time until it becomes active for the first time. The best possible bound $O((n/d) \cdot 2^d)$ can be proved if p is small enough. Afterwards, we apply this result in order to bound the expected optimization time on the monotone polynomial. The bound we obtain here is close to the lower bound from Theorem 4.

Lemma 1. *Let f be a monotone polynomial of degree $d \leq c \log n$ and let m be one of its monomials. There is a constant $\alpha > 0$ such that RLS_p with $p = \min\{1/n, \alpha/(n^{c/2} \log n)\}$ activates m in an expected time of $O((n/d) \cdot 2^d)$ steps.*

Sketch of proof. The idea is to prove that RLS_p activates m with a constant probability $\varepsilon > 0$ within a phase of $c' \cdot (n/d) \cdot 2^d$ steps, for some constant c' . Since our analysis does not depend on the starting point, this implies an upper bound $c' \cdot (n/d) \cdot 2^d / \varepsilon$ on the expected time to activate m . We assume w.l.o.g. that $m = x_1 \cdots x_d$ and call it the *prefix* of the search point.

We bound the probability of three events we consider as a failure. The first one is that we have a step flipping at least three prefix bits in the phase. The second one is that, under the condition that the first type of failure does not happen, we do not create a search point where m is active in the phase. The third one occurs if the first search point in which m is active is not accepted. If none of the failures occurs, m is obviously activated.

The first and third type of failure can be handled by standard techniques. A simple calculation shows that the first type of failure occurs with probability at most $d^3 p^2 / n$ in one step. Multiplying by the length of the phase, we obtain a failure probability bounded by a constant if α is small enough. For the third failure type it is necessary that at least one of the suffix bits $x_{d+1}, \dots, x_n$ flips. Since we assume m to be activated in the considered step, the related conditional probability of not flipping a suffix bit can be bounded below by the constant $1/(2e)$. All this holds also under the condition that the first two types of failure do not happen.

For the second type of failure, we apply the techniques developed in the appendix by comparing the Markov chains Y_0 and Y_1 . Y_0 equals RLS_p^* , namely RLS_p on the monomial m , where the condition holds that no step flips more than two bits of the prefix. Y_1 equals RLS_p^* on the monotone polynomial f , which again equals RLS_p under the condition that no step flips more than two prefix bits. Both Markov chains are investigated on the compressed state space $D = \{0, \dots, d\}$ representing the number of 1-bits in the prefix. We can ignore the fact that the Markov chain Y_1 is not time-homogeneous by deriving bounds on its transition probabilities that hold for all search points. We denote these bounds

still by $P_1(i, j)$. Then the following conditions for Lemma 7 imply the bound $O((n/d) \cdot 2^d)$ of the lemma. (See also Definitions 1, 2 and 3 in the appendix.)

1. Y_1 has a relative advantage to Y_0 for c -values such that $c_{\min} \geq 1/(2e)$,
2. Y_0 has a $(2e - 1)$ -advantage, and
3. $E(\tau_0^i) = O((n/d) \cdot 2^d)$ for all i .

The third claim is shown in Theorem 1. The second one follows from Lemma 4 since $d \leq (n - 1)/(4e + 1)$ if n is large enough. For the first claim, recall that at most two prefix bits flip. Now Definition 3 implies that we have to consider $c(i, j) = P_1(i, j)/P_0(i, j)$ for $j \in \{i - 2, i - 1, i + 1, i + 2\}$ and to prove that

1. $1/(2e) \leq c(i, i + 1) \leq 1$
2. $c(i, i + 2) \geq c(i, i + 1)$,
3. $c(i, i - 1) \leq c(i, i + 1)$, and
4. $c(i, i - 2) \leq c(i, i + 1)$ (or even $c(i, i - 2) \leq c(i, i - 1)$).

The inequality $c(i, i + 1) \leq 1$ holds since RLS_p^* on m accepts each new string as long as the optimum is not found. The bound $c(i, i + 1) \geq 1/(2e)$ follows from the fact that RLS_p^* on the monotone polynomial f accepts a step where one prefix bit flips from 0 to 1 and no suffix bit flips.

For the remaining inequalities, observe that $c(i, j)$ is the conditional probability of RLS_p^* accepting (for f) a search point x given that x contains j prefix ones and has been created from a string with i prefix ones. The idea is to condition these probabilities even further by considering a fixed change of the suffix bits. Let the suffix change from c to c' , and let b be a prefix containing i ones. If RLS_p^* accepts the string (b', c') , where b' is obtained from b by flipping a zero to one, then RLS_p^* also accepts (b'', c') , where b'' is obtained from b' by flipping another zero. Estimating the number of such strings (b'', c') leads to $c(i, i + 2) \geq c(i, i + 1)$. By a dual argument, we prove $c(i, i - 2) \leq c(i, i - 1)$. Finally, the inequality $c(i, i + 1) \geq c(i, i - 1)$ follows from the following observation. If there is at least one string (b', c') that is not accepted and where b' has been obtained by flipping a zero of b , then all strings (b'', c') , where b'' has been obtained by flipping a one of b , are also rejected. This completes the proof. $\square$

Theorem 5. *The expected optimization time of RLS_p on a monotone polynomial f of degree $d \leq c \log n$ is bounded above by $O((n^2/d) \cdot 2^d)$ if $0 < p \leq \min\{(1 - \gamma)/(2dn), \alpha/(n^{c/2} \cdot \log n)\}$ for the constant α from Lemma 1 and each constant $\gamma > 0$.*

Sketch of proof. The optimization process is not reflected by the f -value of the current search point. An f -value of v can be due to a single monomial of degree 1 or to many monomials of large degree. Instead, we count the number of *essential ones* (with respect to f). A 1-entry of a search point is called essential if it is contained in an activated monomial of f . All other 1-entries may flip to 0 without decreasing the f -value and are therefore called inessential. 0-entries are always called inessential. An essential one can only become inessential if simultaneously some monomial is activated. A step where a monomial is activated is called

essential. By Lemma 1, it suffices to prove an $O(n)$ bound on the expected number of essential steps.

To prove this bound, we apply Lemma 8, an approach sometimes called *drift analysis* (see Hajek 5; Sasaki and Hajek 12; He and Yao 6). Let X_i be the number of essential ones after the i -th essential step, i. e., X_0 is the number of essential ones after initialization. Let $D_0 = X_0$ and $D_i = X_i - X_{i-1}$ for $i \geq 1$. Then we are interested in τ , the minimal i where $D_0 + D_1 + \dots + D_i = n$. Some conditions of Lemma 8 are verified easily. We have $|D_i| \leq n$ and $E(\tau) < \infty$ since there is always a probability of at least p^n to create the optimal string. If we can prove that $E(D_i \mid \tau \geq i) \geq \varepsilon$ for some $\varepsilon > 0$, Lemma 8 implies $E(\tau) = O(n)$.

At least one monomial is activated in an essential step, i. e., at least one bit turns from inessential into essential. We have to bound the expected number of bits turning from essential into inessential. Since the assumption that the new search point is accepted only decreases this number, we consider the number of flipped ones under the condition that a 0-bit is flipped.

Let Y be the random number of additional bits flipped by RLS_p under the assumption that a specified bit (activating a monomial) flips. A lengthy calculation shows that $E(Y) \leq (1 - \varepsilon)/d$ for some $\varepsilon > 0$ since $p \leq (1 - \gamma)/(2dn)$. The problem is that given $Y = i$, more than i bits may become inessential. Therefore, we upper bound the expected number of bits turning from essential into inessential if $Y = i$. In the worst case, these i flipped bits contain essential ones. Since we do not take into account whether the new search point is accepted, each subset of size i of the essential ones has the same probability of being the flipped ones. We apply the accounting method on the random number L of essential ones becoming inessential if a random essential one flips. The idea is as follows. In order to make the essential one in bit j inessential, some essential one contained in all monomials that contain x_j flips. This leads to $E(L) \leq d$. Then we can show that by flipping i essential ones, we lose on average at most id essential ones. Since $E(Y) \leq (1 - \varepsilon)/d$, the expected number of essential ones becoming inessential is at most $1 - \varepsilon$. Since at least one bit gets essential, this implies $E(D_i \mid \tau \geq i) \geq \varepsilon$ and the theorem. $\square$

6 On the Analysis of the (1+1) EA

Since the (1+1) EA flips too many bits in a step, the bound of Theorem 5 cannot be transferred to the (1+1) EA, and we only obtain a bound depending on the parameter N here. However, a result corresponding to Lemma 1 can be proved.

Lemma 2. *Let f be a monotone polynomial of degree d , and let m be one of its monomials. There is a constant α such that the (1+1) EA activates m in an expected number of $O((n/d) \cdot 2^d)$ steps if $d \leq 2 \log n - 2 \log \log n - \alpha$.*

Sketch of proof. We follow the same structure as in the proof of Lemma 1 and need only few different arguments.

First, the probability of at least three flipped prefix bits in one step is bounded above by $d^3 n^{-3}/6$ for the (1+1) EA. Therefore, the probability that such a step

happens in a phase of length $c \cdot (n/d) \cdot 2^d$ for some constant c is still smaller than 1 by choosing α large enough. Second, the probability that no suffix bit flips is at least $(1 - 1/n)^{n-1} \geq 1/e$. Also Lemma 7 can be applied with a value $c_{\min} \geq 1/e$ and an $(e-1)$ -advantage of Y_0 . It is again possible to apply Theorem 1. Instead of Lemma 4, Lemma 3 is applied. Here it is sufficient that $d \leq (n-1)/e$. Finally, the argument that Y_1 has a relative advantage to Y_0 for c -values such that $c_{\min} \geq 1/e$ can be used in the same way here. $\square$

Theorem 6. *The expected optimization time of the (1+1) EA on a monotone polynomial with N monomials and degree $d \leq 2 \log n - 2 \log \log n - \alpha$ for the constant α from Lemma 2 is bounded above by $O(N \cdot (n/d) \cdot 2^d)$.*

Sketch of proof. Here we use the method of measuring the progress by fitness layers. Let the positive weights of the N monomials be sorted, i.e., $w_1 \geq \dots \geq w_N > 0$. We partition the search space $\{0, 1\}^n$ into $N + 1$ layers $L_0, \dots, L_N$, where

$$L_i = \{a \mid w_1 + \dots + w_i \leq f(a) < w_1 + \dots + w_{i+1}\}$$

for $i < N$, and L_N contains all optimal search points. Each layer L_i , $i < N$, is left at most once. Hence, it is sufficient to prove a bound of $O((n/d) \cdot 2^d)$ on the expected time to leave L_i . Let $a \in L_i$. Then there exists some $j \leq i + 1$ such that the monomial m_j corresponding to w_j is passive. By Lemma 2, the expected time until m_j is activated is bounded by $O((n/d) \cdot 2^d)$. We can bound the probability of not leaving L_i in the step activating m_j by $1 - e^{-1}$. The expected number of such phases is therefore bounded by e . $\square$

Conclusions

We have analyzed randomized search heuristics like random local search and a simple evolutionary algorithm on monotone polynomials. The conjecture is that all these algorithms optimize monotone polynomials of degree d in an expected number of $O((n/d) \cdot \log(n/d+1) \cdot 2^d)$ steps. It has been shown that some functions need that amount of time. Moreover, for random local search the bound has been verified. If the expected number of flipped bits per step is limited, a little weaker bound is proved. However, for the evolutionary algorithm only a bound depending on the number of monomials with non-zero weights has been obtained. Although there is room for improvement, the bounds and methods are a step to understand how randomized search heuristics work on simple problems.

References

- Droste, S., Jansen, T., Tinnefeld, K., Wegener, I.: A new framework for the valuation of algorithms for black-box optimization. In: Proc. of FOGA 7. (2002) 197–214. Final version of the proceedings to appear in 2003.
- Droste, S., Jansen, T., Wegener, I.: On the analysis of the (1+1) evolutionary algorithm. Theoretical Computer Science **276** (2002) 51–81

- Feller, W.: An Introduction to Probability Theory and its Applications. Wiley, New York (1971)
- Garnier, J., Kallel, L., Schoenauer, M.: Rigorous hitting times for binary mutations. *Evolutionary Computation* **7** (1999) 173–203
- Hajek, B.: Hitting-time and occupation-time bounds implied by drift analysis with applications. *Advances in Applied Probability* **14** (1982) 502–525
- He, J., Yao, X.: Drift analysis and average time complexity of evolutionary algorithms. *Artificial Intelligence* **127** (2001) 57–85
- Jansen, T., Wegener, I.: Real royal road functions – where crossover provably is essential. In: *Proc. of GECCO 2001*. (2001) 375–382
- Jansen, T., Wegener, I.: The analysis of evolutionary algorithms – a proof that crossover really can help. *Algorithmica* **34** (2002) 47–66
- Mitchell, M., Forrest, S., Holland, J.H.: The royal road for genetic algorithms: Fitness landscapes and GA performance. In Varella, F.J., Bourguine, P., eds.: *Proc. of the First European Conference on Artificial Life*, Paris, MIT Press (1992) 245–254
- Mitchell, M., Holland, J.H., Forrest, S.: When will a genetic algorithm outperform hill climbing. In Cowan, J.D., Tesauro, G., Alspector, J., eds.: *Advances in Neural Information Processing Systems*. Volume 6., Morgan Kaufmann (1994) 51–58
- Motwani, R., Raghavan, P.: *Randomized Algorithms*. Cambridge University Press (1995)
- Sasaki, G.H., Hajek, B.: The time complexity of maximum matching by simulated annealing. *Journal of the ACM* **35** (1988) 387–403
- Wegener, I.: Theoretical aspects of evolutionary algorithms (invited paper). In: *Proc. of ICALP 2001*. Number 2076 in LNCS (2001) 64–78
- Wegener, I., Witt, C.: On the analysis of a simple evolutionary algorithm on quadratic pseudo-boolean functions. To appear in *Journal of Discrete Algorithms* (2003).

A Some Results on Markov Chains

The behavior of randomized search heuristics on single monomials is of special interest. For a monomial of degree d , the current state can be identified with the number of ones among the variables of the monomial. This leads to the state space $D = \{0, \dots, d\}$. In order to obtain an ergodic Markov chain, we replace the selection operator by the selection operator that accepts each a' , i. e., a' always replaces a . Then we are interested in the minimal t such that in time step t the state d is reached. The transition probabilities for the (1+1) EA under the condition that each step changes the state number by at most 2 are denoted by $Q(i, j)$ and the corresponding transition probabilities for RLS_p by $R(i, j)$.

We prove that these Markov chains have the property that it is more likely to reach from i “higher” states than from $i - 1$. This intuitive notion is formalized as follows.

Definition 1. *Let $P(i, j)$ be the transition probabilities of a time-homogeneous Markov chain on $D = \{0, \dots, d\}$. The Markov chain has an ε -advantage, $\varepsilon \geq 0$, if for all $i \in \{0, \dots, d - 2\}$ the following properties hold.*

1. $P(i, j) \geq (1 + \varepsilon) \cdot P(i + 1, j)$ for $j \leq i$,
2. $P(i + 1, j) \geq (1 + \varepsilon) \cdot P(i, j)$ for $j > i$.

Lemma 3. *Let $\varepsilon \geq 0$ and $d \leq (n-1)/(1+\varepsilon)$. Then the Markov chain with transition probabilities $Q(i, j)$ has an ε -advantage.*

Lemma 4. *Let $\varepsilon \geq 0$ and $d \leq (n-1)/(3+2\varepsilon)$. Then the Markov chain with transition probabilities $R(i, j)$ has an ε -advantage.*

We are interested in the random variable τ^k describing for a time-homogeneous Markov chain Y on D with transition probabilities $P(i, j)$ the first point of time when it reaches state d if it starts in state k . If Y has a 0-advantage, it should be advantageous to start in a “higher state.” This is made precise in the following lemma.

Lemma 5. *Let $P(i, j)$ be the transition probabilities of a time-homogeneous Markov chain with 0-advantage on $D = \{0, \dots, d\}$. Then $\text{Prob}(\tau^i \geq t) \geq \text{Prob}(\tau^{i+1} \geq t)$ for $0 \leq i \leq d-1$ and each t . Moreover, $E(\tau^i) \geq E(\tau^{i+1})$.*

We compare different Markov chains. The complicated Markov chain Y_1 , describing a randomized search heuristic on a monotone polynomial with many terms, is compared with the simple Markov chain Y_0 , describing a randomized search heuristic on a single monomial. The idea is to use results for Y_0 to obtain results for Y_1 . We denote by τ_0^i and τ_1^i the random time to reach state d from state i with respect to Y_0 and Y_1 , respectively.

Definition 2. *Let $P_0(i, j)$ and $P_1(i, j)$ be the transition probabilities of the time-homogeneous Markov chains Y_0 and Y_1 on $D = \{0, \dots, d\}$. The Markov chain Y_1 has an advantage compared to Y_0 if $P_1(i, j) \geq P_0(i, j)$ for $j \geq i+1$ and $P_1(i, j) \leq P_0(i, j)$ for $j \leq i-1$.*

Lemma 6. *If Y_1 has an advantage compared to Y_0 and Y_0 has a 0-advantage, then $\text{Prob}(\tau_1^i \geq t) \leq \text{Prob}(\tau_0^i \geq t)$ and $E(\tau_1^i) \leq E(\tau_0^i)$.*

Finally, we apply Lemma 6 to compare two Markov chains Y_0 and Y_1 where weaker conditions hold than in Lemma 6. We compare Y_0 and Y_1 by parameters $c(i, j)$ such that $P_1(i, j) = c(i, j) \cdot P_0(i, j)$. This includes an arbitrary choice of $c(i, j)$ if $P_0(i, j) = P_1(i, j) = 0$.

Definition 3. *Let $P_0(i, j)$ and $P_1(i, j)$ be the transition probabilities of Y_0 and Y_1 such that $P_1(i, j) = c(i, j) \cdot P_0(i, j)$ for some $c(i, j)$. Then Y_1 has a relative advantage compared to Y_0 if $c(i, j) \geq c(i, i+1)$ for $j \geq i+1$, $c(i, j) \leq c(i, i+1)$ for $j \leq i-1$, and $0 < c(i, i+1) \leq 1$ for all $i \leq d-1$.*

Lemma 7. *If Y_1 has a relative advantage compared to Y_0 and Y_0 has a $(c_{\min}^{-1} - 1)$ -advantage, then $E(\tau_1^i) \leq c_{\min}^{-1} \cdot E(\tau_0^i)$ for $c_{\min} := \min\{c(i, i+1) \mid 0 \leq i \leq d-1\}$.*

The last result in this technical section is a generalization of Wald’s identity (see Feller 3). We do not claim to be the first to prove this result, but we have not found it in the literature.

Lemma 8. *Let D_i , $i \in \mathbb{N}$, be a sequence of random variables such that $|D_i| \leq c$ for a constant c . For $s > 0$, let τ_s be the minimal i where $D_1 + \dots + D_i = s$. If $E(\tau_s) < \infty$ and $E(D_i \mid \tau_s \geq i)$ is bounded below by a positive constant ℓ for all i where $\text{Prob}(\tau_s \geq i) > 0$, then $E(\tau_s) \leq s/\ell$.*

A Forest Representation for Evolutionary Algorithms Applied to Network Design

A.C.B. Delbem¹ and Andre de Carvalho¹

University of Sao Paulo – ICMC – USP, Sao Carlos – SP, Brazil,
{acbd, andre}@icmc.usp.br

Abstract. Network design involves several areas of engineering and science. Computer networks, electrical circuits, transportation problems, and phylogenetic trees are some examples. In general, these problems are *NP-Hard*. In order to deal with the complexity of these problems, several strategies have been proposed. Among them, approaches using evolutionary algorithms have achieved relevant results. However, the graph encoding is critical for the performance of such approaches in network design problems. Aiming to overcome this drawback, alternative representations of spanning trees have been developed. This article proposes an encoding for generation of spanning forests by evolutionary algorithms.

1 The Proposed Representation

The proposed forest representation basically consists of linear lists (which may be an array T) containing the tree **nodes** and their **depths**. The order the pairs $(node, depth)$ are disposed in the list is important and it must follow a preorder traversal. The forest representation is composed by the union of the encodings of all trees of a forest.

Two operators are proposed (named **operator 1** and **operator 2**) to generate new spanning forests using the node-depth encoding. Both operators generate a spanning forest F' of a graph G when they are applied to another spanning forest F of G . The results produced by the application of the operators are similar. The application of the operator 1 (or 2) to a forest is equivalent to transfer a subtree from a tree T_{from} to another tree T_{to} of the same forest. Applying operator 1, the root of the pruned subtree will be also the root of this subtree in its new tree (T_{to}). On the other hand, the transferred subtree will have a new root when applying operator 2.

In the description of the operator 1, we consider that two nodes were previously chosen: the prune node p , which indicates the root of the subtree of T_{from} to be transferred; and the adjacent node a , which is a node of a tree different from T_{from} . This node is also adjacent to p in G . An efficient procedure to determine such nodes are proposed in [1]. Besides, we assume that the node-depth representation was implemented using arrays and that the indices of p (i_p) and a (i_a), respectively, in the arrays T_{from} and T_{to} are also known. The operator 1 can be described by the following steps:

1. Determine the range (i_p-i_l) of indices in T_{from} corresponding to the subtree rooted at node p . Since we know i_p , we only need to find i_l ;
2. Copy the data in the range i_p-i_l from T_{from} into a temporary array T_{tmp} (corresponding to the subtree being transferred) and update the node depths using the depth of a .
3. Create an array T'_{to} (new tree) copying T_{to} and inserting T_{tmp} (pruned subtree) after the node a in T_{to} .
4. Construct an array T'_{from} copying T_{from} without the nodes of T_{tmp} .
5. Copy the forest F to F' exchanging the pointers to T_{from} and T_{to} for pointers to T'_{from} and T'_{to} , respectively.

The operator 2 requires a new root node r , besides the nodes p and a . The copy of the pruned subtree for the operator 2 can be divided in two steps: The first step corresponds to the step 2 for the operator 1 exchanging the range i_p-i_l by i_r-i_l . The array returned by this procedure is named T_{tmp1} . The second step considers the nodes in the path from r to p (i.e. $r_0, r_1, r_2, \dots, r_n$, where $r_0 = r$ and $r_n = p$) as roots of subtrees. The subtree rooted at r_1 contains the subtree rooted at r_0 . The subtree rooted at r_2 contains the subtree rooted at r_1 , and so on. The algorithm for the second step copies the subtrees rooted at r_j ($j = 1, \dots, n$) without the subtree rooted at r_{j-1} , updates the depths¹, and store the resultant subtrees in a temporary array T_{tmp2} . The step 3 of the operator 2 is equivalent to the same step of the operator 1, exchanging T_{tmp} for the concatenation of T_{tmp1} and T_{tmp2} ($T_{tmp} = [T_{tmp1}|T_{tmp2}]$). The steps 4 and 5 are equal in both operators.

2 Final Considerations

This proposal focuses on the production of spanning forests instead of trees (usually found in the literature). As consequence, the operator complexity depends on, for example, the size of the modified trees from F to F' , while the complexity of the operators found in the literature are usually functions of the number of nodes and/or edges in the underlying graph. The proposed operators do not require a graph G to be complete in order to produce only feasible spanning forests of G . Many practical problems do not involve complete graphs (in fact, several networks correspond to sparse graphs).

References

- [1] A. C. B. Delbem and Andre de Carvalho. New data structure for spanning forest operators for evolutionary algorithms. *Centro LatinoAmericano de Estudios en Informatica – CLEI 2002*, CD-ROM, 2002.

¹ The updated depth of node x is given by $T_{from}[i_x].depth - T_{from}[i_r].depth + T_{from}[i_r].depth - T_{from}[i_{r_j}].depth + depth\_of\_a + 1$.

Solving Three-Objective Optimization Problems Using Evolutionary Dynamic Weighted Aggregation: Results and Analysis

Yaochu Jin, Tatsuya Okabe, and Bernhard Sendhoff

Honda Research Institute Europe
Carl-Legien-Str. 30, 63073 Offenbach/Main, Germany
yaochu.jin@honda-ri.de

The main purposes of this paper is twofold. First, the evolutionary dynamic weighted aggregation (EDWA) [1] approaches are extended to the optimization of three-objective problems. Fig. 1 shows two example patterns for weight change. Through two three-objective test problems [2], the methods have shown to be effective. Theoretical analyses reveal that the success of the weighted aggregation based methods can largely be attributed to the following facts:

- The change of the weights is equivalent to the rotation of the Pareto front about the origin. All Pareto-optimal solutions, no matter whether they are located in the convex or concave region, are dynamically capturable. In contrast, classical analyses of the weighted aggregation method only consider the static stability of the Pareto-optimal solutions. Note that a dynamically capturable Pareto-optimal solution is not necessarily statically stable.
- Many multiobjective optimization problems exhibit the characteristics known as global convexity, which means that most Pareto-optimal solutions are concentrated in a small fraction of the parameter space. Furthermore, the solutions in the neighborhood in the fitness space are also in the neighborhood in the parameter space, and vice versa. This property is also known as the connectedness.
- The evolution strategies are able to carry out locally causal search. Once the population has reached any point on the Pareto front, the local search ability is very important for the algorithms to “scan” the Pareto front point by point smoothly. The resolution of the scanning is determined by the speed of the weight change.

In the second part of the paper, we show some additional nice properties of the Pareto-optimal solutions beyond the global convexity. It is empirically shown that the Pareto-optimal set exhibits surprising regularity and simplicity in the parameter space, which is very interesting and helpful. By taking advantage of such regularities, it is possible to build simple models from the obtained Pareto-optimal solutions for approximating the definition function. Such an approximate model can be of great significance in the following aspects.

- It allows to get more accurate, more complete Pareto solutions from the approximate solutions obtained by an optimizer. Fig. 2(a) shows the Pareto front obtained by the EDWA. The Pareto front reconstructed from the approximate definition function is presented in Fig. 2(b).

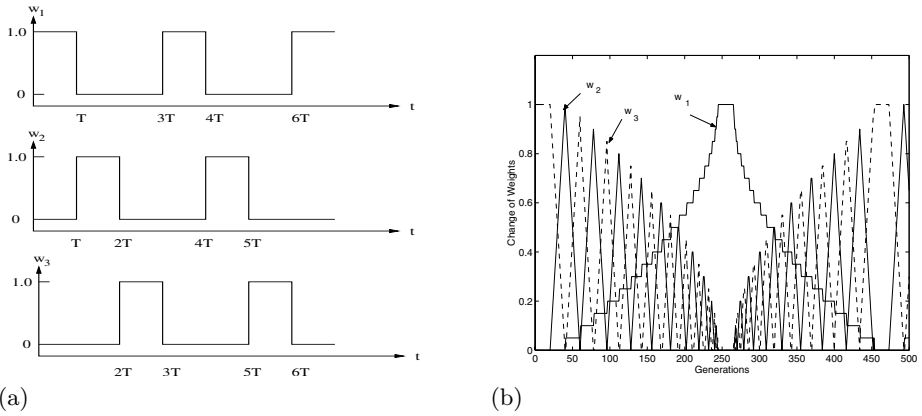


Fig. 1. An example of changing weights for (a) BWA and (b) DWA for solving three-objective optimization problems.

- It alleviates many difficulties in multiobjective optimization. If the whole Pareto front can be reconstructed from a few Pareto solutions, then many requirements on the optimizer can be alleviated, e.g., a uniform distribution is no more critical in approximating Pareto-optimal solutions.

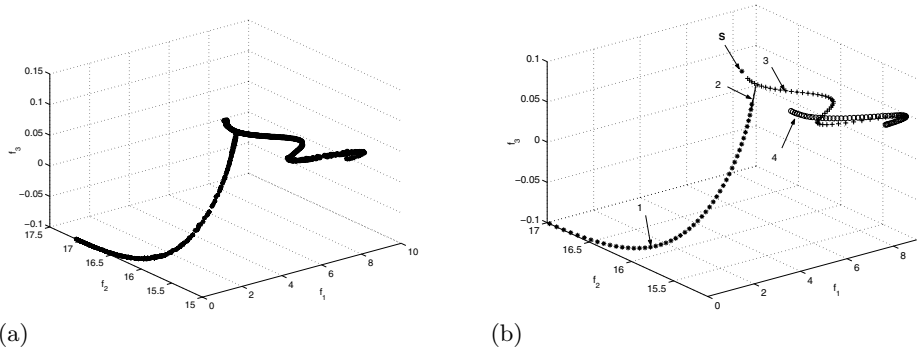


Fig. 2. (a) Obtained by the optimizer, (b) Reconstructed.

References

1. Y. Jin, M. Olhofer, and B. Sendhoff. Evolutionary dynamic weighted aggregation for multiobjective optimization: Why does it work and how? In *Genetic and Evolutionary Computation Conference*, pages 1042–1049, San Francisco, CA, 2001.
2. R. Viennet, C Fonteix, and I. Marc. Multicriteria optimization using genetic algorithms for determining a pareto set. *International Journal of Systems Science*, 27(2):255–260, 1996.

The Principle of Maximum Entropy-Based Two-Phase Optimization of Fuzzy Controller by Evolutionary Programming

Chi-Ho Lee, Ming Yuchi, Hyun Myung, and Jong-Hwan Kim

Dept. of Electrical Engineering and Computer Science,
Korea Advanced Institute of Science and Technology (KAIST),
373-1 Guseong-dong, Yuseong-gu, Daejeon, 305-701, Republic of Korea
{chiho,ycm,johkim}@vivaldi.kaist.ac.kr

Abstract. In this paper, a two-phase evolutionary optimization scheme is proposed for obtaining optimal structure of fuzzy control rules and their associated weights, using evolutionary programming (EP) and the principle of maximum entropy (PME) based on the previous research [1].

1 Two-Phase Evolutionary Optimization

A fuzzy logic controller (FLC) with weighted rules, which is equivalent to a conventional fuzzy controller with a weighting factor of each rule, is adopted [2] and a two-phase evolutionary optimization scheme is applied to the FLCs.

In the first phase, initial population for rule structures are given as a stable fuzzy rule. Rule structures and scale factors of the error, change of error and input to the FLC are optimized by EP. The variation of the rule structures is done by the adjacent mutation operator and the scale factors are mutated by the Gaussian random variables. The objective function is constituted by the sum of error, sum of input and the number of used rules.

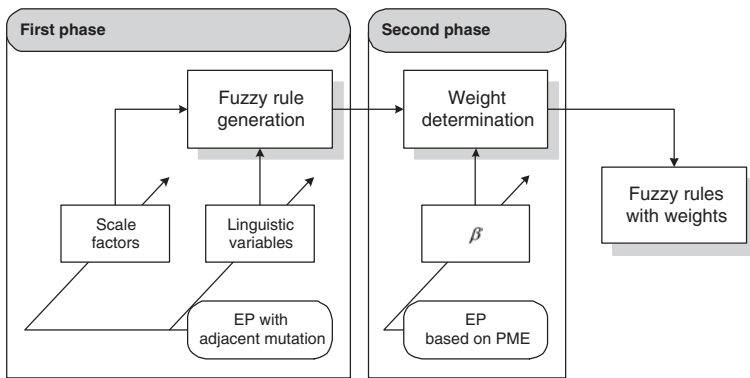


Fig. 1. Overall structure of the two-phase evolutionary optimization

In the second phase, the resultant rules and scale factors of the first step are used. Then PME is applied to determine the weight of each fuzzy rule efficiently. The application of the PME in finding the weights is based on the assumption that all the rules should be utilized to the greatest extent.

The optimization of the second phase can be regarded as fine tuning for the desired output response of the controlled system. Since only several decades of generation is needed for determining the weights in the second phase, the proposed scheme can be used for the on-line control of the time-varying plant. The effectiveness of the proposed scheme is demonstrated by computer simulations.

2 Simulation Results

Consider the following plant:

$$H(z^{-1}) = \frac{1}{2\pi} \cdot \frac{0.02940z^{-1} + 0.01532z^{-2} + 4.643 \times 10^{-5}z^{-3}}{1 - 1.039z^{-1} + 0.03870z^{-2} - 8.993 \times 10^{-8}z^{-3}} \quad (1)$$

In Figure 2(a), solid line is the step response of the second phase, while dotted line is the response of the first phase. The figure shows that the performance can be considerably improved by employing the second phase. The control input is also compared in Figure 2(b).

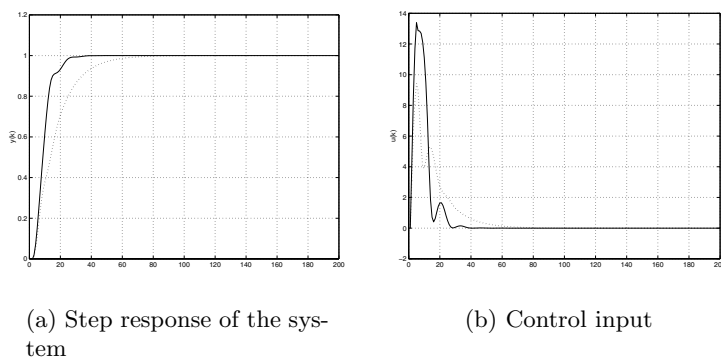


Fig. 2. Step response and control input using fuzzy rule obtained in the second phase

References

1. J.-H. Kim and H. Myung, "Fuzzy Logic Control Using Evolutionary Programming and Principle of Maximum Entropy", Proc. First International ICSC Symposium on Fuzzy Logic, Zurich, Switzerland, pp. C122-C127, 1995.
2. M. Mizumoto, "Fuzzy controls by fuzzy singleton-type reasoning method," Proc. of the Fifth IFSA world congress, Seoul, Korea, pp. 945-948, 1993.

A Simple Evolution Strategy to Solve Constrained Optimization Problems

Efrén Mezura-Montes and Carlos A. Coello Coello

CINVESTAV-IPN
Evolutionary Computation Group (EVOCINV)
Departamento de Ingeniería Eléctrica
Sección de Computación
Av. Instituto Politécnico Nacional No. 2508
Col. San Pedro Zacatenco
México D.F. 07300, MÉXICO
emezura@computacion.cs.cinvestav.mx
ccoello@cs.cinvestav.mx

1 Our Approach

In this paper, we argue that the self-adaptation mechanism of a conventional evolution strategy combined with some (very simple) tournament rules based on feasibility similar to some previous proposals (e.g., [1]) can provide us with a highly competitive evolutionary algorithm for constrained optimization. In our proposal, however, no extra mechanisms are provided to maintain diversity. In order to verify our hypothesis, we performed a small comparative study among five different types of ES: $(\mu + \lambda)$ -ES with and without correlated mutation and a $(\mu + 1)$ -ES using the “1/5-success rule”. The tournament rules adopted in the five types of ES implemented are the following: Between 2 feasible solutions, the one with the highest fitness value wins, if one solution is feasible and the other one is infeasible, the feasible solution wins and if both solutions are infeasible, the one with the lowest sum of constraint violation is preferred. To evaluate the performance of the five types of ES under study, we decided to use ten (out of 13) of the test functions described in [2]. The $(\mu + 1)$ -ES had the best overall performance (both in terms of the best solution found and in terms of its statistical measures). The algorithm of the type of ES adopted (due to its simplicity, we decided to call it Simple Evolution Strategy, or SES) is presented in Figure 1.

Compared with respect to other state-of-the-art techniques (due to space limitations we only compare with respect to [2]), our algorithm produced very competitive results (See Table 1). Besides being a very simple approach, it is worth reminding that SES does not require any extra parameters (besides those used with an evolution strategy) and the number of fitness function evaluations performed (350,000) is the same used in [2].

Acknowledgments. The first author acknowledges support from the Mexican Consejo Nacional de Ciencia y Tecnología (CONACyT) through a scholarship to pursue graduate studies at CINVESTAV-IPN’s Electrical Engineering Department. The second author acknowledges support from (CONACyT) through project number 32999-A.

```

Begin
  t=0
  Create a random initial solution  $x^0$ 
  Evaluate  $f(x^0)$ 
  For t=1 to MAX_GENERATIONS Do
    Produce  $\mu$  mutations of  $x^{(t-1)}$  using:
       $x_i^j = x_i^{t-1} + \sigma[t] \cdot N_i(0, 1) \forall i \in n, j = 1, 2, \dots, \mu$ 
    Generate one child  $x^c$  by the combination of the  $\mu$  mutations using
      m=randint(1,  $\mu$ )
       $x_i^c = x_i^m, \forall i \in n$ 
    Evaluate  $f(x^c)$ 
    Apply comparison criteria to select the best individual  $x^t$  between  $x^{(t-1)}$  and  $x^c$ 
     $t = t + 1$ 
  If (t mod n = 0) Then
    
$$\sigma[t] = \begin{cases} \sigma[t-n]/c & \text{if } p_s > 1/5 \\ \sigma[t-n] \cdot c & \text{if } p_s < 1/5 \\ \sigma[t-n] & \text{if } p_s = 1/5 \end{cases}$$

  End If
End For
End

```

Fig. 1. SES algorithm (n is the number of decision variables of the problem)

Table 1. Comparison of results between our approach (SES) and Stochastic Ranking (SR) [2].

Problem	Optimal	Best Result		Mean Result		Worst Result	
		SES	SR	SES	SR	SES	SR
g01	-15.000000	-15.000000	-15.000	-14.848614	-15.000	-12.999997	-15.000
g02	0.803619	0.793083	0.803515	0.698932	0.781975	0.576079	0.726288
g03	1.000000	1.000497	1.000	1.000486	1.000	1.000424	1.000
g04	-30665.539000	-30665.539062	-30665.539	-30665.441732	-30665.539	-30663.496094	-30665.539
g06	-6961.814000	-6961.813965	-6961.814	-6961.813965	-6875.940	-6961.813965	-6350.262
g07	24.306000	24.368050	24.307	24.702525	24.374	25.516653	24.642
g08	0.095825	0.095825	0.095825	0.095825	0.095825	0.095825	0.095825
g09	680.630000	680.631653	680.630	680.673645	680.656	680.915100	680.763
g11	0.750000	0.749900	0.750	0.784395	0.750	0.879522	0.750
g12	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000

References

1. Kalyanmoy Deb. An Efficient Constraint Handling Method for Genetic Algorithms. *Computer Methods in Applied Mechanics and Engineering*, 186(2/4):311–338, 2000.
2. Thomas P. Runarsson and Xin Yao. Stochastic Ranking for Constrained Evolutionary Optimization. *IEEE Transactions on Evolutionary Computation*, 4(3):284–294, September 2000.

Effective Search of the Energy Landscape for Protein Folding

Eugene Santos Jr.¹, Keum Joo Kim¹, and Eunice E. Santos²

¹ University of Connecticut, Storrs, CT 06269
{eugene, keumjoo}@engr.uconn.edu,

² Virginia Polytechnic Institute and State University, Blacksburg, VA 24061
santos@cs.vt.edu

Abstract. We propose a new algorithmic approach for global optimization in protein folding. We use the information found in various local minima to direct the search for the global minimum. In this way, we explore the energy landscape efficiently by considering only the space of local minima instead of the whole feasible space of conformations.

Our fundamental approach is to sample only the space of local minima and guide the sampling process by exploring protein structure building blocks found in sampled local minima. These building blocks form the basis of information in searching for the global minimum. In particular we employ an iterative algorithm that begins with an initial pool of local minima; construct a new pool of solutions by combining the various building blocks found in the original pool; take each solution and map them to their representative local minima; and, repeat the process. Our procedure seems to share a great deal of commonality with evolutionary computing techniques. Indeed, we even employ genetic operators in our algorithm. However, unlike existing hybrid evolutionary computing algorithms where local minimization algorithms are simply used to “fine-tune” the solutions, we focus primarily on constructing local minima from previously explored minima and only use genetic operators to assist in diversification. Hence, our total number of iterations/generations were demonstrated (empirically) to be quite low (≈ 50) whereas standard genetic algorithms and Monte Carlo are very high ranging from 150,000 to nearly 20,000,000 generations in order to provide sufficient opportunity for these methods to converge and achieve their best solution. We applied our idea to several proteins from the Protein Data Bank (PDB) using the UNRES model[1]. We compared against Standard Genetic Algorithms(SGA) and Metropolis Monte Carlo(MMC) approaches. In all cases, our new approach computed the lowest energy conformation.

Procedure LMBE

begin

$t = 0$;

initialize $P(t)$ with **local minima**;

while termination condition not satisfied *do*

begin

select individuals $P_{new}^{sub}(t)$ from current pool $P(t)$;

recombine structures with selected individuals $P_{new}^{sub}(t)$;


```
determine local minima corresponding to  $P_{new}(t)$ 
replace local minima in  $P_{new}(t)$ ;
evaluate structures  $P_{new}(t)$ ;
end
end.
```

Although LMBE is clearly derived from standard genetic algorithm approaches, our emphasis is on exploring the local minima space and exploits the genetic operators for diversification of the population. Furthermore, this is potentially more systematic in local minimization than memetic algorithms.

Given the prohibitive amount of time to conduct multiple runs of each method over all 100 proteins, each method was run exactly once using the parameter settings determined from pre-trial runs. Hence, the weaknesses and strengths of each method is averaged over the testbed.

For each protein, we initially constructed 100 random conformations. Next, we found the local minimum for each conformation with the gradient descent algorithm [2]. The initial pool consists of these 100 random minimized conformations. The same initial pool was used for LMBE, SGA and MMC for algorithm comparison. The computation time of LMBE varied from 10 mins to 13 hrs, depending on the protein length, amino acid sequence and the genetic parameters (i.e. crossover rate, mutation rate). For MMC, the time was between 21 mins and 16 hours. For SGA, the time varied between 13 mins to 14 hours. Table 1 shows the average energy improvement of LMBE compared with SGA, MMC, and the baseline from PDB. For all 100 proteins, LMBE computed the best energy conformation. Finally, it is interesting to observe that the improvement using LMBE seems to improve significantly for longer proteins on comparison to the existing baseline.

Table 1. Percentage improvement of LMBE over SGA, MMC, and the baseline

Protein Group	SGA(%)	MMC(%)	baseline(%)
Group A (11–20 res.)	8.75	8.82	25.81
Group B (21–30 res.)	11.94	12.50	40.45
Group C (31–40 res.)	13.67	14.05	44.95
Group D (41–50 res.)	13.93	14.30	56.47

References

1. Liwo, A., Kazmierkiewicz, R., Oldziej, S., Pincus, M. R., Wawak, R. J., Rackovsky, S., and Scheraga, H. A.: A United-Residue Force Field for Off-Lattice Protein-Structure Simulations: III. Origin of Backbone Hydrogen-Bonding Cooperativity in United-Residue Potentials. J. Com. Chem. (1998) 19, 259–276
2. Gay, David M.: Algorithm 611: Subroutines for Unconstrained Minimization Using a Model/Trust-Region Approach. ACM ToMS(1983) 9, 503–524

A Clustering Based Niching Method for Evolutionary Algorithms

Felix Streichert¹, Gunnar Stein², Holger Ulmer¹, and Andreas Zell¹

¹ Center for Bioinformatics Tübingen (ZBIT), University of Tübingen,
Sand 1, 72074 Tübingen, Germany,
streiche@informatik.uni-tuebingen.de,
<http://www-ra.informatik.uni-tuebingen.de>

² Institute of Formal Methods in Computer Science (FMI), University of Stuttgart,
Breitwiesenstr. 20/22, D-70565 Stuttgart, Germany,
http://www.informatik.uni-stuttgart.de/ifi/fk/index_e.html

1 Clustering Based Niching

We propose the Clustering Based Niching (CBN) method for Evolutionary Algorithms (EA) to identify multiple global and local optima in a multimodal search space. The basic idea is to apply the biological concept of species in separate ecological niches to EA to preserve diversity. We model species using a multi-population approach, one population for each species. To identify species in a EA population we apply a clustering algorithm based on the most suitable individual geno-/phenotype representation. One of our goals is to make the niching method as independent of the underlying EA method as possible in such a way that it can be applied to multiple EA methods and that the impact of the niching method on the EA mechanism is as small as possible.

CBN starts with a single primordial unclustered population P_0 . Then the CBN-EA generational cycle is entered. First for each population P_i one complete EA generation of evaluation, selection and reproduction is simulated. Now CBN starts with the differentiation of the populations by calling the clustering algorithm on each P_i . If multiple clusters are found in P_i , it splits into multiple new populations. All individuals of P_i not included in the clusters found are moved to P_0 as straying loners. To prevent multiple populations to explore the same niche CBN uses representatives (e.g. a centroid) of all populations $P_{i>0}$ to determine if populations are to be merged. To stabilize the results of the clustering algorithm we currently reduce the mutation step size within all clustered populations $P_{i>0}$. A detailed description of the CBN model can be found in [2].

Of course the performance of CBN depends on the clustering algorithm used, since this algorithm specifies the number and kind of niches that can be distinguished. We decided to use the density-based clustering [1] which can identify an a priori unknown number of niches of arbitrary size, shape and spacing.

This multi-population approach of CBN replaces the global selection of a standard EA with localized niche based selection and mating. This ensures the survival of each identified niche if necessary. Also each converged population $P_{i>0}$ directly designates a local/global optimum.

Table 1. Mean of found optima, in parentheses the number of evaluations needed.

	M0		M1		M2		M3	
No. of optima	5		5		6		10	
MS-HC	4.80	(6.000)	4.90	(6.000)	4.52	(6.000)	8.70	(6.000)
Sharing	4.66	(6.000)	4.54	(6.000)	1.98	(6.000)	8.40	(6.000)
MN-GA(W)	4.83	(355.300)	5.00	(355.300)	5.60	(812.300)	8.98	(1.221.600)
MN-GA(N)	4.94	(355.300)	4.99	(355.300)	3.91	(812.300)	9.80	(1.221.600)
CBN-ES	5.00	(6.000)	4.64	(6.000)	3.94	(6.000)	8.10	(6.000)

2 Results and Conclusions

We examined a CBN Evolution Strategy (ES), a standard ES with fitness sharing with an additional hill-climbing post-processing step and a μ -multi-start hill-climber (MS-HC). We used a $(\mu + 2 \cdot \mu)$ -ES, $\mu = 100$ and $T = 60$ generations as default settings. We compared these algorithms the Multinational GA (MN-GA) on four real-valued two-dimensional test functions [3]. The performance is measured by the number of optima each algorithm has found, averaged over fifty runs. An optimum o_j is considered as found if $\exists x_i \in P_{t=T} \mid \|x_i, o_j\| \leq \epsilon = 0.005$, with the final population $P_{t=T} = \bigcap_i P_{i,t=T}$ in the case of CBN.

Tab. 1 shows that the MN-GA needs much more fitness evaluation than the ES based methods. It shows also that the MS-HC performs well on these simple test functions, so does Sharing in combination with the HC post-processing. Although the parameters for MS-HC and Sharing where optimized for each problem, the CBN-ES proves to be competitive with default parameters.

The advantages of CBN are that it does not alter the search space, that it is able to find niches of arbitrary size, shape and spacing and that it inherits all properties of the applied EA method, since it does not significantly interfere with the EA procedure. There are a number of extensions that can further enhance the CBN. First applying a population size balancing in the case of unevenly sized areas of attraction. Second using a greedy strategy of convergence state management to save function evaluations if a population $P_{i>0}$ is converged.

References

1. M. Ester, H.-P. Kriegel, J. Sander, and X. Xiaowei. A density-based algorithm for discovering clusters in large spatial databases with noise. In E. Simoudis, J. Han, and U. Fayyad, editors, *2nd International Conference on Knowledge Discovery and Data Mining*, pages 226–231, Portland, Oregon, 1996. AAAI Press.
2. G. Stein. Verteiltes dynamisches Nischenmodell fuer JavaEvA. (in German), Diploma Thesis at the Institute of Formal Methods in Computer Science (FMI), University of Stuttgart, Germany, 2002.
3. R. K. Ursem. Multinational evolutionary algorithms. In P.J. Angeline, Z. Michalewicz, M. Schoenauer, X. Yao, and A. Zalazala, editors, *Proceedings of the Congress on Evolutionary Computation*, volume 3, pages 1633–1640, Mayflower Hotel, Washington D.C., USA, 6–9 1999. IEEE Press.

A Hybrid Genetic Algorithm for the Capacitated Vehicle Routing Problem

Jean Berger and Mohamed Barkaoui

Defence Research and Development Canada - Valcartier, Decision Support Technology Section
2459 Pie-XI Blvd. North, Val-Bélair, PQ, Canada, G3J 1X5
jean.berger@drdc-rddc.gc.ca

Abstract. Recently proved successful for variants of the vehicle routing problem (VRP) involving time windows, genetic algorithms have not yet shown to compete or challenge current best search techniques in solving the classical capacitated VRP. In this paper, a hybrid genetic algorithm to address the capacitated vehicle routing problem is proposed. The basic scheme consists in concurrently evolving two populations of solutions to minimize total traveled distance using genetic operators combining variations of key concepts inspired from routing techniques and search strategies used for a time-variant of the problem to further provide search guidance while balancing intensification and diversification. Results from a computational experiment over common benchmark problems report the proposed approach to be very competitive with the best-known methods.

1 Introduction

In the classical vehicle routing problem (VRP) [1], customers with known demands and service time are visited by a homogeneous fleet of vehicles with limited capacity and initially located at a central depot. Routes are assumed to start and end at the depot. The objective is to minimize total traveled distance, such that each customer is serviced exactly once (by a single vehicle), total load on any vehicle associated with a given route does not exceed vehicle capacity, and route duration combining travel and service time, is bounded to a preset limit.

A variety of algorithms including exact methods and efficient heuristics have already been proposed for VRP. For a survey on the capacitated Vehicle Routing Problem and variants see Toth and Vigo [1]. The authors present both exact and heuristic methods developed for the VRP and its main variants, focusing on issues common to VRP. Overview of classical heuristics and metaheuristics may also be found in Laporte et al. [2], and Gendreau et al. [3,4] respectively.

Tabu search techniques [5,6] and (hybrid) genetic algorithms represent some of the most efficient metaheuristics to address VRP and/or its variants. The basic idea in tabu search is to allow selection of worse solutions once a local optimum has been reached. Different memory structures are then used to prevent repeating the same solutions (cycling), and to diversify and intensify the search. Genetic algorithms [7–9]

are adaptive heuristic search methods that mimic evolution through natural selection. They work by combining selection, recombination and mutation operations. The selection pressure drives the population toward better solutions while recombination uses genes of selected parents to produce offspring that will form the next generation. Mutation is used to escape from local minima. Hybrid genetic algorithms combine the above scheme with heuristic methods to further improve solution quality. Tabu search heuristics have proved so far the most successful technique for the capacitated VRP [2], [3], [10], [11]. Alternatively, despite its relative success reported for the traveling salesman problem (see Gendreau et al., [3]) and variants of the vehicle routing problem (VRP) involving time windows [3], [12-21], genetic algorithms have not yet shown to compete with tabu search techniques in solving the capacitated VRP. Limited work using genetic-based techniques for the classical capacitated VRP reports mitigated success so far. As recently proposed procedures match the performance of well-known classical methods [22], others fail to report comparative performance with the best well-known routing techniques, while sometime demonstrating prohibitive run-time to obtain modest solution quality [15], [23]. It is nonetheless believed that genetic-based methods targeted to the classical capacitated VRP have not yet been fully exploited.

In this paper, a competitive hybrid genetic algorithm (HGA-VRP) to address the classical capacitated vehicle routing problem is proposed for the first time. It consists in concurrently evolving two populations of solutions subject to periodic migration in order to minimize total traveled distance using genetic operators combining variations of key concepts inspired from routing techniques and search strategies used for a time-variant of the problem to further provide search guidance while balancing intensification and diversification. A computational experiment conducted on common benchmark problems shows the proposed hybrid genetic approach to be competitive with the best-published methods.

The paper is outlined as follows. Section 2 introduces the main concepts of the proposed hybrid genetic algorithm. Basic principles and features of the algorithm are first introduced. Then, the selection scheme, recombination and mutation operators are presented. Concepts derived from well-known heuristics such as large neighborhood search [24], route neighborhood-based two-stage metaheuristic [25] and λ -interchange mechanism [26] are briefly outlined. Section 3 presents the results of a computational experiment to assess the value of the proposed approach and reports a comparative performance analysis to alternate methods. Finally, some conclusions and future research directions are presented in Section 4.

2 Hybrid Genetic Approach

2.1 General Description

The proposed HGA-VRP algorithm mainly relies on the basic principles of genetic algorithms, disregarding explicit solution encoding issues for problem representation. Genetic operators are simply applied to a population of solutions rather than a population of encoded solutions (chromosomes). We refer to these solutions as solution individuals.

Emphasizing genetic diversity, our approach consists in concurrently evolving two populations of solutions (Pop_1 , Pop_2) while exchanging a certain number of individuals (migration) at the end of a new generation. Exclusively formed of feasible solution individuals, populations are evolved to minimize total traveled distance using genetic operators based upon variations of known routing methods. Whenever a new best solution emerges, a post-processing procedure (RC_M) aimed at reordering customers is applied to further improve its solution quality. The RC_M mutation operator is introduced in Section 2.3. The evolutionary process is repeated until a predefined stopping condition is met. The proposed technique is significantly different from the algorithm presented by Berger and Barkaoui [14] in many respects, including the introduction of new and more efficient operators and its application to a problem variant.

The proposed steady-state genetic algorithm resorts to overlapping populations to ensure population replacement for Pop_1 and Pop_2 . At first, new individuals are generated and added to population Pop_p ($p=1, 2$). The process continues until the overlapping population outnumbers the initial population by n_p . Then, the n_p worst individuals are eliminated to maintain population size using the following individual evaluation:

$$Eval_i = d_i / \max(d_m, d_i). \quad (1)$$

where

d_i = total traveled distance related to individual i ,

d_m = average total traveled distance over the individuals forming the initial populations.

The lower the evaluation value the better the individual score (minimization problem). An elitist scheme is also assumed, meaning that the best solution ever computed from a previous generation is automatically replicated and inserted as a member of the next generation.

The general algorithm is specified as follows:

Initialization

Repeat

$p=1$

Repeat {evolve population Pop_p - new generation}

For $j=1..n_p$ *do*

Select two parents from Pop_p

Generate a new solution S_j using recombination and mutation operators associated with Pop_p

Add S_j to Pop_p

end for

Remove from Pop_p the n_p worst individuals using the evaluation function (1)

$p=p+1$

Until (all populations Pop_p have been visited)

if (new best feasible solution) *then* apply RC_M on best solution {cust. reordering}

Population migration {local best solutions exchange across populations}

Until (convergence criteria or max number of generations)

The initialization phase involves the generation of initial populations Pop_1 and Pop_2 using a random procedure to construct feasible solution individuals. Solutions are generated using a sequential insertion heuristic in which customers are inserted in random order at randomly chosen insertion positions within routes. This strategy is fast and simple while ensuring unbiased solution generation. Migration consists in exchanging local best individuals from one population to another. Convergence is assumed to occur either when solution quality fails to significantly improve over a consecutive number of generations or, after a maximum number of generations.

2.2 Selection

The selection process consists in choosing two individuals (parent solutions) within the population for mating purposes. The selection procedure is stochastic and biased toward the best solutions using a roulette-wheel scheme [9]. In this scheme, the probability to select an individual is proportional to its fitness value. Individual fitness for both populations Pop_1 and Pop_2 is computed as follows:

$$fitness_i = d_i . \quad (2)$$

The notation is the same as in Equation (1). Better individuals show a shorter total traveled distance (minimization problem).

2.3 Genetic Operators

The proposed genetic operators incorporate and combine key feature variations of efficient routing techniques such as Solomon's insertions heuristic I1 [27], large neighborhood search [24] and the route neighborhood-based two-stage metaheuristic (RNETS) [25] successfully applied for the Vehicle Routing problem with Time Windows [1]. Details on the recombination and mutation operators used are given in the next sections.

Recombination. A single recombination operator is considered, namely IB_X(k). It recombines two parent solutions by removing and reinserting customers exploiting a variant of a well-known customer insertion heuristic in constructing a child solution. The insertion-based IB_X crossover operator creates an offspring by combining, one at a time, k routes (R_1) of parent solution P_1 with a subset of customers, formed by nearest-neighbor routes (R_2) in parent solution P_2 . The neighborhood R_2 includes the routes of P_2 whose centroid is located within a certain range of $r_1 \in R_1$ (centroid). A route centroid corresponds to a virtual site whose coordinates refer to the average position of its specific routed customers. The related range corresponds to the average distance separating r_1 from the routes defining P_2 . The routes of R_1 are selected either randomly, with a probability proportional to the number of customers characterizing a tour or based on average distance separating consecutive customers over a route.

A stochastic removal procedure is first carried out to remove from r_1 , customers likely to be migrated to alternate routes. Targeted customers are either selected according to waiting times, distance separating them from their immediate neighbors,

or randomly. Then, using a modified insertion heuristic inspired from Solomon [27] a feasible child tour is constructed, expanding the altered route r_1 by inserting customer visit candidates derived from the nearest-neighbor routes R_2 defined earlier. The proposed insertion technique consists in adding a stochastic feature to the standard customer insertion heuristic I1 [27], by selecting randomly the next customer visit over the three best candidates with a bias toward the best. Once the construction of the child route is completed, and reinsertion is no longer possible, a new route construction cycle is initiated. The overall process is repeated for the k routes of R_1 . Finally, the child inherits the remaining “diminished” routes (if any) of P_1 . If unvisited customers still remain, additional routes are built using a nearest-neighbor procedure.

The whole process is then iterated once more to generate a second child by interchanging the roles of P_1 and P_2 . Further details of the operator may be found in Berger and Barkaoui [14].

Mutation. A suite of four mutation operators is proposed, namely LNSB_M(d), EE_M, IEE_M and RC_M(I). Each mutator is briefly described next.

The LNSB_M (d) (large neighborhood search -based) mutation operator relies on the concept of the Large Neighborhood Search (LNS) method proposed by Shaw [24]. The LNS consists in exploring the search space by repeatedly removing related customers and reinserting them using constraint-based tree search (constraint programming). Customer relatedness defines a relationship linking two customers based upon specific properties (e.g. proximity and/or identical route membership), such that when both customers are considered simultaneously for a visit, they can compete with each other for reinsertion creating new opportunities for solution improvement. Therefore, customers close to one another naturally offer interchange opportunities to improve solution quality. Similarly, solution number of tours is more likely to decrease when customers sharing route membership are removed all together. As stated in Shaw [24], a set of related customers is first removed. The reinsertion phase is then initiated.

The proposed customer reinsertion technique differs from the procedure introduced by Shaw [24] resorting to alternate insertion cost functions and, customer visit ordering schemes (variable ordering scheme) to carry out large neighborhood search. Customer visit ordering determines the effective sequence of customers to be consecutively visited while exploring the solution space (search tree expansion). For diversification purposes, two customer reinsertion methods are proposed, one of them being randomly selected (50% probability) on mutator invocation.

The first reinsertion method relies on the insertion cost function prescribed by Solomon's procedure I1 [27] for the VRP with time windows and, a rank-based customer visit ordering scheme. Customer insertion cost is defined by the sum of key contributions referring respectively to traveled distance increase, and delayed service time. As for customer ordering, customers ($\{c\}$) are sorted (*CustOrd*) according to a composite ranking, departing from the myopic scheme originally proposed by Shaw. The ranking is defined as an additive combination of two separate rankings, previously achieved over best insertion costs ($Rank_{Cost}(c)$) on the one hand, and number of feasible insertion positions ($Rank_{Pos}(c)$) on the other hand:

$$CustOrd \leftarrow Sort \left(Rank_{Cost}(c) + Rank_{|Pos|}(c) \right). \quad (3)$$

The smaller the insertion cost (short total distance, traveled time) and the number of positions (opportunities), the better (smaller) the ranking. The next customer to be visited within the search process is selected according to the following expression:

$$customer \leftarrow CustOrd[INTEGER(L \times rand^D)]. \quad (4)$$

where

L = current number of customers to be inserted,

$rand$ = real number over the interval $[0,1]$ (uniform random number generator),

D = parameter controlling determinism. If $D=1$ then selection is purely random (default: $D=15$).

Customer position selection (value ordering) is then based on insertion cost minimization.

The second reinsertion method involves features of the successful insertion heuristic proposed by Liu and Shen [25], for the VRP with time windows, exploiting the maximization of a regret insertion cost function which concurrently takes into account multiple insertion opportunities (regret cost), to determine customer visit ordering. The regret cost -based customer visit ordering scheme is specified as follows. In the insertion procedure proposed by Liu and Shen [25], route neighborhoods associated to unvisited customers are repeatedly examined for customer insertion. This new route-neighborhood structure relates one or multiple routes to individual customers. In our approach the route neighborhood which differs from the one reported by Liu and Shen [25], is strictly bounded to two tours, comprising routes whose distance separating their centroid from the customer location is minimal. Each feasible customer insertion opportunity is explored over its entire route neighborhood. The next customer visit is selected by maximizing a so-called regret cost function that accounts for multiple route insertion opportunities:

$$Regret\ Cost = \sum_{r \in RN(c)} \{C_c(r) - C_c(r^*)\} \quad (5)$$

where

$RN(c)$ = route neighborhood of customer c ,

$C_c(r)$ = minimum insertion cost of customer c within route r (see [25]),

$C_c(r^*)$ = minimum insertion cost of customer c over its route neighborhood.

For both reinsertion methods, once a customer is selected, search is carried out over its different insertion positions (value ordering) based on insertion cost minimization, exploiting limited discrepancy search [28] as specified in Shaw [24]. However, search tree expansion is achieved using a non-constant discrepancy factor d , selected randomly (uniform probability distribution) over the set $\{1,2\}$. Remaining unvisited customers (if any) are then inserted in additional routes.

The EE_M (edge exchange) mutator focuses on inter-route improvement. EE_M attempts to shift customers to alternate routes as well as to exchange sets of customers between two routes. It is inspired from the λ -interchange mechanism of Osman [26], performing reinsertions of customer sets over two neighboring routes. In the proposed

mutation procedure, each customer is explored for reinsertion in its surrounding route neighborhood made up of two tours. Tours are being selected such that the distance separating their centroid from customer location is minimal. Customer exchanges occur as soon as the solution improves, i.e., we use a "first admissible" improving solution strategy. Assuming the notation (x, y) to describe the different sizes of customer sets to be exchanged over two routes, the current operator explores values running over the range $(x=1, y=0,1,2)$. The IEE_M (intra-route edge exchange) mutation operator is similar to EE_M except that customer migration is restricted to the same route.

The RC_M (I) (reorder customers) mutation operator is an intensification procedure intended to reduce total traveled distance of feasible solutions by reordering customers within a route. The procedure consists in repeatedly reconstructing a new tour using the sequential insertion heuristic I1 over I different sets (e.g. $I=20$) of randomly generated parameter values, returning the best solution generated shall an improved one emerge.

3 Computational Results

A computational experiment has been conducted to compare the performance of the proposed algorithm with some of the best techniques designed for VRP. The algorithm has been tested on the well-known VRP benchmark proposed by Christofides et al. [29]. For these instances, travel time separating two customers corresponds to their relative Euclidean distance. Based on the study reported in Cordeau et al. [10], the experiment consisted in performing a single simulation run for each problem instance and reporting on average performance. HGA-VRP has been implemented in C++, using the GALib genetic algorithm library of Wall [30] and the experiment carried out on a 400 MHz Pentium processor. Solution convergence is assumed to occur when its quality fails to improve by at least 1% over 20 consecutive generations. The parameter values for the investigated algorithm are described below.

In the LNSB_M(d) mutation operator the number of customers considered for elimination runs in the range [15, 21]. The discrepancy factor d is randomly chosen over {1,2}.

Parameter values for the proposed genetic operators are defined as follows:

Population size: 15

Migration: 5

Population replacement:

Elitism

Population overlap per generation: $n_1 = n_2 = 2$

Recombination:

IB_X($k=2$) (20%)

Mutation:

LNSB_M(d) (80%)

EE_M (50%), IEE_M (50%)

RC_M($I=20$) - whenever a new best feasible solution is found.

The migration parameter, a feature provided by GALib, refers to the number of (best) chromosomes exchanged between populations after each generation. Because of limited computational resources, parameter values were determined empirically

over a few intuitively selected combinations, choosing the one that yielded the best average output.

Comparative performance is reported for some of the best-known VRP methods, namely referred to as OS [26], GHL [31], CGL [32], TV [33], WH [34], RR [35], RT [36], TA [37] and BB for HGA-VRP. The results are expressed in terms of total traveled distance. Published competing methods with an average performance gap exceeding about 1% (over all instances) of the best-known result, and/or failing to specify run-time and computational resource characteristics, or reporting prohibitive run-time have been deliberately omitted for comparison purposes. Additional results involving other techniques including classical heuristics may nonetheless be found in Cordeau et al. [10].

Table 1. Comparison of selected heuristics for VRP

Probl. Inst (n)	Perf. Time	OS	GHL	CGL	TV	WH	RR	BB	Best
1 (50)	Dist	524.61	524.61	524.61	524.61	524.61	524.61	524.61	524.61
	(min)	1.90	6.0	4.57	0.81	20.0	1.05	2.00	
2 (75)	Dist	844	835.77	835.45	838.60	835.8	835.32	835.26	835.26
	(min)	0.84	53.8	7.27	2.21	50.0	43.38	14.33	
3 (100)	Dist	838	829.45	829.44	828.56	830.7	827.53	827.39	826.14
	(min)	25.72	18.4	11.23	2.39	145.0	36.72	27.90	
4 (150)	Dist	1044.35	1036.16	1038.44	1033.21	1038.5	1044.35	1036.16	1028.42
	(min)	59.33	58.8	18.72	4.51	285.0	48.47	48.98	
5 (199)	Dist	1334.55	1322.65	1305.87	1318.25	1321.3	1334.55	1324.06	1291.45
	(min)	54.10	90.9	28.10	7.50	480.0	77.07	55.41	
6 (50)	Dist	555.43	555.43	555.43	555.43	555.4	555.43	555.43	555.43
	(min)	2.88	13.5	4.61	0.86	30.0	2.38	2.33	
7 (75)	Dist	911.00	913.23	909.68	920.72	911.8	909.68	909.68	909.68
	(min)	17.61	54.6	7.55	2.75	45.0	82.95	10.5	
8 (100)	Dist	878.00	865.94	866.38	869.48	878.0	866.75	868.32	865.94
	(min)	49.99	25.6	11.17	2.90	165.0	18.93	5.05	
9 (150)	Dist	1184.00	1177.76	1171.81	1173.12	1176.5	1164.12	1169.15	1162.55
	(min)	76.26	71.0	19.17	5.67	345.0	29.85	17.88	
10(199)	Dist	1441.00	1418.51	1415.40	1435.74	1418.3	1420.84	1418.79	1395.85
	(min)	76.02	99.8	29.74	9.11	535.0	42.72	43.86	
11(120)	Dist	1043.00	1073.47	1074.13	1042.87	1043.4	1042.11	1043.11	1042.11
	(min)	24.07	22.2	14.15	3.18	275.0	11.23	22.43	
12(100)	Dist	819.59	819.56	819.56	819.56	819.6	819.56	819.56	819.56
	(min)	14.87	16.0	10.99	1.10	95.0	1.57	7.21	
13(120)	Dist	1547.00	1573.81	1568.91	1545.51	1548.3	1550.17	1553.12	1541.14
	(min)	47.23	59.2	14.53	9.34	510.0	1.95	34.91	
14(100)	Dist	866.37	866.37	866.53	866.37	866.4	866.37	866.37	866.37
	(min)	19.60	65.7	10.65	1.41	140.0	24.65	4.73	
Average									
Deviation from		<i>1.03%</i>	<i>0.86%</i>	<i>0.69%</i>	<i>0.64%</i>	<i>0.63%</i>	<i>0.55%</i>	<i>0.48%</i>	
Best									
Average Time		<i>33.60</i>	<i>46.8</i>	<i>13.75</i>	<i>3.84</i>	<i>222.85</i>	<i>24.65</i>	<i>21.25</i>	
(min)									

Computational results for all problem data sets are summarized in Table 1. The first column describes the various instances and their related size whereas the second specifies total traveled distance and run-time (in minutes). The following columns refer to particular problem-solving methods. Best-known results are depicted in the last column (Taillard [37] and, Rochat and Taillard [36] for instances 5 and 10). The

last row refers to average run-time and performance deviation from the best-known solutions over all problem instances. Related computer platforms include VAX 8600 for OS, Silicon Graphics 36 MHz for GHL, Sun Ultrasparc 10 (440 MHz) for CGL, Pentium PC 200 MHz for TV, Sun 4/630 MP for WH, Sun Sparc4 IPC for RR, Silicon Graphics 100 MHz for RT, Silicon Graphics 4D/35 for TA and Pentium 400 MHz for BB respectively. Explicit results for RT and TA have been omitted because no run-time was provided. It is worth noticing that reported results for WH includes the best computed solution over five execution runs as well as cumulative run-time.

The results of the experiment do not show any conclusive evidence to support a dominating heuristic over the others. But, solution quality and run-time reported for BB proves the HGA-VRP method to be competitive in comparison to alternate techniques as it mostly matches the performance of best-known heuristic routing procedures. Accordingly, the average solution quality deviation (0.48%) and reasonable run-time obtained certainly shows that hybrid genetic algorithms can be comparable to tabu search techniques.

4 Conclusion

A hybrid genetic algorithm (HGA-VRP) to address the classical capacitated vehicle routing problem was presented. Focusing on total traveled distance minimization, HGA-VRP concurrently evolves two populations of solutions in which respective best individuals are mutually exchanged through migration over each generation. Genetic operators were designed to incorporate and combine variations of key concepts emerging from recent promising techniques for a time-variant of the problem, to further emphasize search diversification and intensification. Results from a limited computational experiment showed that HGA-VRP is cost-effective and very competitive in comparison to the best-known VRP metaheuristics.

Future work will be conducted to further improve the proposed algorithm. Existing alternate metaheuristic features and insertion procedures including techniques explicitly designed for the capacitated VRP will be examined to enhance genetic operators while reducing computational cost. Other improvements lie in the introduction of alternate population replacement schemes, fitness models, and an adaptive scheme to dynamically adjust parameters simplifying the configuration procedure in selecting suitable parameters. Application of the approach to other related problems will be explored as well.

References

1. Toth, P. and D. Vigo (2002), "The Vehicle Routing Problem", *SIAM Monographs Discrete Mathematics and Applications*, edited by P. Toth and D. Vigo, Philadelphia, USA.
2. Laporte, G., M. Gendreau, J.-Y. Potvin and F. Semet (1999), "Classical and Modern Heuristics for the Vehicle Routing Problem", *Les Cahiers du GERAD, G-99-21*, Montreal, Canada.
3. Gendreau, M., G. Laporte and J.-Y. Potvin (1998), "Metaheuristics for the Vehicle Routing: Problem", *Les Cahiers du GERAD, G-98-52*, Montreal, Canada.

4. Gendreau, M., G. Laporte and J.-Y. Potvin (1997), "Vehicle routing: modern heuristics. Local Search in Combinatorial Optimization", eds.: E. Aarts and J.K. Lenstra, 311–336, Wiley:Chichester.
5. Glover, F. (1986), "Future Paths for Integer Programming and Links to Artificial Intelligence", *Computers and Operations Research* 13, 533–549.
6. Glover, F. and M. Laguna (1997), *Tabu Search*, Kluwer Academic Publishers, Boston.
7. Holland, J. H. (1975), *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor.
8. Jong De, K. A. (1975), An Analysis of the Behavior of a Class of Genetic Adaptive Systems, Ph.D. Dissertation, University of Michigan, U.S.A.
9. Goldberg, D.E (1989), *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, New York.
10. Cordeau, J.-F., M. Gendreau, G. Laporte, J.-Y. Potvin and F. Semet (2002), "A Guide to Vehicle Routing Heuristics", *Journal of the Operational Research Society* 53, 512–522.
11. Cordeau, J.-F. and G. Laporte (2002), "Tabu Search Heuristics for the Vehicle Routing Problems", *Les Cahiers du GERAD, G-2002-15*, Montreal, Canada.
12. Bräysy, O. and M. Gendreau (2001), "Vehicle Routing Problem with Time Windows, Part II: Metaheuristics", Internal Report STF 42 A01025, SINTEF Applied Mathematics, Department of Optimization, Norway.
13. Dalessandro, S.V., L.S. Ochi and L.M. de A. Drummond (1999), A Parallel Hybrid Evolutionary Metaheuristic for the Period Vehicle Routing Problem. *IPPS/SPDP 1999, 2nd Workshop on Biologically Inspired Solutions to Parallel Processing Problems*, San Juan, Puerto Rico, USA, 183–191.
14. Berger, J. and M. Barkaoui (2000), "An Improved Hybrid Genetic Algorithm for the Vehicle Routing Problem with Time Windows", International ICSC Symposium on Computational Intelligence, part of the International ICSC Congress on Intelligent Systems and Applications (ISA'2000), University of Wollongong, Wollongong, Australia.
15. Machado, P., J. Tavares, F. Pereira and E. Costa (2002), "Vehicle Routing Problem: Doing it the Evolutionary Way", *Proc. of the Genetic and Evolutionary Computation Conference*, New York, USA.
16. Gehring, H. and J. Homberger (2001), "Parallelization of a Two-Phase Metaheuristic for Routing Problems with Time Windows", *Asia-Pacific Journal of Operational Research* 18, 35–47.
17. Tan, K.C., L.H. Lee and K. Ou (2001), "Hybrid Genetic Algorithms in Solving Vehicle Routing Problems with Time Window Constraints", *Asia-Pacific Journal of Operational Research* 18, 121–130.
18. Thangiah, S.R., I.H. Osman, R. Vinayagamoorthy and T. Sun (1995), "Algorithms for the Vehicle Routing Problems with Time Deadlines", *American Journal of Mathematical and Management Sciences* 13, 323–355.
19. Thangiah, S.R. (1995), "Vehicle Routing with Time Windows Using Genetic Algorithms", In *Application Handbook of Genetic Algorithms: New Frontiers, Volume II*, 253–277, L. Chambers (editor), CRC Press, Boca Raton.
20. Thangiah, S.R. (1995), "An Adaptive Clustering Method using a Geometric Shape for Vehicle Routing Problems with Time Windows", In *Proceedings of the 6th International Conference on Genetic Algorithms*, L.J. Eshelman (editor), 536–543 Morgan Kaufmann, San Francisco.
21. Blanton, J.L. and R.L. Wainwright (1993), "Multiple Vehicle Routing with Time and Capacity Constraints using Genetic Algorithms", In *Proceedings of the 5th International Conference on Genetic Algorithms*, S. Forrest (editor), 452–459 Morgan Kaufmann, San Francisco.
22. Sangheon, H. (2001), "A Genetic Algorithm Approach for the Vehicle Routing Problem", *Journal of Economics*, Osaka University, Japan.

23. Peiris, P. and S.H. Zak (2000), "Solving Vehicle Routing Problem Using Genetic Algorithms", *Annual Research Summary – Part I – Research, Section I*
24. *Automatic Control*, School of Electrical and Computer Engineering, Purdue University, http://www.ece.purdue.edu/ECE/Research/ARS/ARS2000/PART_I/Section1/1_19.whml.
25. Shaw, P. (1998), "Using Constraint Programming and Local Search Methods to Solve Vehicle Routing Problems", In *Principles and Practice of Constraint Programming, Lecture Notes in Computer Science*, M. Maher and J.-F. Puget.(eds.), 417–431, Springer-Verlag, New York.
26. Liu, F.-H. and S.-Y. Shen (1999), "A Route-Neighborhood-based Metaheuristic for Vehicle Routing Problem with Time Windows", *European Journal of Operational Research* 118, 485–504.
27. Osman, I.H. (1993), "Metastrategy Simulated Annealing and Tabu Search Algorithms for the Vehicle Routing Problem", *Annal of Operations Research* 41, 421–451.
28. Solomon, M.M. (1987), "Algorithms for the Vehicle Routing and Scheduling Problems with Time Window Constraints", *Operations Research* 35, 254–265.
29. Harvey, W.D. and M.L. Ginsberg (1995), "Limited Discrepancy Search", In *Proceedings of the 14th IJCAI*, Montreal, Canada.
30. Christofides N., A. Mingozzi and P. Toth (1979), "The Vehicle Routing Problem", in Christofides N., Mingozzi A., Toth P. and Sandi C. (eds). *Combinatorial Optimization*, Wiley, Chichester 315–338.
31. Wall, M. (1995), *GALib - A C++ Genetic Algorithms Library, version 2.4*. (<http://lancet.mit.edu/galib-2.4/>), MIT, Boston.
32. Gendreau, M., A. Hertz and G. Laporte (1994), "A Tabu Search Heuristic for the Vehicle Routing: Problem", *Management Science* 40, 1276–1290.
33. Cordeau, J.-F., M. Gendreau and G. Laporte (1997), "A Tabu Search Heuristic for the Periodic and Multi-depot Vehicle Routing Problems", *Networks* 30, 105–119.
34. Toth, P. and D. Vigo (1998), "The Granular Tabu Search and its Application to the Vehicle Routing Problem", *Technical Report OR/98/9*, DEIS, University of Bologna, Bologna, Italy.
35. Wark, P. and J. Holt (1994), "A Repeated Matching Heuristic for the Vehicle Routing Problem", *Journal of Operational Research Society* 45, 1156–1167.
36. Rego, C. and C. Roucairol (1996), "A Parallel Tabu Search Algorithm Using Ejection Chains for the Vehicle Routing Problem", In: Osman IH and Kelly JP (eds). *Meta-Heuristics: Theory and Applications*, Kluwer, Boston, 661–675.
37. Rochat, Y. and E.D. Taillard (1995), "Probabilistic Diversification and Intensification in Local Search for Vehicle Routing", *Journal of Heuristics* 1, 147–167.
38. Taillard E.D. (1993), "Parallel Iterative Search Methods for Vehicle Routing Problems", *Networks* 23, 661–673.

An Evolutionary Approach to Capacitated Resource Distribution by a Multiple-Agent Team

Mudassar Hussain¹, Bahram Kimiaghalam¹, Abdollah Homaifar¹, Albert Esterline¹,
and Bijan Sayyarodsari²

¹NASA Autonomous Control and Information Technology Center, Department of Electrical Engineering, North Carolina A&T State University, Greensboro, NC 27411
muddie27@yahoo.com, {bahram, homaifar, esterlin}@ncat.edu

²Pavilion Technologies, 11100 Metric Blvd., #700
Austin, TX 78758
bijan@pav.com

Abstract. A hybrid implementation of an evolutionary metaheuristic scheme with local optimization has been applied to a constrained problem of routing and scheduling a team of robotic agents to perform a resource distribution task in a possibly dynamic environment. In this paper a central planner is responsible for planning routes and schedules for the entire team of cooperating robots. The potential computational complexity of such a centralized solution is addressed by an innovative genetic approach that transforms the task of multiple route design into a special manifestation of the traveling salesperson problem. The key advantage of this approach is that globally optimal or near optimal solutions can be produced in a timeframe amenable for real-time implementation. The algorithm was tested on a set of standard problems with encouraging results.

1 Introduction

In the era of digital technology, the demand for technological solutions to increasingly complex problems is climbing rapidly. With this increase in demand, the tasks which robots are required to execute also rapidly grow in variety and complexity. A single robot is no longer the best solution for many of these new application domains; instead, teams of robots are required to coordinate intelligently for successful task execution. For example, a single robot is not an efficient solution to automated construction [1], urban search and rescue, assembly-line automation [2], mapping and investigation of unknown and hazardous environments [3], and many other similar tasks.

In this work the problem of resource distribution to a set of distributed goal points by a team of agents is addressed. The formulation is called the Multi-Source Multi-Robot Scheduling (MSMRS) problem. In the MSMRS problem a number of robotic vehicles are available to service a set of goal points with certain demands for a specific type of resource stored at a number of depots or source points in the environment. The capacitated multi-source multi-robot scheduling problem (MSMRS) is an extension to the traditional vehicle routing problem (VRP) in the sense that it in-

corporate additional features and constraints, e.g., multiple depots or resource distribution points for serving the demands at the distributed goal points. The vehicles can use the nearest or optimally located depot for reloading in case the need arises while serving the assigned customers or goal points. The problem has an apparent analogy to the VRP. The difficulty in finding a solution lies in the added complexity and generality of the MSMRS problem. The VRP has itself been proven to be NP complete in [4] and hence cannot be solved to optimality in polynomial time. Optimal solutions for small instances of the VRP have been reported in the literature using exact methods like branch and bound, branch and cut, column generation and dynamic programming techniques [5].

2 Problem Formulation

In the capacitated MSMRS problem a number of possibly heterogeneous robotic vehicles with capacity c_i are available to service a set of goal points. Each goal point has a demand for a specific type of resource stored at different depots or source points in the environment. The objective is to minimize a measure of time and/or distance required to distribute the desired resources to the goal points using optimum number of vehicles. We have treated the MSMRS problem as a form of multiple traveling salesperson problem (MTSP), and the core component of our algorithm is the transformation of this MTSP into a single Traveling salesperson representation that can be solved efficiently.

2.1 Multi-vehicle Resource Distribution with One Source/Depot

We define a multiple robot-scheduling (MRS) problem without capacity constraints as one in which n goal points have to be visited by m robotic vehicles, represented by $(R_0, R_1, R_2, \dots, R_{m-1})$, after first going to one source or depot point s . This means the n goal points can be divided into at most m groups to be assigned to the m available vehicles. If the n goal points are represented by an n element permutation vector, we have to use at most $m-1$ delimiters or markers to indicate the separate subgroups of goal points assigned to different vehicles. These delimiters will also be referred to as virtual sources or as copies of the source point in the rest of this paper. One such delimiter is implicitly assumed to be present at the start and end of the permutation array. To represent the delimiters, we append $m-1$ elements to the original n element array to make it an array of length $n+m-1$. These delimiters can have any random distribution within the permutation vector. If two or more of them appear adjacent in the array, it means that only one of the whole group of vehicles represented by the adjacent delimiter points will be used to serve the following group of goal points, and the number of subgroups will be less than m . The tour or group assigned to the vehicles contains all the following points until a new delimiter or a group of delimiters is encountered. Hence, in case of any adjacent delimiters appearing within the array, $q < m$ vehicles will serve the n goal points. We call the sequence of goal points assigned to a robotic vehicle a subtour. The different arrangements of the delimiters within a solution array have different associated costs, and the algorithm looks for

improvement in this cost. Since the $m-1$ additional elements of the array are only hypothetical markers, we can treat the whole solution array as a graph $G(N, V)$, where $N=n+m-1$ is the set of nodes (goal points and additional virtual points appearing as markers and representing vehicles serving their individual tours), and V is the set of arcs connecting these goal points. The task is to construct a Hamiltonian cycle, as in the case of the single TSP, that starts and ends at the source implicitly represented by an invisible delimiter and is assumed to be assigned to vehicle R_0 . The rest of the q subtours are assigned to vehicles $R_1, R_2, \dots, R_{q-1}$. The cost of all the subtours are calculated and summed up to give a measure of total cost of the overall tour scheme represented by each candidate solution array.

Let $k_1, k_2, k_3, \dots, k_q$ be the numbers of goal points in the subtours 1 to q and $g_i(j)$, $1 \leq j \leq k_i$ is a goal point in the subtour i then the subtour can be represented as

$$\text{subtour}(i) = (g_i(1), g_i(2), \dots, g_i(k_i)) \quad \text{where } i = 1, 2, \dots, q \quad (1)$$

and the cost for each sub-tour $i = 1, 2, \dots, q$ can be calculated as:

$$\text{cost}(\text{subtour}(i)) = d(i) + \text{dist}(s, g_i(1)) + \text{dist}(g_i(k_i), s) + \sum_{j=1}^{k_i-1} \text{dist}(g_i(j), g_i(j+1)) \quad (2)$$

Where s is the source, $d(i)$ is the initial distance of robot i from the source s , and $\text{dist}(a, b)$ is the distance measure between points a and b . The overall cost C can be calculated as follows, where the objective is to minimize the total distance traveled or the total time for the trips for all the vehicles:

$$C = a * \max_j \{\text{cost}[\text{subtour}(j)]\} + \sum_{j=1}^q \text{cost}[\text{subtour}(j)] \quad (3)$$

where $q \leq m$, and a is a scaling factor whose value determines whether more weight is given to the use of less or more vehicles to do the entire tour. The overall objective is to minimize C , subject to the constraint that no goal point can be visited more than once. This constraint is enforced in the permutation array, where each goal point can be assigned to only one vehicle and can never be visited more than once. In the more complex capacitated multiple source multiple vehicle scenario, we have additional constraints such as capacities and demands.

2.2 Multi-vehicle Capacitated Resource Distribution with Multiple Sources

To maintain its advantageous computational properties, the solution representation for the multi-source, multi-vehicle, capacitated resource distribution problem is kept identical to the one for the single-source, un-capacitated problem described in Section 2.1. Hence the effect of multiple depots ($S(1), S(2), \dots, S(s)$), vehicle capacities, and goal point demands are all accounted for by a revised cost function. The revised cost function calculates the individual subtour costs, accounts for the reload trips required by the individual vehicles, and checks the availability of the resources at each

source/depot prior to the vehicle's trip to the source. These trips become necessary when, in the middle of an assigned subtour, a vehicle runs out of resources and has to visit a source point (depot) for reload. If within a subtour the reload trip happens between the edges $g(k)$ and $g(k+1)$ and the optimum source point for the reload is $S(m)$ then the distance for the edge $g(k)$ and $g(k+1)$ i.e. $dist(g(k), g(k+1))$ will be replaced by $dist(g(k), S(m)) + dist(S(m), g(k+1))$ in the cost function. This adjustment is in turn done for all the reloading trips in the all the subtours assigned to different vehicles.

We have adopted a common-sense strategy for the selection of the resource to which the vehicle must travel for reload: choose the resource that minimizes the cost of vehicles trip to the next goal point. While such strategy does not, in general, guarantee an optimal overall solution for an assigned tour to a vehicle (for example, a re-ordering of the cities to be visited by a vehicle may result in a lower overall cost for that tour), the computational burden of seeking an optimal reloading strategy convinced us to adopt the above-mentioned heuristic to preserve the real-time plausibility of the proposed algorithm. An alternate heuristic based reload optimization strategy was also developed to seek local improvement through adjustment of reload points and is discussed in some detail in section 3.4.

The data structure for tour representation is still the $n + m - 1$ length permutation array and the reload trips are not explicitly represented in the candidate solutions.

3 The Evolutionary Algorithm: Genetic Structure

A permutation of integer values, representing the labels of n goal points to be visited, has been used along with $m-1$ points treated as delimiters (virtual sources) to divide the array into at most m sub tours. The use of a vehicle at the start of the permutation is implicitly assumed. This makes the total length of each permutation array to be $n + m - 1$ as shown in figure 1.

1	2	3	..	..	..	$n = 8$	$n + 1$	$n + 2$	...	..	$n + m - 1$

Fig. 1. Representation of a Tour plan for 8 Goals and m robots

The $m-1$ extra points representing virtual sources are used as markers (delimiters) that can divide the n goal points into at most m tours. All the virtual source points are represented by an integer of some value greater than n . So, every time this integer appears in the permutation, the sequence of goal points following this number up to the next virtual source point is a tour associated with one robot. If two or more of the virtual sources happen to appear side by side or if one appears at the beginning or the end of the permutation then the arrangement represents the use of only one of the agents, and the rest of the robots represented by the adjacent virtual sources will not be used. Figure 2 shows a sample chromosome with eight goal points and five robots. Here only one of the two robots represented by virtual sources at positions 7 and 8 will be used. Hence, one robot at the beginning of the tour has to go to goal points 2, 1 and 3, the second robot goes to 4 and 6, the third robot goes to 5 and 8, while the fourth robot at position 11 goes to only point 7. All the robots can be made to go back to the source point where they started the tour and the cost function will account for the cost of this additional journey. Therefore, four robots, out of a total of five, will be

used to accomplish the combined task. Each subtour will be assigned a robot for which the distance to the first goal point in the tour, accounting for the necessary trip to a resource closest to the robot, is the smallest.

2	1	3	$n + 1$	4	6	$n + 2$	$n + 4$	5	8	$n + 3$	7

Fig. 2. Tour with virtual sources distributed throughout

Note that different arrangements of these virtual sources within the candidate solution, and hence different numbers of sub-tours, are possible. The fitness value of the chromosome will of course vary with each arrangement. Thus, by preserving a permutation representation for the multi-vehicle multi-source capacitated distribution problem, we can also determine the optimal number of vehicles needed.

3.1 Recombination and Mutation Operators

The representation allows for the use of the standard genetic operators applied to the TSP-like sequencing problems based on the permutation representation of the candidate solutions. The crossover operators include partially mapped crossover (PMX) [6], cycle crossover and modified cycle crossover (CX) [7], and the edge recombination crossover (ER) [8] and many others. Different versions of these operators can be found in the literature and all have been coded and used in different combinations with other genetic operators to assess their impact on the quality of the off-spring produced.

The edge recombination operator has proved to work best on the problems where the edge information is of critical importance and not the position of the goal points, e.g., for all variants of the TSP problems and this conclusion proved true during the test performed for the MSMRS evolutionary algorithm.

The swap mutation operator has been used with a low probability in this work. In this procedure, two goal points or nodes are randomly picked from the parent and the positions are swapped. This operation is meant to introduce diversity in the population to prevent premature convergence.

This is a “steady state” Evolutionary Algorithm (EA), where the population changes incrementally, one by one, rather than with the replacement of the entire generation. In each iteration, one new child is produced by breeding and replaces the worst population member. The replacement scheme allows new individuals to be inserted into the population only if they differ from existing best by a certain percentage there by preserving the diversity in the population.

3.2 2-Opt Edge Exchange Local Improvement Heuristic

To speed up the convergence of the algorithm to good solutions, a local improvement heuristic has also been tested in the algorithm run to yield a hybridized version of the EA. The hybrid EA incorporates the local search techniques at various stages of the genetic process. The k -Opt like procedure [9] is used to locally optimize the sub-tours assigned to each robotic vehicle by eliminating the crossing edges. The k -Opt ex-

change process basically comprises deletion of k edges in the tour and their replacement by k new edges. If the change results in tour cost improvement, then the modified tour is kept, otherwise it is discarded. Either the whole random population generated initially (preprocessing) or the offspring, produced after the recombination and mutation operations (post processing), can be improved. The implications of applying the heuristic at different stages are discussed in Section 4.

3.3 Back Stepping Heuristic for Improved Reload Point Assignment

One very important aspect affecting the cost of distribution resources is the cost of making reload trips during the execution of the tour plans. The reload trips have to be planned in such a way that they add minimum possible cost to the overall tour. A local optimization process was developed with the intention of making minor adjustments to the reload points along vehicle subtours to obtain an improvement in overall tour costs. The process is referred to as Reload Back Stepping (RBS). To begin with, all the subtours, assigned to different vehicles, within the complete tour are extracted. The reload points, based on the full exhaustion of vehicle capacity as discussed earlier for the cost evaluation process, are then sorted out. The parts of the subtour separated by the reload operation will be referred to as sub-subtours here. In almost all the cases, the vehicles have unused capacity based on this kind of reload scheme, i.e., the points serviced after the last reload trip of the vehicle (referred to as “tail” here) do not use all of the vehicle capacity. This unused capacity provides an opportunity for adding more goal points to the tail, i.e., the points that were a part of the sub-subtour before the last reload can now be added to the tail. More options are hence available to shift (back step) the last reload point in the actual subtour to a point that minimizes the reload trip cost. This minimization is possible because of the flexibility in the choice of the reload points instead of having to make the trip at a fixed prescribed point as in previous case. The available new choices for the reload point are then evaluated by calculating the cost of a reload trip to the closest source point in each case, and the position with the best result is picked. The last reload point is then shifted back if needed, hence adding new points to the tail if the shift is profitable. The points after this new last reload point are curtailed from the subtour vector and stored in a separate array called the newtour array. The whole process is repeated for the reduced tour and eventually the newtour array becomes the new possibly improved subtour. The adjustment is propagated back toward the beginning of the subtour, where the tail is always the sequence of points after the last reload point that has not been considered for readjustment. This procedure is done to the subtours assigned to all the vehicles, and new subtours for each are then put back together to obtain a new overall tour with possibly lower cost. Figure 3 presents the example single source scenario with Figure 4 presenting one with multiple sources. Here 9 point each with demand one have to be serviced by a robotic vehicle with capacity 6.

The reload trips based on initial scheme are shown with solid lines and the new improved reload point assignments are represented by dotted lines. The back stepping moves yield a decrease in cost associated to the reload trip and hence to the tour assigned to the vehicle.

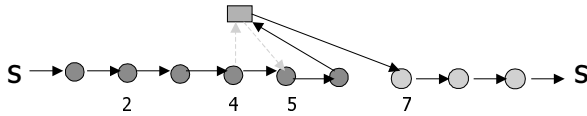


Fig. 3. Sample single source tour for reload back stepping

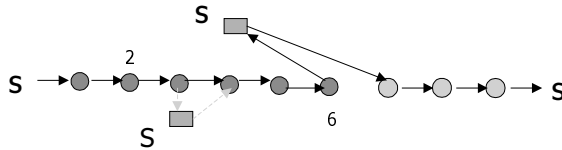


Fig. 4. Sample multiple source tour for reload back stepping

4 Discussion of the Results

The datasets that we have used for testing and comparison include the three datasets by Augerat [10] and one by Eilon [11]. All these datasets have been used extensively and best solutions have been reported. The datasets include problems of varying dimensions. The coordinates of goal points and the respective demands have been provided as well as the coordinates of the single source point or the depot. To make comparison to the available results in the literature possible, we have reduced the number of sources to one, and assumed that all the vehicles are stationed at that one source point. This effectively means that one vehicle is making the entire tour and that the subtours indicated by the reload trips can be treated as independent trips by different vehicles in order to make the comparison to the VRP benchmark problems feasible. This can be done without loss of generality since the algorithm is flexible in the number and location of source points and the vehicle starting positions.

Several exploratory runs were made to find effective values for the population size and operator probabilities. The final parameters chosen are a population size of 100 for the small sized problems, 150 for the eighty goal point problem, and 250 for the two larger ones. This choice of population size depicts the values that performed best during rigorous testing. Bigger initial population size is required for the bigger problems due to the requirement of the representation of more diverse regions of the search space in the initial candidate solution pool. Recombination probability was set to 1 and the mutation process has a low probability of 0.001. The algorithm was allowed to run for more than 10000 iterations for all the sample problems tested. The results obtained for the seven test problems have been tabulated in Table 1.

As shown in Table 1, the results obtained with the pure EA with no local improvement for the 32, 33, 44, 64 goal points show near optimal outputs whereas, for

larger problems, the algorithm was not able to find good solutions within the 12000 iterations and needs more run-time to converge. The same problems were tested with a 2-Opt like local improvement heuristic, described above, applied to seed the initial population with some quality solutions. 30% of the candidates in the initial population were pre-improved using the 2-Opt exchange process. All of the initial population could be pre-optimized, but this was not done in the interest of maintaining diversity in the genetic information processed by the genetic operators. Local improvement resulted in better solutions for all the problem instances and reduced the convergence time for the algorithm (columns 7 and 8).

Table 1. Simulation results and comparison to the reported best solutions

Prob size	No of Iterations x100	Pop size	Best reported	Best with pure EA	Sol. reached at iteration. (pure EA)	Best with 2-opt Improvement	Sol. reached at iteration (2-opt Imp)	Number of vehicles used	% Deviation from best (pure EA)	% Deviation from best (2-opt Imp)
32	200	100	784	798.35	13000	786.5	8000	5	1.83	.32
33	200	100	742	751.23	7000	742	7000	6	1.24	0
44	200	100	944	974.46	8000	973	8470	6	3.23	3.07
64	200	100	1402	1463.76	9800	1421.8	7300	9	4.41	1.41
80	200	150	1764	2313.5	7500	1816	18000	10	31.15	2.95
100	1000	250	681	1002.78	10700	731.3	95000	4	47.25	6.87
135	1200	250	1165	1859	11300	1180	40500	7	59.5	1.29

Table 2. Comparison of results to the existing known best solutions applying

Prob size	No of Iterations	Best reported	Best with Hybrid EA (BS)	%Deviation from best (Hybrid EA)
32	20000	784	786.5	.32
33	20000	742	742	0
44	20000	944	972.1	2.98
64	20000	1402	1421.8	1.41
80	20000	1764	1816	2.95
100	100000	681	706.8	3.79
135	120000	1165	1180	1.29

The same set of problems was solved using the EA augmented by both of the local improvement heuristics, i.e. the 2-Opt local improvement and the reload back stepping process for reload tour improvement. The results for the test runs obtained with the same set of parameters and stopping criteria are tabulated in Table 2. It can be

seen from the table that some improvement was achieved through the reassignment of reload points for the 44 and 100 goal point problems. The reason for improvement in only these two cases is that either enough extra capacity in the tail part of the solutions was not available for the other problems or they were already close to optimal and the initial assignment of reload points was good enough so that the RBS procedure could not make any significant improvements.

The relationship between problem size and the time to reach the solution, for the problem instances tested, has somewhere between a linear and quadratic rate of increase. The time to reach the best solution is measured on a 600 MHz Intel Pentium III based computer with 512 megabytes of physical memory and MS Windows 2000 operating system.

A sample route plot for the 64-city capacitated benchmark problem by Augerat is provided in figure 5. It can be seen that all the robot tours are locally optimal and the overall result is within 1.5% of the global optimum reported in the literature (Table 1)

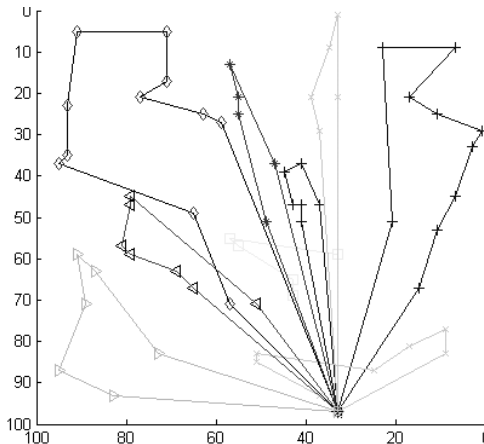


Fig. 5. Route plot of 64-goal point problem (Augerat, et. al.)

Since our literature search did not produce any benchmark resource distribution/vehicle routing problem with multiple resources and capacitated vehicles, we created some hypothetical problem instances to test the utility of our proposed algorithm. Sample results for a very simple and a relatively complex problem are shown in Figure 6. Figure 6(a) shows the route distribution for a problem with nine goal points, each having a demand of 1, two robots with capacity three, and two sources. Figure 6(b) shows the route distribution of a ninety-six goal point problem with four source points and five available vehicles each having a capacity of 195. The simple problem yielded optimal solution where as the ninety-six point problem yielded a good feasible solution, as can be seen from Figure 6(b). The exact route distributions for the sub-tours depicted in Figure 6(b) are shown in Table 3. Column 2 shows the breakdown of the assigned tours for each vehicle into sub-subtours depicting the number of reloads that particular vehicle has to make to one of the source points.

Table 3. Route distribution of multi-source, multi-vehicle resource distribution problem using the EA

Vehicle	Sub-Subtour	Route	Demand	Cost
1	1	84,59,41,43,63,13,8,68,38,93,92,74,55,44,73,62,19,81	188	603
	2	86,27,20,16,17,37,69,9,72,60,25	184	463
2	1	78,94,7,5,39,64,32,87,65,47,1,88,33	184	537
	2	42	13	52
3	1	56,66,36,71,53,12,3,76,50,51,24,80,48,10,2,18,14,67,96	183	444
	2	6,22,85,15	87	247
4	1	21,91,23,30,83,40,49,34,4,77,31,35,82,79,45	186	485
	2	28,11,26,75,46,57,95,29,54,58,70,61,90,52,89	195	465
Objective Function:				3296

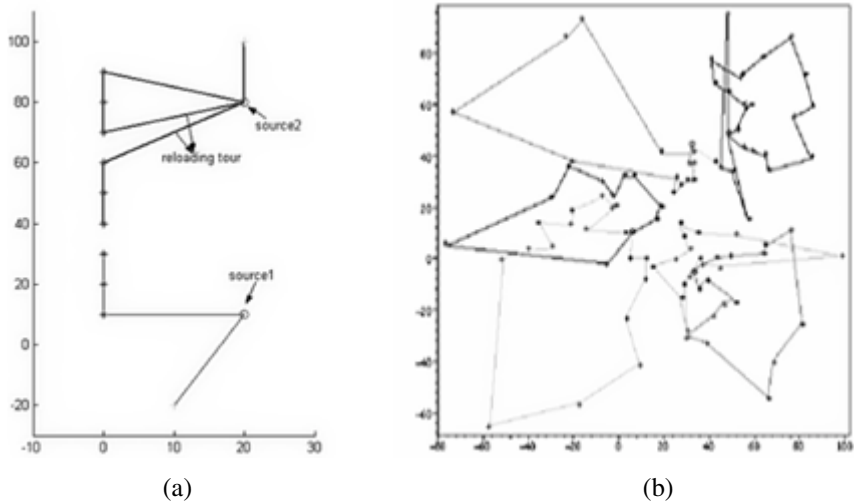


Fig. 6. (a) 9 goal point problem (b) 96 goal point problem

The EA hybridized with both the 2-Opt and RBS reload local optimization schemes was also applied to the multiple source point problems to study the effect of any adjustment of reload points for individual vehicle subtours. The effect of the reload back stepping local improvement to the simple 9-point example of figure 6(a) is shown in Figure 7. In this case, the RBS applied to the original tour with a cost of 204.63 reduced the cost to 196.34 after adjustment.

The effect of application of The RBS process to the example of Figure 6(b) is tabulated in Table 4. It can be seen from that the original overall tour cost of 3296 was reduced to 3228 due the back stepping adjustment of reload points in subtours for the vehicles two and three. No improvement in other subtours was obtained due to the

unavailability of flexibility in the tail part of those subtours. Moreover, the change due to the application of the RBS heuristic is not very significant because of the close proximity of the source points to the vehicle subtour clusters. It can be much more significant if the source points are located farther from the subtours assigned to respective vehicles

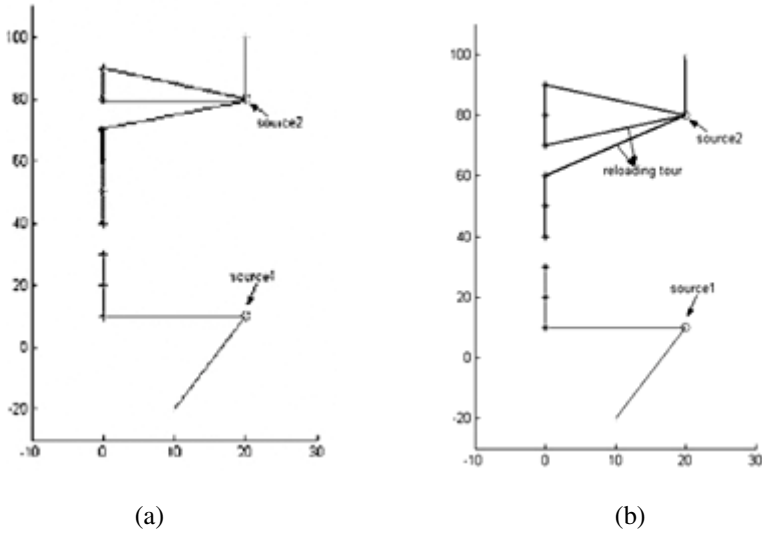


Fig. 7. Effect of the application of RBS heuristic to 9 goal point multiple source problem: (a)After improvement, (b) original assignment

Table 4. Effect of cost improvement with RBS heuristic for 96 goalpoint multiple source problem

Vehicles	Tourcost (without RBS)	Tourcost (with RBS)
1	1066	1066
2	589	553
3	691	659
4	950	950
Total	3296	3228

5 Conclusions and Future Work

A permutation based steady state GA and a modified version of this algorithm with local improvements have been used to efficiently solve a Multi-robot Multi-Source, Capacitated Resource Distribution problem. A novel formulation of the problem is used to translate the original problem into a variant of the well known TSP problem

for which efficient GA-based solver is developed.. The results verify the utility of the approach for routing different sized robot teams for a resource delivery application. The algorithm has been tested in a static environment.

The algorithm has achieved acceptable results with favorable numerical properties. Ongoing research aims at introducing more realistic constraints encountered in real-life logistics problems into the problem. The end product will be a good robust algorithm for logistics problems with spatial and temporal as well as precedence constraints.

References

1. Bohringer, K., Brown, R., Donald, B., Jennings, J., and Rus, D., "Distributed Robotic Manipulation: Experiments in Minimalism", *Proceedings of the International Symposium on Experimental Robotics (ISER)*, 1995.
2. Cicerello, V., and Smith, S., "Insect Societies and Manufacturing", *the IJCAI-01 Workshop on Artificial Intelligence and Manufacturing: New AI Paradigms for Manufacturing*, 2001.
3. Burgard, W., Moors, M., Fox, D., Simmons, R., and Thrun, S., "Collaborative Multi-Robot Exploration", *Proceedings of the IEEE International Conference on Robotics and Automation*, San Francisco CA, April 2000.
4. Parker, G. R. and R. L. Rardin. "An Overview of Complexity Theory in Discrete Optimization: Part II. Results and Implications," *IIE Transactions*, 14(2): 83–89, 1982.
5. Araque, J.R., Kudva, G., Morin, T.L., and J.F. Pekny, "A Brach-and-Cut Algorithm for Vehicle Routing Problems", *Annals of Operations Research* 50, 1994.
6. Goldberg, D. "Genetic Algorithms in Search Optimization and Machine Learning", Addison Wesley 1989.
7. Oliver, I. Smith, D. and Holland, J. "A Study of Permutation Crossover Operators on the Traveling Salesman Problem". In the proceedings of *Second International Conference on Genetic Algorithms and their Applications* 1987.
8. Whitley, D., Starkweather, T. and Fukuay, D. "Scheduling Problems and the travelling Salesman: The genetic Edge Recombination Operator" in *Proceedings of third International Conference on Genetic Algorithms and their Applications*, pp 133–139, 1989.
9. Johnson, D. S., "The Traveling Salesman Problem: A case Study". *Local Search in Combinatorial Optimization*, John Wiley and Sons, Chichester, UK. 215–310.
10. Augerat, P. Vrp-instances. <http://www-apache.imag.fr/~paugerat/VRP/INSTANCES>.
11. Eilon, S. and Christofides, N. (1969), "An Algorithm for Vehicle Dispatching Problem." *Operational Research Quarterly* 20(3), 309–318.

A Hybrid Genetic Algorithm Based on Complete Graph Representation for the Sequential Ordering Problem

Dong-Il Seo and Byung-Ro Moon

School of Computer Science & Engineering, Seoul National University
Sillim-dong, Kwanak-gu, Seoul, 151-742 Korea

{diseo, moon}@soar.snu.ac.kr

<http://soar.snu.ac.kr/~{diseo, moon}/>

Abstract. A hybrid genetic algorithm is proposed for the sequential ordering problem. It is known that the performance of a genetic algorithm depends on the survival environment and the reproducibility of building blocks. For decades, various chromosomal structures and crossover operators were proposed for the purpose. In this paper, we use Voronoi quantized crossover that adopts complete graph representation. It showed remarkable improvement in comparison with state-of-the-art genetic algorithms.

1 Introduction

Given n nodes, *sequential ordering problem* (SOP) is the problem of finding a Hamiltonian path of minimum cost satisfying given *precedence constraints*. Formally, given a set of nodes $V = \{1, 2, \dots, n\}$ and cost matrix $C = (c_{ij})$, $c_{ij} \in \mathbb{N} \cup \{\infty\}$, $i, j \in V$, it is the problem of finding a Hamiltonian path π that satisfies precedence constraints and minimizes the following:

$$\text{Cost}(\pi) = \sum_{i=1}^{n-1} c_{\pi(i)\pi(i+1)}.$$

Here, the precedence constraints are marked by infinity (∞) in the cost matrix, i.e., if $c_{ji} = \infty$, node j cannot precede node i in the path. The relationship is denoted by $i \prec j$; node i is called a *predecessor* of node j and node j is called a *successor* of node i . It is assumed that the path starts at node 1 and ends at node n , i.e., $1 \prec i$ and $i \prec n$ for all $i \in V \setminus \{1, n\}$. Generally, the cost matrix C is asymmetric and the precedence constraints are transitive and acyclic. The problem is also called ‘asymmetric Hamiltonian path problem with precedence constraints’. The special case of SOP with empty precedence constraints is reduced to asymmetric traveling salesman problem (ATSP). As ATSP is an NP-hard problem, so is SOP.

The problem arises in various practical fields such as manufacturing, routing, and scheduling. However, not very much attention has been paid to the

problem, while TSP, which is a reduction of SOP, has been one of the most popular problems in the combinatorial optimization area. Cutting-plane approach [1], Lagrangian relax-and-cut method [2], and branch-and-cut algorithm [3] are mathematical model-based approaches. The genetic algorithm using a crossover called maximum partial order/arbitrary insertion (MPO/AI) [4] and the hybrid ant colony system called HAS-SOP [5] are state-of-the-art metaheuristics for SOP. Path preserving 3-Opt (pp-3-Opt) algorithm and its variants such as SOP-3-exchange [5] are the most popular local improvement heuristics for hybrid metaheuristics.

In this paper, we propose a new genetic algorithm for SOP. We adopt Voronoi quantized crossover to exploit the topological linkages of genes in the genetic search. The crossover is based on complete graph representation.

The rest of this paper is organized as follows. We mention the background in Section 2 and describe the proposed genetic operators in Section 3. The experimental results are provided in Section 4. Finally, the conclusions are given in Section 5.

2 Background

The building block hypothesis implies that the power of a genetic algorithm lies in its ability to create and grow the building blocks efficiently. Building blocks appear in interactive gene groups. The interaction between genes means the dependence of a gene's contribution to the fitness upon the values of other genes. The interaction is also called *epistasis* in GA, although it is wider than the biological definition of epistasis [6,7,8].

A gene group is said to have *strong linkage* if the survival probability of the corresponding schema is higher than normal, and it is said to have *weak linkage* otherwise [6]. To make building blocks survive through recombinations, we must let the strongly epistatic gene groups have stronger linkage than ordinary gene groups [6,9]. The linkage of a gene group is affected by various factors. Particularly, the linkage determined by the relative positions of genes in the chromosome is called *topological linkage* [10]. In the case, each gene is placed in an Euclidean or non-Euclidean space, called *chromosomal space*, to represent the linkages between genes. In order to make the topological linkages reflect well the epistatic structure of a given problem, we need to choose an appropriate *chromosomal structure*. The chromosomal structure here means the conceptual structure of genes used for the crossover operator. A typical chromosomal structure is one-dimensional array. In general, multi-dimensional representations are more advantageous than simple one-dimensional representations for highly epistatic problems [10]. For example, two-dimensional array, two-dimensional real space (plane), and complete graph are available.

Recently, a large number of genetic algorithms that exploit the topological linkages of genes have been proposed. They are classified into three models: *static linkage model*, *adaptive linkage model*, and *evolvable linkage model* [10]. The linkages are fixed during the genetic process in the static linkage model.

```

1.  VQX( $n, k, d_g, p_1, p_2$ )
2.  {
3.       $I \leftarrow \{1, 2, \dots, n\}; K \leftarrow \{1, 2, \dots, k\};$ 
4.      Select a subset  $R = \{s_1, s_2, \dots, s_k\} \subset I$  at random;
5.      for each  $i \in I$  {
6.           $r[i] \leftarrow \arg \min_{j \in K} \{d_g(s_j, i)\}, s_j \in R;$ 
7.      }
8.      for each  $j \in K$  {  $u[j] \leftarrow 0$  or  $1$  at random; }
9.      for each  $i \in I$  {
10.         if ( $u[r[i]] = 0$  and  $u[r[p_1[i]]] = 0$ ) then  $o[i] \leftarrow p_1[i];$ 
11.         else if ( $u[r[i]] = 1$  and  $u[r[p_2[i]]] = 1$ ) then  $o[i] \leftarrow p_2[i];$ 
12.         else  $o[i] \leftarrow nil;$ 
13.     }
14.      $o \leftarrow \text{GreedyRepair}(o);$ 
15.     return  $o;$ 
16. }
```

Fig. 1. Voronoi quantized crossover for SOP.

They adaptively changes in the adaptive linkage model, and evolve in parallel with the allele values in the evolvable linkage model. We adopt the Voronoi quantized crossover [11] and apply the static linkage model in this paper.

3 Genetic Operators

3.1 Voronoi Quantized Crossover

In Voronoi quantized crossover (VQX), a chromosome is a complete graph of genes where each edge weight, called *genic distance*, reflects the epistatic strength between the two corresponding genes. The graph is directed if the genic distance is asymmetric. In fact, the genes are assigned a position in a non-Euclidean space defined by the genic distances. By adopting such a non-Euclidean chromosomal space, we aim to reflect the epistases with minimal distortion in the crossover. The proposed heuristic for the genic distance assignment is described in Section 3.2. VQX was applied to the traveling salesman problem for the first time [11]. Applying VQX to SOP needs considerable modification. We describe the VQX for SOP in the following.

For the problem, we use the locus-based encoding¹ as in [12]; one gene is allocated for every node and the gene value represents the index of its next node in the path. VQX has a simple structure. Figure 1 shows the pseudo code

¹ The term *encoding* here must be distinguished from the term *representation* because we mean by encoding the actual scheme to store solutions not for crossover in this paper.

```

1. GreedyRepair( $o$ )
2. {
3.    $S \leftarrow$  Extract path segments from  $o$ ;
4.    $S \leftarrow$  PrecCycleDecomposition( $S$ );
5.    $s_0 \leftarrow$  the segment that contains node 1 in  $S$ ;
6.    $S \leftarrow S \setminus \{s_0\}$ ;
7.   do {
8.      $s \leftarrow$  the nearest segment from  $s_0$  among the segments, in  $S$ ,
9.       all whose predecessors are already contained in the segment
10.    itself or in  $s_0$ ;
11.    Attach  $s$  to  $s_0$ ;  $S \leftarrow S \setminus \{s\}$ ;
12.  } while ( $|S| > 0$ );
13.   $o' \leftarrow$  the solution of the segment  $s_0$ ;
14.  return  $o'$ ;
15. }
```

Fig. 2. Greedy repair.

of VQX where n is the number of genes and k is the crossover degree ranged from 2 to n . The function $d_g : I^2 \rightarrow R$ represents the genic distance. The two parents and the offspring are denoted by p_1 , p_2 , and o , respectively. Following the convention, the notation “arg min” takes the argument that minimizes the value. Given a number of vectors, the Voronoi region of a vector is defined to be the nearest neighborhood of the vector [13]. In VQX, the chromosomal space defined by d_g is quantized into k Voronoi regions determined by the k randomly selected genes (lines 4–7), then a sort of block-uniform crossover [14] is performed on the regions (lines 8–13). We use a random tie-breaking in the calculation of “arg min” in the crossover (line 6).

The part of gene inheritance (lines 8–13) goes as follows. At first, each region is masked white or gray at random. The white and gray correspond to 0 and 1, respectively, in line 8. Then the genes in the white regions are inherited from parent 1 and the others are inherited from parent 2 (lines 9–13). At this time, the gene values are not always copied but only when a gene (gene i) and the gene pointed by it (gene $p_1[i]$ or gene $p_2[i]$) belong to the same-colored region. That is, an arc in a parent has a chance to survive in the offspring when both end points belong to the same-colored region(s). The word *nil* is used for the genes whose values are not determined. As a result, a partial solution consisting of path segments is generated. We use a greedy approach to repair it. Figure 2 shows the pseudo code of the greedy repair. Beginning with the segment containing node 1 (lines 5–6), it repeatedly merge segments available (lines 7–12). An available segment is a segment all whose predecessors are contained in the segment itself or in the segments already merged.

Because the segments are inherited from the two parents, it may include precedence cycles. Therefore, a precedence cycle decomposition algorithm is re-

```

1.  PrecCycleDecomposition( $S$ )
2.  {
3.      START:
4.       $D \leftarrow \emptyset$ ;  $T \leftarrow \emptyset$ ;
5.      do {
6.          Select a segment  $s$  from  $S \setminus D$  at random;
7.           $D \leftarrow D \cup \{s\}$ 
8.          for each node  $i$  in  $s$  {
9.              for each predecessor  $i_p$  of  $i$  {
10.                  $s_p \leftarrow$  the segment contains  $i_p$  in  $S$ ;
11.                 if ( $s_p \neq s$  and  $(s, s_p) \notin T$ ) {
12.                     if  $((s_p, s) \in T)$  {
13.                         Split  $s$  into  $s'$  and  $s''$ ;
14.                          $S \leftarrow S \setminus \{s\} \cup \{s', s''\}$ ;
15.                         goto START;
16.                     } else {
17.                          $T \leftarrow T \cup \{(s, s_p)\}$ ;
18.                          $T \leftarrow \text{TransitiveClosure}(T)$ ;
19.                     }
20.                 }
21.             }
22.         }
23.     } while ( $|D| < |S|$ );
24.     return  $S$ ;
25. }
```

Fig. 3. Precedence cycle decomposition algorithm.

quired before merging the segments (line 4 in Figure 2). Figure 3 shows the pseudo code of the algorithm. The algorithm inspects the precedence relationships between the segments and if it finds a precedence cycle, it decomposes the cycle by splitting a segment involved in the cycle into two sub-segments (lines 13–14). The splitting point is determined to be the position before the node i or the position after the node i in the figure. The position with more balanced sizes of the resulting segments is preferred. The splitting is repeated until no cycle is found (lines 3–23). TransitiveClosure() returns the transitive closure of a precedence relation T (line 18).

Figure 4 shows an example of VQX for SOP. In the figure, the nodes (genes) and the non-trivial precedence constraints are drawn by small circles and dashed arrows, respectively. For the convenience of illustration, we assumed the chromosomal space to be a two-dimensional Euclidean space. The assumption is merely for the visualization. At first, the chromosomal space is quantized into nine Voronoi regions as in (a). Then, the offspring inherits path segments from the parents. Figures 4(b)–(c) shows the two parents and Figure 4(d) shows the

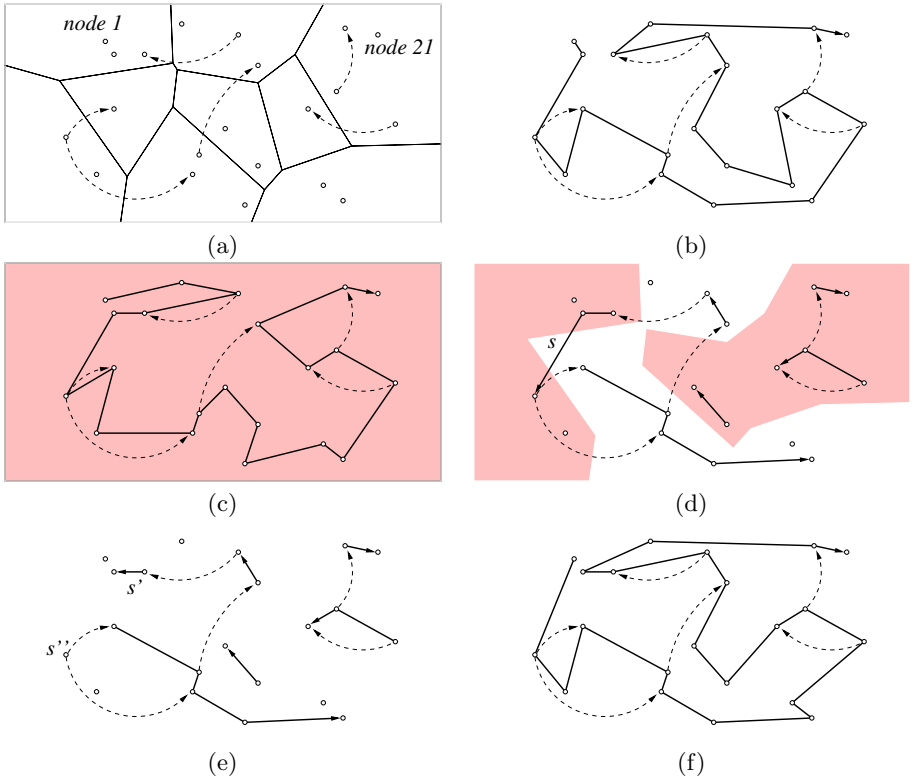


Fig. 4. An illustration of VQX for SOP. (a) A chromosomal space quantized into nine Voronoi regions. (b) Parent 1. (c) Parent 2. (d) Inherited path segments. (e) After precedence cycle decomposition. (f) Repaired path segments.

inherited path segments. By the precedence cycle decomposition, the segment s in (d) is split into segments s' and s'' in (e). Finally, an offspring is generated by the greedy repair as in (f).

3.2 Genic Distance Assignment

We apply the static linkage model to the genetic algorithm, i.e., the genic distances are assigned statically before running the genetic algorithm. Intuitively, an ideal value of a genic distance is a value inversely proportional to the strength of the epistasis. However, no practical method to get the exact values of the epistases is known yet. Therefore, we rely on heuristics. The genic distance from gene i to gene j is defined as

$$d_g(i, j) = |\{l \in V : c_{il} < c_{ij}\}| \quad (1)$$

where V is the set of nodes and c_{pq} is the (p, q) element of the cost matrix. It is based on the fact that the epistasis reflects the topological locality of the nodes. The genic distance is asymmetric as the cost matrix C is asymmetric.

3.3 Heterogeneous Mating

It is known that VQX shows faster convergence than other crossovers; this may cause the premature convergence of genetic algorithms. To avoid it, we use a special type of mating used in [11]. In the mating, each individual is mated with one of its dissimilar individuals. Hollstien called this type of breeding a *negative assortive mating* [15]. The *heterogeneous mating* is done similarly to a selection method called crowding [16]. First, given an individual p_1 , m candidate individuals are selected from the population P by roulette-wheel selection. Among them, the most different one from p_1 is selected as p_2 . Hamming distance² is used for the distance measure. The heterogeneous mating improved the performance of VQX by slowing down the convergence of the genetic algorithm. It is notable that we could not found any synergy effect between the mating and other crossovers such as k -point crossover and uniform crossover in our experiments.

3.4 Properties of VQX

VQX has two notable properties:

- *Convexity* — Voronoi regions are convex³ (see [13] p. 330).
- *Diversity* — It has $\binom{n}{k}2^k$ crossover operators.

In VQX, genes in the chromosome are quantized into several groups by randomly selected Voronoi regions, and the gene values in the same group are inherited from the same parent. Therefore, the first property that Voronoi regions are convex implies that the gene groups of relatively short genic distance have high survival probabilities, i.e., strong linkages.

The other property means that VQX has a lot of crossover operators. The number of crossover operators affects the creativity of new schemata. The number of crossover operators of k -point crossover is $\binom{n-1}{k}$. For $n = 10000$ and $k = 12$, for example, VQX has about 10^{43} crossover operators, while k -point crossover has about 10^{39} . However, we should mention that we do not pursue the maximal number of crossover operators.

4 Experimental Results

The genetic algorithms used in this paper are steady-state hybrid genetic algorithms. Figure 5 shows the template. In the template, n is the problem size, m is the group size of heterogeneous mating, k is the crossover degree, and d_g is

² the number of different edges between two paths.

³ A set $S \in R^k$ is *convex* if $a, b \in S$ implies that $\alpha a + (1 - \alpha)b \in S$ for all $0 < \alpha < 1$.

```

1.  VGA( $n, m, k, d_g$ )
2.  {
3.      Initialize population  $P$ ;
4.      repeat {
5.           $p_1 \leftarrow \text{Selection}(P)$ ;
6.           $p_2 \leftarrow \text{MateSelection}(P, m, p_1)$ ;
7.           $o \leftarrow \text{VQX}(n, k, d_g, p_1, p_2)$ ;
8.           $o \leftarrow \text{Mutation}(o)$ ;
9.           $o \leftarrow \text{LocalImprovement}(o)$ ;
10.          $P \leftarrow \text{Replacement}(P, p_1, p_2, o)$ ;
11.     } until (stopping condition);
12.     return the best of  $P$ ;
13. }

```

Fig. 5. The steady-state hybrid genetic algorithm for SOP.

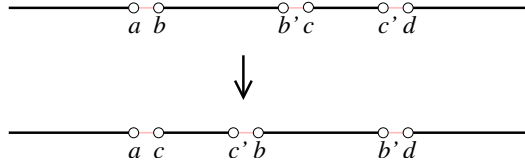


Fig. 6. An illustration of the path-preserving 3-exchange.

the genic distance. The two selected parents and the offspring are denoted by p_1 , p_2 , and o , respectively. The genetic operators and their parameters used in this paper are summarized in the following.

- Population Initialization — Initial solutions are generated at random, then the local improvement algorithm is applied to each of them. All the solutions in the population are feasible.
- Population Size — $|P| = 50$.
- Selection — Roulette-wheel selection, i.e., the fitness value f_i of the solution i is calculated as

$$f_i = (C_w - C_i) + (C_w - C_b)/4 \quad (2)$$

where C_i , C_w , and C_b are the costs of the solution i , the worst solution, and the best solution in the population, respectively. The fitness value of the best solution is five times as great as that of the worst solution in the population.

- Group Size of Heterogeneous Mating — $m = 3$.
- Crossover Degree — $k = 3$.
- Mutation — Five random feasible-path-preserving 3-exchanges are applied to each offspring with probability 0.1. Figure 6 shows a symbolic drawing of the exchange.

Table 1. The experimental results for ESC78 and ft70.*.

Graph (Bst-Kn)	GA	BK#/t	Best (%)	Avg (%)	$\sigma/\sqrt{t}$	Gen	Time (s)
ESC78 (18230)	DGA	1000/1000	18230 (0.000)	18230.00 (0.000)	0.00	223	2.68
	MGA	1000/1000	18230 (0.000)	18230.00 (0.000)	0.00	335	1.83
	VGA	1000/1000	18230 (0.000)	18230.00 (0.000)	0.00	115	0.91
ft70.1 (39313)	DGA	953/1000	39313 (0.000)	39315.75 (0.007)	0.39	268	4.40
	MGA	548/1000	39313 (0.000)	39351.03 (0.097)	1.35	2256	8.48
	VGA	1000/1000	39313 (0.000)	39313.00 (0.000)	0.00	629	6.27
ft70.2 (40419)	DGA	718/1000	40419 (0.000)	40421.26 (0.006)	0.59	710	7.66
	MGA	117/1000	40419 (0.000)	40424.45 (0.013)	0.68	601	3.48
	VGA	930/1000	40419 (0.000)	40419.18 (0.000)	0.02	1190	7.66
ft70.3 (42535)	DGA	526/1000	42535 (0.000)	42549.87 (0.035)	0.50	205	2.41
	MGA	619/1000	42535 (0.000)	42546.86 (0.028)	0.48	177	1.45
	VGA	909/1000	42535 (0.000)	42537.82 (0.007)	0.28	319	2.38
ft70.4 (53530)	DGA	405/1000	53530 (0.000)	53560.35 (0.057)	0.88	594	4.59
	MGA	12/1000	53530 (0.000)	53571.90 (0.078)	0.29	666	2.54
	VGA	618/1000	53530 (0.000)	53543.97 (0.026)	0.58	559	3.83

- Local Improvement — A simple path-preserving 3-Opt (pp-3-Opt) algorithm is used. In the algorithm, a path-preserving 3-exchanges of maximum gain is selected and performed repeatedly. The gain of an exchange, with Figure 6 as an example, is computed by

$$gain = c_{ab} + c_{b'c} + c_{c'd} - c_{ac} - c_{c'b} - c_{b'd} \quad (3)$$

where c_{pq} is the (p, q) element of the cost matrix. For efficient feasibility checking, a marking technique is used as the SOP labeling procedure in [5].

- Replacement — A variant of preselection [17] is used as in [12]. Each offspring is replaced with (i) its more similar parent if the offspring is better, (ii) the other parent if the offspring is better, (iii) the worst solution in the population, otherwise.
- Stopping Condition — Until 70 percent of the population converges with the same cost as the best solution. This takes account of the cases that more than one best solution of the same quality competes with each other.

The algorithms were implemented in C on Pentium III 1132 MHz running Linux 2.2.14. We tested on eighteen SOP instances taken from [18]. They are all instances that have more than seventy nodes.

Tables 1–3 compare the performance of VGA with DGA and MGA. VGA represents the genetic algorithms using Voronoi quantized crossover (VQX) with the genic distance assignment heuristic described in Section 3.2. DGA and MGA represent the genetic algorithms using distance preserving crossover (DPX) and

Table 2. The experimental results for kro124p.* and prob.100.

Graph (Bst-Kn)	GA	BK#/t	Best (%)		Avg (%)	$\sigma/\sqrt{t}$	Gen	Time (s)
kro124p.1 (39420)	DGA	357/1000	39420	(0.000)	39481.95 (0.157)	1.58	431	25.46
	MGA	565/1000	39420	(0.000)	39505.79 (0.218)	6.12	902	15.64
	VGA	930/1000	39420	(0.000)	39426.45 (0.016)	0.95	518	12.92
kro124p.2 (41336)	DGA	876/1000	41336	(0.000)	41344.27 (0.020)	0.70	529	27.96
	MGA	543/1000	41336	(0.000)	41566.05 (0.557)	12.77	1079	14.91
	VGA	789/1000	41336	(0.000)	41353.22 (0.042)	1.76	688	12.49
kro124p.3 (49449)	DGA	6/1000	49499	(0.000)	50035.24 (1.083)	9.16	3884	42.68
	MGA	78/1000	49499	(0.000)	50029.73 (1.072)	12.81	3051	17.05
	VGA	705/1000	49499	(0.000)	49582.64 (0.169)	6.27	1146	12.71
kro124p.4 (76103)	DGA	999/1000	76103	(0.000)	76103.27 (0.000)	0.27	227	11.75
	MGA	841/1000	76103	(0.000)	76138.68 (0.047)	2.61	298	7.00
	VGA	1000/1000	76103	(0.000)	76103.00 (0.000)	0.00	249	8.36
prob.100 (1190)	DGA	0/50	1197	(0.588)	1260.72 (5.943)	5.62	112869	5108
	MGA	1/50	1175	(-1.261)	1244.36 (4.568)	4.28	2165330	54166
	VGA	2/50	1163	(-2.269)	1255.86 (5.534)	5.85	122586	1767

maximum partial order/arbitrary insertion (MPO/AI)⁴ [4], respectively. DPX tries to generate an offspring that has equal Hamming distance to both of its parents, i.e., its aim is to achieve that the three Hamming distances between offspring and parent 1, offspring and parent 2, and parent 1 and parent 2 are identical. It was proposed originally for traveling salesman problem [19]. In MPO/AI, the longest common subsequence (maximum partial order) of the two parents is inherited to the offspring and the crossover is completed by repeatedly inserting arbitrary nodes (arbitrary insertion) not yet included into a feasible position of minimum cost. The same local improvement algorithm was used in all the genetic algorithms. In the tables, the frequency of finding solutions better than or equal to the best-known (BK#), the best cost (Best), average cost (Avg), group standard deviation ($\sigma/\sqrt{t}$), average generation (Gen), and average running time (Time) are presented. We got the results from 1000 ($= t$) runs on ESC78, ft70.*, kro124p.*, rbg1*, and 50 runs on prob.100, rbg2*, and rbg3*. The values (%) after the best and average costs represent the percentages above the best-known⁵. VGA outperformed other genetic algorithms for twelve instances, while DGA and MGA outperformed the others for four instances and one instance, respectively. VGA broke the best-known for prob.100, rbg323a, and rbg341a. All three genetic algorithms consumed comparable running time for all instances except prob.100, rbg341a, rbg358a, and rbg378a. The overall results show that VGA is the most efficient and stable among them.

⁴ Available at http://www.cs.cmu.edu/afs/cs.cmu.edu/user/chens/WWW/MP0AI_SOP.tar.gz.

⁵ Available at <http://www.idsia.ch/~luca/has-sop.html>.

Table 3. The experimental results for rbg*.

Graph (Bst-Kn)	GA	BK#/t	Best (%)		Avg (%)	$\sigma/\sqrt{t}$	Gen	Time (s)
rbg109a (1038)	DGA	956/1000	1038	(0.000)	1038.07 (0.007)	0.01	97	11.19
	MGA	177/1000	1038	(0.000)	1039.88 (0.181)	0.04	772	16.51
	VGA	953/1000	1038	(0.000)	1038.12 (0.011)	0.02	209	11.88
rbg150a (1750)	DGA	987/1000	1750	(0.000)	1750.04 (0.002)	0.01	77	31.14
	MGA	108/1000	1750	(0.000)	1752.63 (0.150)	0.03	331	33.12
	VGA	901/1000	1750	(0.000)	1750.30 (0.017)	0.03	216	34.20
rbg174a (2033)	DGA	994/1000	2033	(0.000)	2033.01 (0.001)	0.01	192	78.37
	MGA	623/1000	2033	(0.000)	2033.71 (0.035)	0.04	381	67.14
	VGA	927/1000	2033	(0.000)	2033.15 (0.007)	0.02	433	85.85
rbg253a (2950)	DGA	36/50	2950	(0.000)	2950.32 (0.011)	0.08	199	346
	MGA	47/50	2950	(0.000)	2950.08 (0.003)	0.05	155	222
	VGA	50/50	2950	(0.000)	2950.00 (0.000)	0.00	382	325
rbg323a (3141)	DGA	1/50	3141	(0.000)	3144.20 (0.102)	0.28	866	2559
	MGA	0/50	3142	(0.032)	3142.42 (0.045)	0.07	628	1281
	VGA	16/50	3140	(−0.032)	3141.94 (0.030)	0.13	1358	2515
rbg341a (2570)	DGA	0/50	2572	(0.078)	2575.30 (0.206)	0.33	1281	4262
	MGA	0/50	2571	(0.039)	2578.32 (0.324)	0.55	1686	3174
	VGA	12/50	2568	(−0.078)	2571.88 (0.073)	0.28	5620	10164
rbg358a (2545)	DGA	3/50	2545	(0.000)	2553.98 (0.353)	0.76	1890	7345
	MGA	0/50	2549	(0.157)	2555.24 (0.402)	0.54	17355	34675
	VGA	9/50	2545	(0.000)	2548.56 (0.140)	0.41	8640	24340
rbg378a (2816)	DGA	0/50	2819	(0.107)	2819.86 (0.137)	0.31	1065	7785
	MGA	2/50	2816	(0.000)	2818.96 (0.105)	0.22	3873	11669
	VGA	22/50	2816	(0.000)	2818.44 (0.087)	0.45	7814	33774

5 Conclusions

In this paper, we proposed a new hybrid genetic algorithm for the sequential ordering problem (SOP). It adopts a crossover, called Voronoi quantized crossover (VQX), on a complete graph representation. The crossover was modified by employing several new features for SOP. In the experiments, the proposed genetic algorithm outperformed state-of-the-art genetic algorithms for SOP. We suspect that the power of VQX is based on two main properties, convexity and diversity. The properties are believed to improve the performance of genetic algorithms by encouraging the survival probability and reproducibility of high-quality building blocks in the genetic process.

Acknowledgments. This work was partly supported by Optus Inc. and Brain Korea 21 Project. The RIACT at Seoul National University provided research facilities for this study.

References

1. N. Ascheuer, L. F. Escudero, M. Grottschel, and M. Stoer. A cutting plane approach to the sequential ordering problem (with applications to job scheduling in manufacturing). *SIAM Journal on Optimization*, 3:25–42, 1993.
2. L. F. Escudero, M. Guignard, and K. Malik. A Lagrangian relax-and-cut approach for the sequential ordering problem with precedence relationships. *Annals of Operations Research*, 50:219–237, 1994.
3. N. Ascheuer, M. Jünger, and G. Reinelt. A branch & cut algorithm for the asymmetric traveling salesman problem with precedence constraints. *Computational Optimization and Applications*, 17(1):61–84, 2000.
4. S. Chen and S. Smith. Commonality and genetic algorithms. Technical Report CMU-RI-TR-96-27, The Robotic Institute, Carnegie Mellon University, 1996.
5. L. M. Gambardella and M. Dorigo. An ant colony system hybridized with a new local search for the sequential ordering problem. *INFORMS Journal on Computing*, 12(3):237–255, 2000.
6. J. Holland. *Adaptation in Natural and Artificial Systems*. The University of Michigan Press, 1975.
7. Y. Davidor. Epistasis variance: Suitability of a representation to genetic algorithms. *Complex Systems*, 4:369–383, 1990.
8. D. I. Seo, Y. H. Kim, and B. R. Moon. New entropy-based measures of gene significance and epistasis. In *Genetic and Evolutionary Computation Conference*, 2003.
9. D. E. Goldberg. *Genetic Algorithms in Search, Optimization, Machine Learning*. Addison-Wesley, 1989.
10. D. I. Seo and B. R. Moon. A survey on chromosomal structures and operators for exploiting topological linkages of genes. In *Genetic and Evolutionary Computation Conference*, 2003.
11. D. I. Seo and B. R. Moon. Voronoi quantized crossover for traveling salesman problem. In *Genetic and Evolutionary Computation Conference*, pages 544–552, 2002.
12. T. N. Bui and B. R. Moon. A new genetic approach for the traveling salesman problem. In *IEEE Conference on Evolutionary Computation*, pages 7–12, 1994.
13. A. Gersho and R. M. Gray. *Vector Quantization and Signal Compression*. Kluwer Academic Publishers, 1992.
14. C. Anderson, K. Jones, and J. Ryan. A two-dimensional genetic algorithm for the Ising problem. *Complex Systems*, 5:327–333, 1991.
15. R. B. Hollstien. *Artificial Genetic Adaptation in Computer Control Systems*. PhD thesis, University of Michigan, 1971.
16. K. De Jong. *An Analysis of the Behavior of a Class of Genetic Adaptive Systems*. PhD thesis, University of Michigan, 1975.
17. D. Cavicchio. *Adaptive Search Using Simulated Evolution*. PhD thesis, University of Michigan, 1970.
18. TSPLIB.
<http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/>.
19. B. Freisleben and P. Merz. New genetic local search operators for the traveling salesman problem. In *Parallel Problem Solving from Nature*, pages 890–900. 1996.

An Optimization Solution for Packet Scheduling: A Pipeline-Based Genetic Algorithm Accelerator

Shiann-Tsong Sheu, Yue-Ru Chuang, Yu-Hung Chen, and Eugene Lai

Department of Electrical Engineering, Tamkang University,
Tamsui, Taipei, Taiwan 25137, R.O.C.
stsheu@ee.tku.edu.tw, g8350021@tkgis.tku.edu.tw

Abstract. The dense wavelength division multiplexing (DWDM) technique has been developed to provide a tremendous number of wavelengths/channels in an optical fiber. In the multi-channel networks, it has been a challenge to effectively schedule a given number of wavelengths and variable-length packets into different wavelengths in order to achieve a maximal network throughput. This optimization process has been considered as difficult as the job scheduling in multiprocessor scenario, which is well known as a NP-hard problem. In current research, a heuristic method, genetic algorithms (GAs), is often employed to obtain the near-optimal solution because of its convergent property. Unfortunately, the convergent speed of conventional GAs cannot meet the speed requirement in high-speed networks. In this paper, we propose a novel hyper-generation GAs (HG-GA) concept to approach the fast convergence. By the HG-GA, a pipelined mechanism can be adopted to speed up the chromosome generating process. Due to the fast convergent property of HG-GA, which becomes possible to provide an efficient scheduler for switching variable-length packets in high-speed and multi-channel optical networks.

1 Introduction

The fast explosion of Internet traffic demands more and more network bandwidth day by day. It is evident that the optical network has become the Internet backbone because it offers sufficient bandwidth and acceptable link quality for delivering multimedia data. With the dense wavelength division multiplexing (DWDM) technique, an optical fiber can easily provide a set of parallel channels, each operating at different wavelengths [1], [2]. In each channel, the *statistical multiplexing* technique is used to transport data packets from different sources to enhance the bandwidth utilization. However, this technique incurs the complicated packet scheduling and channel assignment problem in each switching node underlying the tremendous wavelengths. Hence, it is desired to design a faster and more efficient scheduling algorithm for transporting variable-length packets in high-speed and multi-channel optical networks.

So far, many scheduling algorithms for multi-channel networks have been proposed and they are basically designed under two different network topologies: the star-based WDM network and the optical interconnected network. The

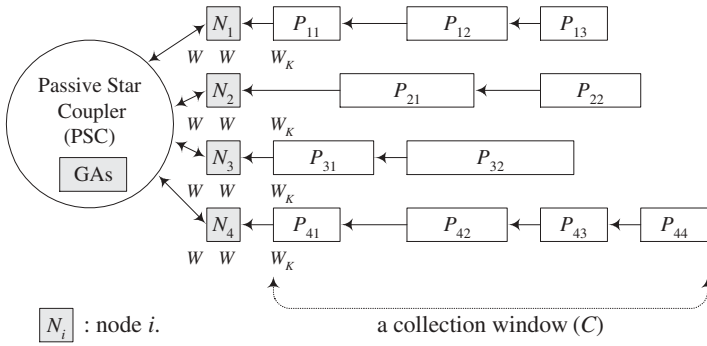


Fig. 1. An example illustrates the packet scheduling problem in a star-based network.

star-based network consists a passive start coupler (PSC), which is in charge of coupling packets/messages from different wavelengths and broadcasting all wavelengths to every connected node. The star-based network is often built for local area network due to its centralized control [3], [4], [5]. On the contrary, the other switching component, optical cross connect (OXC), performs efficiently in the space, timing and wavelength switching/converting, and thus, is often used in the optical backbone network [6]. An example shown in Fig. 1 is given to illustrate the scheduling problem in a star-based network while a number of packets with variable lengths from four nodes (N) arriving at PSC, in which there are K parallel channels per fiber (in usual, the number of nodes is smaller than wavelengths.). In this figure, notation P_{ij} is denoted as the j -th packet from node i . In order to minimize the total packet switching delay and maximize the channel utilization, these packets should be well scheduled in K available channels. In the literatures, the scheduling of sequencing tasks for multiprocessor has been addressed extensively and proved as an NP-hard problem [7]. Similarly, the packet scheduling and wavelength assignment problem under constraint of sequence maintenance is also well known as a difficult-to-solve issue. We believe that it is hard to design a real-time scheduling algorithm to resolve the NP-hard problem by general heuristic schemes.

In the past few years, Genetic Algorithms (GAs) have received considerable attention regarding their potential as an optimization technique for complex problems and have been successfully applied in the areas of scheduling, matching, routing, and so on [7], [8], [9], [10]. The GAs mimic the natural genetic ideas to provide excellent evolutionary processes including *crossover*, *mutation* and *selection*. Although GAs have been already applied in many scheduling and sequencing problems, the slow convergence speed of typical GAs limits the possibility of applying them in real-time systems (e.g., network optimization problems often require short response time for each decision.). It has been a major drawback for this mechanism. To overcome the drawback, in this paper, we proposed a pipeline-based hyper-generation GA (HG-GA) mechanism for solving this tough packet schedule problem. The proposed HG-GA mechanism adopts

a hyper-generation concept to break the thumb rule of performing crossover on two chromosomes of the same generation. By generating more 'better' chromosomes than the general GA (G-GA) mechanisms within a limited time interval, the HG-GA mechanism can improve significantly in convergence speed, as compared with the G-GA mechanisms. Therefore, the proposed HG-GA mechanism makes the possibility of employing GAs to solve the complicated optimization problem in any real-time environment.

The rest of paper is organized as following. The general GAs packet scheduler (G-GAPS) is introduced in Section 2. In Section 3, we describe the proposed hyper-generation GAs packet scheduler (HG-GAPS) and analyze the precisely generating time of each offspring chromosome. Section 4 provides the simulation comparison to compare the performance difference between these two technologies. Finally, some conclusion remarks are given in Section 5.

2 General Genetic Algorithms Packet Scheduler (G-GAPS)

The G-GA mechanisms applied in industrial engineering contain three main components: the Crossover Component (XOC), Mutation Component (MTC) and Selection Component (SLC), as show in Fig. 2. A process to apply the G-GA mechanisms in solving the optimization problem of packet scheduling and wavelength assignment in networks is named as the general GAs packet scheduler (G-GAPS) [11], [12]. Basically, the G-GAPS needs a collection window (C) for collecting packets. As soon as the scheduling process is executed, a new collection window will be started. This workflow will smooth the traffic flow if the window size is properly selected.

2.1 Definition

In G-GAPS, packets destined to the same output port are collected and permuted for all available wavelengths to form a *chromosome* (i.e., a chromosome presents a kind of permutations), in which each packet is referred to a *gene* [11], [12]. The example showed in Fig. 3 demonstrates that a set of collected packets (P) with different lengths (l) and time stamps (T) are permuted for two available wavelengths (W_1 and W_2) to form a chromosome. A number of chromosomes, denoted as N , will be first generated to form the *base generation* (also called the *first generation*). Following the sequencing, each arrival packet is associated with a time stamp and all permutations will follow these time stamps of packets to be executed as in the scheduling principle. Therefore, the problem becomes how to decide the switch timing and the associated wavelength of a packet so that the precedence relations of the same connection can be maintained and the total required switching time (TRST) of schedule can be minimized. More definitely, the TRST presents the maximum scheduled queue length of packets assigned into different wavelengths. Therefore, we can define the TRST(j) by the formula:

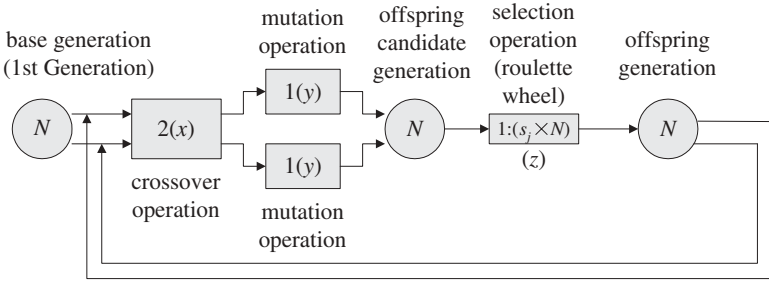


Fig. 2. The flow block diagram of the G-GAPS.

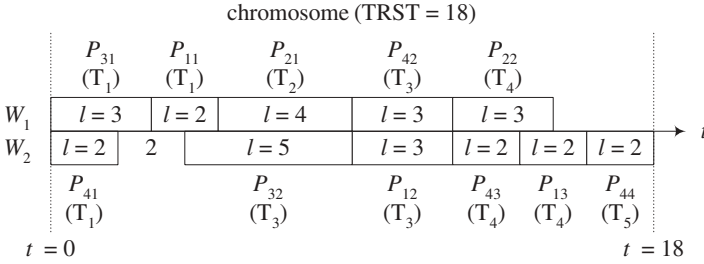


Fig. 3. An example presents a permutation to form a chromosome.

$$TRST(j) = \max\{trst(W_j(1)), trst(W_j(2)), \dots, trst(W_j(K))\}. \quad (1)$$

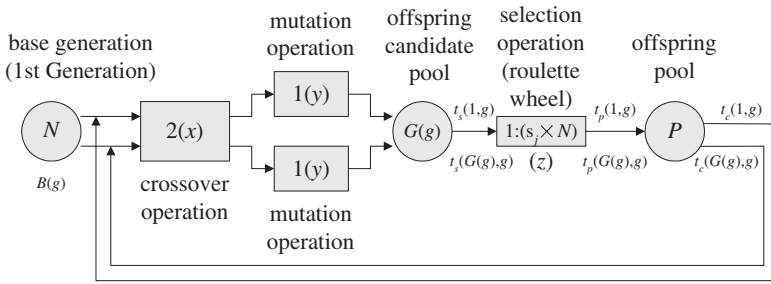
where j is the j -th chromosome, and $trst(W_j(k))$ is the TRST for these packets scheduled in the k -th wavelength of the j -th chromosome. Here we assume an optical fiber carries K wavelengths.

2.2 The Fitness Function

The *fitness function*, denoted as Ω in the G-GAPS is defined as the *objective function* that we want to optimize. It is used to evaluate chromosomes during selection operation to determine which offspring should be remained as the parents for the next generation. The objective function in the scheduling is the TRST and it is often converted into maximization form. Thus, the fitness value of the j -th chromosome, denoted as $\Omega(j)$, is calculated as following:

$$\Omega(j) = \Psi_1^{worst} - TRST(j). \quad (2)$$

where $\Psi_1^{worst} = \sum_u \sum_v l_{uv}$ represents the worst TRST in the first generation (i.e., all packets are scheduled in one wavelength.). Therefore, the optimal schedule will be the chromosome with the largest fitness value denoted as Ω^{opt} .



P : The accumulated number of offspring chromosomes generated from selection operation.

Fig. 4. The flow block diagram of the HG-GAPS.

2.3 Implementation of the Genetic Algorithms

In G-GAPS, each crossover operation selects two chromosomes from the same generation and generates two new offspring chromosomes as the candidates for the next generation. These candidate offspring will involve in mutation operation and selection operation according to their mutation probabilities (P_m) and fitness values, respectively [11], [12].

In the implementation, we simply assume the number of chromosomes in the base generation, say N , is even. Let P_c and P_m denote as the crossover and mutation probabilities, respectively. According to roulette wheel method, the selected probability of the j -th chromosome is $s_j = \Omega(j) / \sum_{r=1}^{r=N} \Omega(r)$.

3 Hyper-generation GAs Packet Scheduler (HG-GAPS)

Basically, the G-GAPS is a generation-based scheme, which processes chromosomes generation by generation. In this scheme, the population size in each generation is kept to N . It means that the selection operation is only triggered when all crossovers and mutations on the chromosomes in a generation are completed, and, also all fitness values of N chromosomes are calculated to support the roulette wheel method. These restraints cause considerable waiting time for propagating good chromosomes to the next generation. For general optimization problems, they do not require quick response time, thus, such a batch behavior will work well to provide an acceptable solution. However, the features of long waiting time and slow convergence speed definitely block the G-GAPS to be a suitable solution for real-time systems. In this section, we will introduce a pipeline-based mechanism, named hyper-generation GAPS (HG-GAPS), to overcome the potential drawbacks of G-GAPS.

As shown in Fig. 4, the key feature of the HG-GAPS is to adopt the pipeline concept and to discard the generation restraint to accelerate convergence speed. From Fig. 4, at the candidate state after mutation operation, the number of

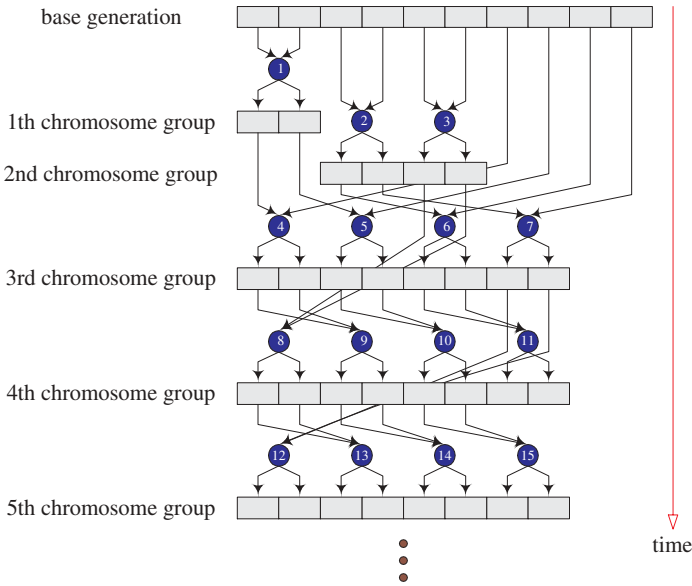


Fig. 5. An example demonstrates the concepts of the chromosome groups and the hyper-generation crossovers in the HG-GAPS when there are $N = 10$ chromosomes in the base generation.

offspring chromosomes is a function of g , which presents the number of 'chromosome group', as shown in Fig. 5. HG-GAPS uses 'chromosome group' concept instead of 'generation' concept to break the crossover limitation in the same generation (i.e., batch operation). In other words, the member of a chromosome group may be generated from parent mating with parent, parent mating with offspring, or offspring mating with offspring.

3.1 Hardware Block Diagram of the HG-GAPS

The detailed HG-GAPS hardware block diagram is designed and shown in Fig. 6. As mentioned before, all arrival packets destined to the same outlet in a collection window are gathered and queued in a Shared Memory. Each of them is tagged with a global time stamp. In the Shared Memory, packets with the same time stamp are linked together. At the end of collection window, packets of the same link are concurrently assigned into K wavelengths through an $M \times K$ switch in a random manner to form a chromosome (where the number of the inlets (I) is M , and K presents the number of wavelengths (W) in a fiber.). This procedure is repeated in the Chromosome Generator until a number of N chromosomes are generated to form the base generation. To promote a more efficient scheduling process, the first two newborn chromosomes in the base generation will be immediately forwarded into the XOC once they are generated.

addition, it also provides a random number for some random processes in the GA operation. After the crossover operation, two mated chromosomes are separately forwarded into the MTCs. Meanwhile, they are also bypassed to Fitness components, one for each, to calculate their fitness values and then to queue in a temp pool (i.e., the Filter). As the pairs of the original parents and the produced offsprings are all stored in the temp pool, two chromosomes with better fitness values will be selected and pushed into the Offspring Candidate Pool for elitism, which is similar to the concept of enlarged sampling space [8].

Finally, the SLC equips two Accumulators: one is used to accumulate the fitness values of the current chromosome group and the other is used to count the number of chromosomes queued in this group. Both of them dedicate the necessary information for the roulette wheel method adopted in SLC. When the last chromosome queued in Offspring Pool is forwarded to the XOC, the Offspring Candidate Pool will pass whole group of chromosomes into SLC by selection and duplication. (That is why we use the '*chromosome group*' to replace '*generation*' as a set of chromosomes to calculate the selection probability of the chromosome in the HG-GAPS.) Thus these two Accumulators will be reset for the next group. As soon as the offspring is produced in the Offspring Pool, it can be as a new parent for the next GA cycle.

4 Simulation Model and Results

4.1 Simulation Model

In the simulation, we construct the GAPS simulation model with several realistic system parameters: the number of time units consumed by each XOC ($= x$), MTC ($= y$) and SLC ($= z$). Besides, there are N chromosomes in both base generations in the G-GAPS and the HG-GAPS. During simulating, we set the time units to be $N = 10$, $x = 2$, $y = 1$ and $z = 2$. (Here, we assume the crossover and the selection operations are more complicated than the mutation operation.) The simulation probabilities of the crossover (P_c) and the mutation (P_m) operations are 0.9 and 0.05, respectively. To simplify the model, we consider the deterministic service rate in each wavelength is measured in the preset time units. The traffic arrival rate of a wavelength in each input fiber is following a Poisson distribution with a mean λ . The packet length is following an exponential distribution with a mean L in the preset time units. The number of the wavelengths in each input or output fiber is K . Thus, the total traffic load Λ is equal to $K \times \lambda \times L$. Furthermore, in order to simulate a real-time system, we fix the scheduling time period to enforce the G-GAPS and the HG-GPAS to output its current optimal schedule within the due time.

4.2 Simulation Results

Fig. 7 shows the average TRSTs derived from the G-GAPS and the HG-GAPS under the variable collection windows (C) and fixed the scheduling time interval

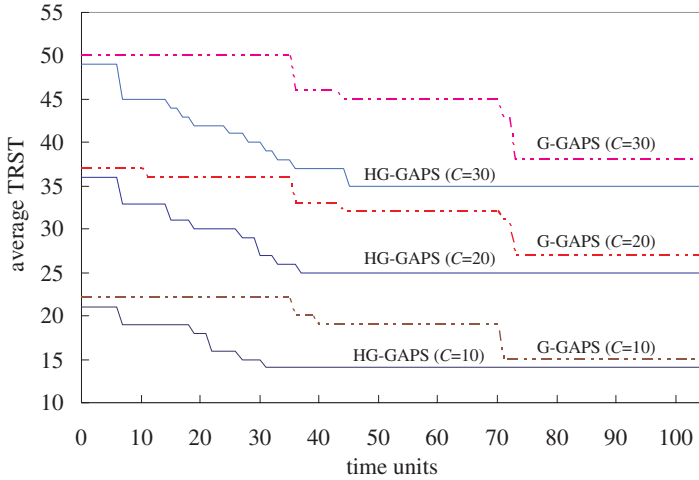


Fig. 7. The average TRSTs are simulated under the different collection window sizes (C) from 10 to 30 time units, when $K = 8$, $\Lambda = 8$ and $L = 5$.

at $K = 8$, $\Lambda = 8$ and $L = 5$ time units. Here, we consider the collection window varying from 10 to 30 time units and the scheduling time period is fixed at 105 time units. In Fig. 7, we can see that the G-GAPS will generate a schedule with a smaller TRST as soon as a generation is completed. That is, the improvements on TRST in the G-GAPS will occur at the time units of 35, 70 and 105. On the contrary, our HG-GAPS starts to minimize the TRST within a short period and obtains the near-optimal TRST at approximate 35 time units. In addition, we also note that under a larger collection window size, the more gain in the decrease TRST will be obtained by the HG-GAPS comparing to the G-GAPS.

Fig. 8 presents the difference in the accumulated chromosome generating rates between the G-GAPS and the HG-GAPS. During the same scheduling time period of 105 time units, the G-GAPS evolves three generations (including the first generation) and only generates 30 chromosomes. On the contrary, the HG-GAPS requires a shorter period to increase the generating rate than the G-GAPS due to its chromosome group and the pipeline concepts. HG-GAPS does not only have the advantage in continuously generating offsprings during a short period, but also keeps the advantage in having a large candidates space for selection operation. Therefore, HG-GAPS can evolve 56 chromosomes during 105 time units.

Fig. 9 shows the consecutively snap shots during a period of 500 time units with a randomly selection from the whole simulation run. We set both of the scheduling time period and the collection windows to be 50 time units. The other system parameters are set as following: $K = 8$, $\Lambda = 6.4$ and $L = 5$ time units. Within a limited scheduling window, the HG-GAPS always provides a

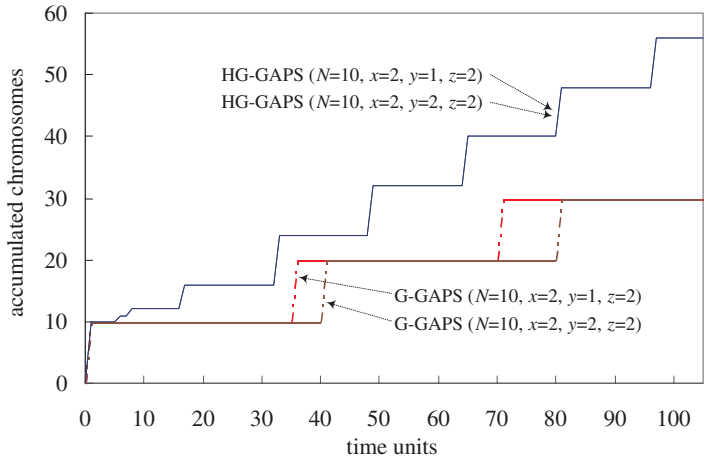


Fig. 8. An illustration to present the accumulated chromosomes generated by the G-GAPS and the HG-GAPS under the different system parameters.

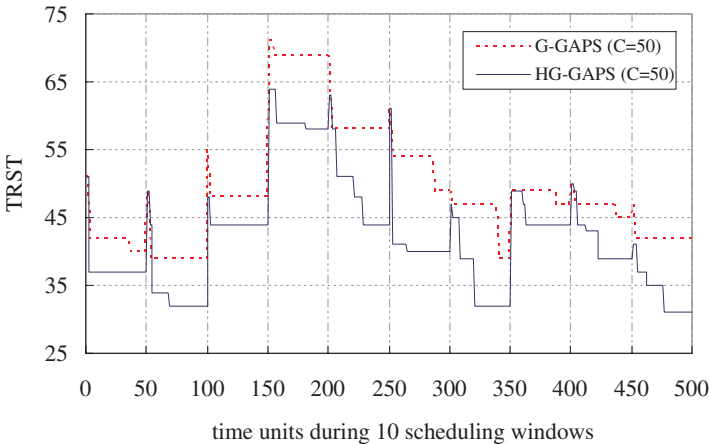


Fig. 9. A comparison between the G-GAPS and the HG-GAPS in the TRSTs of the chromosomes during 10 consecutively scheduling windows.

smaller TRST than the one from the G-GAPS. In fact, if we further shorten the scheduling window to conform a real-time situation, the performance difference between these two GAPSs will become more obvious. During a very short period, the HG-GAPS can generate a scheduling result to approach to a near-optimal solution, but the G-GAPS cannot. In a real continuous transmission environment, a larger TRST from the data transmission will defer the following

scheduling tasks. Thus, the difference between the accumulated TRSTs from the G-GAPS and the HG-GAPS will become larger and larger as the time expired, and the packet loss is also getting larger due to the buffer overflow. Therefore, we conclude that the proposed HG-GAPS cannot only indeed provide a significant improvement in solving an optimization problem, but also further support more complicated real-time systems.

5 Conclusions

In this paper, a novel and faster convergent GAPS mechanism, the hyper-generation GAPS (HG-GAPS) mechanism, for scheduling variable-length packets in high-speed optical networks was proposed. It is a powerful mechanism to provide a near-optimal solution for scheduling an optimization problem within a limited response time. This proposed HG-GAPS utilizes the hyper-generation and pipeline concepts to speed up the way of generating chromosomes and to shorten the evolutionary consuming time produced from traditional genetic algorithms. From the simulation results, we proved that the HG-GAPS is indeed more suitable for solving the complex optimization problems, such as the packets scheduling and the wavelength assignment problem in a real-time environment.

References

1. Charles A. Brackett: Dense Wavelength Division Multiplexing Networks: Principles and Applications. *IEEE J. Select. Areas Communication*, Vol. 8, No. 6, pp. 948–964, August (1990).
2. Paul Green: Progress in Optical Networking. *IEEE Communications magazine*, Vol. 39, No. 1, pp. 54–61, January (2001).
3. F. Jia, B. Mukherjee, J. Iness: Scheduling Variable-length Messages in A Single-hop Multichannel Local Lightwave Network. *IEEE/ACM Trans. Networking*, Vol. 3, pp. 477–487, August (1995).
4. J. H. Lee, C. K. Un: Dynamic Scheduling Protocol for Variable-sized Messages in A WDM-based Local Network. *J. Lightwave Technol.*, pp. 1595–1600, July (1996).
5. Babak Hamidzadeh, Ma Maode, Mounir Hamdi: Efficient Sequencing Techniques for Variable-Length Messages in WDM Network. *J. Lightwave Tech.*, Vol. 17, pp. 1309–1319, August (1999).
6. Sengupta S., Ramamurthy R.: From Network Design to Dynamic Provisioning and Restoration in Optical Cross-connect Mesh Networks: an Architectural and Algorithmic Overview. *IEEE Network*, Vol. 15, Issue 4, pp. 46–54, July-Aug (2001).
7. Edwin S.H. Hou, Nirwan Ansari, Hong Ren: A Genetic Algorithm for Multiprocessor Scheduling. *IEEE Transaction on Parallel and Distributed Systems*, Vol. 5, No. 2, February (1994).
8. Mitsuo Gen, Runwei Cheng: *Genetic Algorithms and Engineering Design*. Wiley Interscience Publication, (1997).
9. J. S.R. Jang, C. T. Sun, E. Mizutani: *Neuro-Fuzzy Soft Computing*. Prentice-Hall International, Inc., Chapter 7.
10. D. E. Goldberg: *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, Reading, MA, (1989).

11. Shiann-Tsong Sheu, Yue-Ru Chuang, Yu-Jie Cheng, Hsuen-Wen Tseng: A Novel Optical IP Router Architecture for WDM Networks. In Proceedings of IEEE ICOIN-15, pp. 335–340, (2001).
12. Shiann-Tsong Sheu, Yue-Ru Chuang: Fast Convergent Genetic Algorithm for Scheduling Variable-Length Packets in High-speed Multi-Channel Optical Networks. Submitted to IEEE Transaction on Evolutionary Computation, (2002).

Generation and Optimization of Train Timetables Using Coevolution

Paavan Mistry and Raymond S.K. Kwan

School of Computing, University of Leeds
Leeds LS2 9JT, United Kingdom
{paavan,rsk}@comp.leeds.ac.uk

Train timetabling is a process of assigning suitable arrival and departure times to trains at the stations they visit and at key track junctions. It is desirable that the timetable focusses on passenger preferences and is operationally viable and profitable for the Train Operating Companies (TOCs). Many hard and soft constraints need to be considered relating to the track capacities, set of trains to be run on the network, platform assignments at stations and passenger convenience. In the UK, train timetabling is mainly the responsibility of a single rail infrastructure operator - Network Rail. The UK rail network has a structure that is complex to integrate, which makes it difficult to achieve regularised train timetables that are common in many European countries. With a large number of independent TOCs bidding for slots to operate over limited capacities, the need for an efficient and intelligent computer-aided tool is obvious. This work proposes a Cooperative Coevolutionary Train Timetabling (CCTT) algorithm concerned with the automatic generation of planning timetables, which still demands a high degree of accuracy and optimization for them to be useful.

Determining the departure times of the train trips at their origins is the most critical step in the timetabling process. Timings of the train trips en route can be computed from the departure times. Pathing is the time added to or removed from a train's journey from one station to another. The amount of duration a train stops at the station is the dwell-time. Along with the departure and arrival times at every station, a train's journey also needs to determine track and platform/siding utilisation from origin to destination.

The idea of parallel evolution of problem subcomponents that interact in useful ways to optimize complex higher level structures was introduced by [3]. The advantages of such decomposition are independent representation and evolution of interacting subcomponents that facilitate an efficient concentrated exploration of the search space. The decision variables of the train timetabling problem are substructured into coevolving subpopulations - the departure times (P_d), scheduled runtime and dwell-time patterns (P_p) and capacity usage (P_c). Departure time of the trains being key to timetable generation, is evolved by Evolution Strategy [2]. An adaptive mutation strategy is used to control the trains' departure time evolution with a higher probability for finer mutations. Scheduled runtime of a train is the normal travel time of a train combined with variations to the travel time during a train's journey. Switching between high and low scheduled runtimes and dwell-times for trains is performed through a binary representation. Hence, P_p is evolved through a Genetic Algorithm [1]. The

rail network being considered assumes a single track system (one track in each direction) between stations with two platforms available at each station. This network set-up facilitates platform allocation and helps identify constraint violations. With either of the two platforms to be utilised by a train at each station, P_c evolves using a simple GA framework using binary chromosomes. The individual being evaluated and the representatives from collaborating populations generate a complete timetable. A greedy collaborator selection method [4] is undertaken. The individual being evaluated is assigned a fitness proportional to that of the complete timetable. The fitness function identifies and penalizes hard and soft constraint violations at the conflict points.

We run the algorithm with different random seeds for 5 times and the results achieved after 1000 iterations by CCTT are promising (shown in table 1). Considering the use of the same cost function, the quality of results i.e. the exploration of search space is better than those from a two-phase Simulated Annealing (SA) algorithm similar to the Planning Timetable Generator (PTG), which is a sophisticated train timetable planning tool developed by AEA Technology, Rail.

Table 1. Test Results from an average of 5 runs of the algorithm

Test Case	SA			CCTT		
	Best Fitness	Avg. Fitness	Time (sec)	Best Fitness	Avg. Fitness	Time (sec)
T-50	4064	4368	3.84	3395	3857	4.79
T-80	6064	6965	5.73	5575	6276	7.13

This research is on-going. The next phase of research will further refine the collaborative coevolution approach with further experiments and testing using real-world data sets.

References

1. D. E. Goldberg (1989). Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley.
2. N. Hansen and A. Ostermeier (2001). Completely Derandomized Self-Adaptation in Evolution Strategies. *Evolutionary Computation*, 9(2):159–195. MIT Press.
3. M. A. Potter and K. A. De Jong (1994). A Cooperative Coevolutionary Approach to Function Optimization. In *Proceedings of Third Conference on Parallel Problem Solving from Nature*, pages 249–257, Jerusalem, Israel. Springer-Verlag.
4. R. P. Weigand, W. C. Liles and K. A. De Jong (2001). An Empirical Analysis of Collaboration Methods in Cooperative Coevolutionary Algorithms. In *Proceedings of the Genetic Evolutionary Computation Conference (GECCO-2001)*, pages 1235–1242, San Fransisco, California, USA. Morgan Kaufmann.

Chromosome Reuse in Genetic Algorithms

Adnan Acan and Yüce Tekol

Computer Engineering Dept., Eastern Mediterranean University, Gazimagusa,
T.R.N.C. Mersin 10, TURKEY

adnan.acan@emu.edu.tr, ytekol@ieee.org

Abstract. This paper introduces a novel genetic algorithm strategy based on the reuse of chromosomes from previous generations in the creation of offspring individuals. A number of chromosomes of above-average quality, that are not utilized for recombination in the current generation, are inserted into a library called the chromosome library. The main motivation behind the chromosome reuse strategy is to trace some of the untested search directions in the recombination of potentially promising solutions. In the recombination process, chromosomes of current population are combined with the ones in the chromosome library to form a population from which offspring individuals are to be created. Chromosome library is partially updated at the end of each generation and its size is limited by a maximum value. The proposed algorithm is applied to the solution of hard numerical and combinatorial optimization problems. It outperforms the conventional genetic algorithms in all trials.

1 Introduction

Genetic algorithms (GA's) are biologically inspired search procedures that have been successfully used for the solution of hard numerical and combinatorial optimization problems. Since their introduction by John Holland in 1975, there has been a great deal on the derivation of various algorithmic alternatives of the standard implementation toward a faster and better localization of optimal solutions. In all these efforts, mechanisms of natural evolution developed over millions of years have become the main source of inspiration. The power and success of GA's is mainly achieved by the diversity of individuals of a population which evolve following the Darwinian principle of "survival of the fittest". In the standard implementation of GA's, the diversity of individuals is achieved using the genetic operators mutation and crossover which facilitate the search for high quality solutions without being trapped into local optimal points [1], [2], [3], [4].

In order to determine the most efficient ways of using GA's, many researchers have carried out extensive studies to understand several aspects such as the role and types of selection mechanism, types of chromosome representations, types and application strategies of the genetic operators, memory-based approaches, parallel implementations, and hybrid algorithms. In particular, several studies

were made concerning the development of problem-specific hybrids combining genetic algorithms with other intelligent search methods, and it has been demonstrated by thousands of applications that these approaches provide better results than the conventional genetic algorithms on very difficult problems [5], [6], [7], [8].

Among many different improvement efforts, memory-based approaches have also been studied and successfully applied for the solution difficult problems. Memory-based approaches aim to improve the learning performance of GAs by reintroducing chromosomes of previous generations into the current population, and their fundamental inspiration comes from redundancy within genetic material in natural biology and intelligent search methods which use the experience-based knowledge developed during the search to decide on the new search directions. In memory-based implementations, information stored within a memory is used to adapt the GAs behavior either in problematic cases where the solution quality is not improved over a number of iterations or to provide further directions of exploration and exploitation. Memory in GAs can be provided internally (within the population) or externally (outside the population) [9].

The most common approaches using internal memory are polyploidy structures and polygenic inheritance. Polyploidy structures in combination with dominance mechanisms use redundancy in genetic material by having more than one copy of each gene. When a chromosome is decoded to determine the corresponding phenotype, the dominant copy is chosen. By switching between copies of genes, the GA can adapt faster to changing environments and recessive genes are used to provide information about fitness values from previous generations [10], [11], [12]. Polygenic inheritance is based on the idea that a trait can depend on more than one gene or gene pair. In this case, the more gene pairs involved in the calculation of a trait, the more difficult it is to distinguish between various phenotypes. This is certainly a situation which smooths the evolution in a variable environment [13], [14].

External memory implementations store specific information and reintroduce that information into the population at a later moment. In most cases, this means that individuals from memory are put into the initial population of a new or restarted GA [15]. Case-based memory approaches, which is actually a long term elitism, is the most typical form of external memory implemented in practice. In general, there are two kinds of case-based memory implementations: in one kind, case-based memory is used to re-seed the population with the best individuals from previous generations when a change in the variable problem domain takes place [15]. A different kind of case-based memory stores both problems and solutions [16], [17]. When GA has to solve a problem similar to problems in its case-based memory, it uses the stored solutions to seed the initial population. Case-based memory aims to increase the diversity by reintroducing individuals from previous generations and achieves exploitation by reintroducing individuals from case-based memory when a restart from a good initial solution is required.

This paper introduces a novel external memory-based genetic algorithms strategy based on the reuse of chromosomes from previous generations in the creation of offspring individuals. At the end of each generation, a number of potentially promising chromosomes, based on their fitness values, are inserted into a library, called the chromosome library. Basically, starting from any point in the solution space, it is possible to form a path to an optimal solution over many different alternatives. Consequently, chromosome reuse aims to trace untested possibilities in the recombination of potentially promising solutions. Those individuals having a fitness value above a threshold, that are not used in the current recombination process, are selected for insertion into the chromosome library. During the recombination process, chromosomes of current population are combined with the ones in the chromosome library to form a population from which offspring individuals are to be created. The size of the chromosome library is limited by a maximum value and in case of excessive insertions, only the best individuals within the limits are accepted. The proposed algorithm is applied to the solution of hard numerical and combinatorial optimization problems. The obtained results demonstrate the superiority of the proposed approach over the conventional genetic algorithms.

The idea of reusing some chromosomes of previous generations, in the formation of offspring individuals, arises from a well-known fact in intelligent search algorithms: a search process has to make frequent backtracks or restarts to find a path to an optimal solution [18], [19]. This is because, an alternative search direction that may not be seen attractive at some point, due to more promising alternatives or due to many alternatives, may provide a link to an optimal solution with smaller number of computational steps. This idea is illustrated with a simple example as follows:

Assume that we want to maximize the objective function $f(x) = x^2$, $x \in [0, 1]$, using 8-bit binary encoding. Certainly, $f(x)$ takes its maximum value for an individual $p^* = 11111111$. Now, consider the following individuals $p1 = 00011111$, $p2 = 11100000$, and $p3 = 10000001$. Due to fitness-based selection procedures, it is obvious that $p2$ and $p3$ will produce much more offspring than $p1$ for the next generation. In addition to that, the number of recombinations between $p2$ and $p3$ will be greater than the ones between $p1$ and $p2$, and between $p1$ and $p3$. However, as can be seen from the structures of $p1$ and $p2$, a one-point crossover between the two from the position $j = 4$ will produce the optimal solution. Hence, it is worth to store chromosomes like $p1$ for a while to give them a chance for recombination with high quality individuals to provide a shorter path to an optimal solution. It is also important to note that, the individuals like $p1$ which can be accessed from the chromosome library are computationally free because their structure and fitness values are known from previous generations.

As explained with the above particular examples, in the recombination of two potential solutions, there are lots of possibilities and only a few of them are randomly tried due to restrictions of fitness-based selection procedures and the population size. In fact, for a binary encoding of length l , two individuals can be recombined in $(l - 1)$ different ways using the 1-point crossover. The number

of offspring that can be produced with 2-point crossover is $l(l-1)$, whereas this number using the uniform crossover is 2^k , $k \leq l$, where k is the number of positions where the two parents are different. Obviously, since the individuals of the current generation are completely replaced by their offspring, there is no way to retry another recombination operation with these individuals unless they are reproduced in future generations. In theoretical models of genetic algorithms, the branching process in genetic evolutionary search is explained by the schema theorem which is based on hyperplane sampling where the convergence process is modelled by increasingly more frequent sampling from high fitness individuals by crossover and mutation acting as a background operator to prevent premature convergence. In this respect, the use of the chromosome library will help the search process by providing additional intensification and diversification alternatives, through potentially promising untried candidates, at all stages of the search process. To clarify these points by experimental analysis, some statistical results for fitness-based selection behavior are given in section 2.

This paper is organized as follows. The statistical bases of chromosome reuse idea are illustrated in section 2. Algorithmic description of GAs with chromosome reuse strategy is given in section 3. Section 4 covers the case studies for numerical and combinatorial optimization problems. Finally, conclusions and future research directions are specified in section 5.

2 Statistical Reasoning on the Chromosome Reuse Strategy

The roulette-wheel and the tournament selection methods are the two most commonly used selection mechanisms in genetic algorithms. Both of these selection methods are fitness-based and they aim to produce more offspring from those high-fitness individuals. However, these selection operators leave a significant number of individuals having close to average fitness value unused in the sense that these individuals don't take part in any recombination operation. The idea of chromosome reuse is based on the fact that a significant percentage of these unused individuals have above average fitness values and they should not be just wasted. On the one hand, their reuse will provide additional intensification and diversification capabilities to the evolutionary search process. On the other hand, the use of the individuals in the chromosome library brings no extra computational cost. This is because, the structure and fitness values of these individuals are already known. When these individuals are reused, it is possible to localize an optimal solution over a shorter computational path as exemplified in Section 1 and as demonstrated by experimental evaluations in Section 4.

In order to understand the above reasoning more clearly, let's take the minimization problem for the Ackley's function of 20 variables [20]. A genetic algorithm with 200 individuals, uniform crossover with a crossover rate 0.7 and a mutation rate 0.01 is considered. Since it is more commonly used, the tournament selection operator is selected for illustration. Statistical data are collected over 1000 generations. First, the ratio of the unused individuals to population

size is shown in Figure 1. Obviously, on the average, 74% of the individuals in every generation remain unused, they are simply discarded and replaced by the newly produced offspring individuals. This ratio of unused individuals is independent of the encoding method used. That is, almost the same ratio is obtained with binary-valued and real-valued encodings.

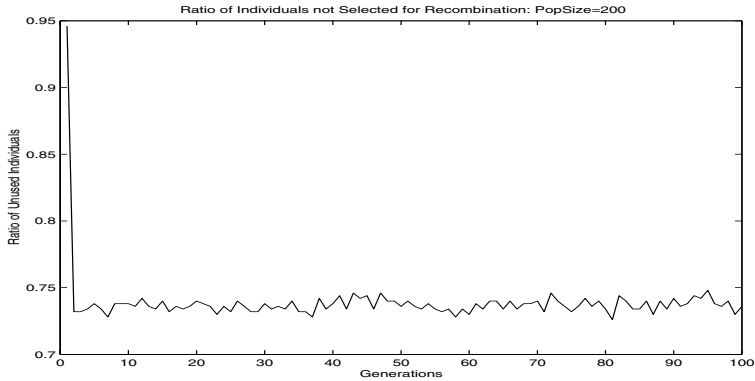


Fig. 1. The ratio of individuals which are not selected in any recombination operation for a population of 200 individuals.

The average ratio of individuals not selected for recombination changes with the population size. For example, this average is 52% for 100 individuals and 85% for 1000 individuals. In addition to this, these average ratios are approximately the same for the roulette-wheel selection method also.

A more clear insight can be obtained from the ratio of unused individuals having a fitness value greater than the population's average fitness. As illustrated in Figure 2, on the average, 32% of the individuals having a fitness value above the population average are not used at all in any recombination operation. The main motivation behind the chromosome reuse strategy is to put these close to average quality individuals into a chromosome library and make use of them for a number of future generations. This way, possible alternative paths to optimal solutions over these potentially promising solutions may be traced. In these experimental evaluations, it is also seen that 24% of the individuals having a fitness value above $0.75 * \text{Average\_Fitness}$ are not selected for recombination in all generations. Instead of totally wasting these potentially promising solutions, we can reuse them for a while to speedup the convergence process and to reduce the computational cost of constructing new individuals because chromosomes and fitness values of the individuals in the chromosome library are already determined.

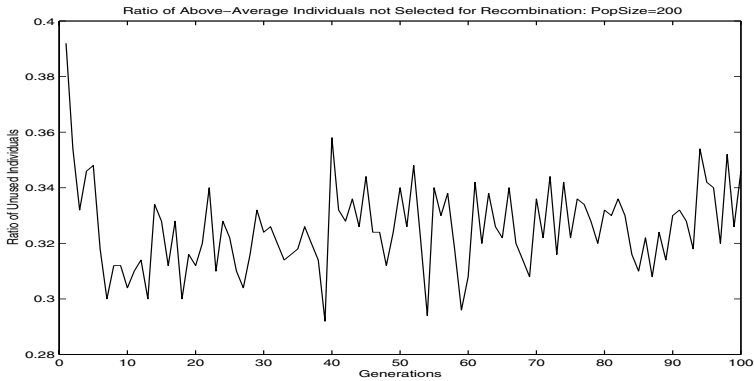


Fig. 2. The ratio of individuals having above average fitness and not selected in any recombination operation for a population of 200 individuals.

3 GAs with Chromosome Reuse Strategy

GAs with chromosome reuse strategy differs from the conventional GAs in the formation and maintenance of a chromosome library and the union of its individuals with the current population during the recombination procedure. The algorithmic description of the proposed approach is given in Figure 3.

In the proposed approach, the total memory space used to store individuals does not increase compared to the memory space needed by conventional GAs, because GAs with chromosome reuse strategy achieves better performance with smaller size populations. In experimental studies, the total number of individuals in the population and in the chromosome library is set equal to the number of individuals in the population of conventional GAs implementation, with which the proposed approach achieved better performance.

4 Two Case Studies

To study the performance of the described chromosome reuse strategy, it is compared with the conventional GAs for the solution of some benchmark problems from numerical and combinatorial optimization fields. Those benchmark numerical optimization problems handled in evaluations are listed in Table 1. They are taken from [20] and [21], which are claimed to provide reasonable test cases for the necessary combination of path-oriented and volume-oriented characteristics of a search strategy. For the combinatorial optimization problems, the 100-city symmetric traveling salesman problem, kroA100, taken from the website <http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/tsp/> is taken as a representative problem instance.

In all experiments, real-valued chromosomes are used for problem representations. The selection method used is the tournament selection with elitism.

1. $Max\_Library\_Size = \alpha * Population\_Size, 0 < \alpha < 1.0$;
2. $Fitness\_Threshold = \beta, 0 < \beta < 1.0$;
3. $Life\_Time = K$, where K is a predefined integer constant;
4. Generate chromosome library with randomly generated individuals;
5. Set the life time of individuals in the chromosome library to $Life\_Time$;
6. Evaluate chromosome library;
7. Generate initial population;
8. Evaluate initial population;
9. While (NOT DONE)
10. Combine the individuals in the current population and the chromosome library;
11. Reproduction;
12. Crossover;
13. Mutation;
14. Evaluate new population;
15. Decrease the life time of individuals in the chromosome library by 1;
16. Update chromosome library with individuals having $Fitness\_Values > \beta * Average\_Fitness$ and not used in any recombination operation.
17. end

Fig. 3. Genetic algorithms with chromosome reuse strategy.

The elite size is 10% of the population size. The uniform crossover operator is employed with a crossover rate equal to 0.7 and the mutation rate is 0.01. Experiments are carried out using a population size of 200 individuals for conventional GAs, also the total number of individuals in the population and the chromosome library for the proposed approach is 200, i.e. 100 individuals in each. This way, total number of individuals in conventional GAs and GAs with chromosome reuse strategy are kept the same. Individuals in the chromosome library have a predefined life duration, taken as 5 iterations in the experiments, and the removal of an individual from the chromosome library occurs either at the end its life time or the chromosome library is full and an individual with a better fitness replaces it. Each experiment is performed 10 times. All the tests were run over 1000 generations.

In the following worked examples, the results obtained with conventional GA and GAs with the chromosome reuse strategy are compared for relative performance evaluations.

4.1 Performance of Chromosome Reuse Strategy in Numerical Optimization

Conventional GAs are compared with GAs with chromosome reuse strategy for the minimization of functions listed in Table 1. Each function has 20 variables. The best solution found using the conventional GAs and GAS with chromosome reuse strategy are given in Table 2. Chromosome reuse strategy provided very

Table 1. Bechmark functions considered for numerical optimization.

Function Name	Expression
Michalewicz	$f(x) = -\sum_{i=1}^{n-1} \sin(x_i) \sin\left(\frac{\sin(ix_i^2)}{\pi}\right)^{(2m)}$ $-\sum_{i=1}^{n-1} \sin(x_{i+1}) \sin\left(\frac{2x_{i+1}^2}{\pi}\right)^{(2m)}$ $0 \leq x_i \leq \pi$
Griewangk	$f(x) = 1 + \sum_{i=1}^n \frac{x_i^2}{4000} - \prod_{i=1}^n \cos\left(\frac{x_i}{\sqrt{x_i}}\right)$ $-100 \leq x_i \leq 100$
Rastrigin	$f(x) = 10n + \sum_{i=1}^n (x_i^2 - 10\cos(2\pi x_i))$ $-5.12 \leq x_i \leq 5.12$
Schwefel	$f(x) = \sum_{i=1}^n (-x_i \sin(\sqrt{ x_i }))$ $-512 \leq x_i \leq 512$
Ackley's	$f(x) = -ae^{-b\sqrt{\frac{1}{n} \sum_{i=1}^n x_i^2}} - e^{\frac{1}{n} \sum_{i=1}^n \cos(cx_i)} + a + e$ $a = 20, b = 0.2, c = 2\pi$ $-32.768 \leq x_i \leq 32.768$
De Jong (Step)	$f(x) = 6n + \sum_{i=1}^n \lfloor x_i \rfloor$ $-5.12 \leq x_i \leq 5.12$

close to optimal results in all trials. These results demonstrate the success of the implemented GAs strategy for the numerical optimization problems.

Table 2. Performance evaluation of conventional GAs and GAs with chromosome reuse for numerical optimization.

Function	Global Opt., n=Num. Vars.	Best Found: Conv. GA		Best Found: Proposed	
		Global Min.	ITER	Global Min.	ITER
Michalewicz	-9.66 , $n = m = 10$	-8.55	100	-9.36	100
Griewangk	0, $n = 20$	0.0001	85	$1.0e^{-8}$	35
Rastrigin	0, $n = 20$	0.1	100	0.001	100
Schwefel	$-n * 418.9829$, $n = 20$	-8159	100	-8374	100
Ackley's	0, $n = 20$	0.03	100	0.001	100
De Jong (Step)	0, $n=20$	3	100	0	77

4.2 Performance of Chromosome Reuse Strategy in Combinatorial Optimization

To test the performance of the chromosome reuse strategy over a difficult problem of combinatorial type, the 100-city TSP kroA100 is selected. The best found solution for this problem is 21282 obtained using a branch-and-bound algorithm. In the ten experiments performed, the best solution found for this problem using the conventional GAs is 21340 which is obtained in 1000 generations with

population size equal to 200. The best solution obtained with the chromosome reuse strategy is 21282 which is obtained after 620 generations. Figure 4 shows the relative performance of chromosome reuse approach compared to the conventional GAs implementation, the straight line plot shows the results for the chromosome reuse strategy.

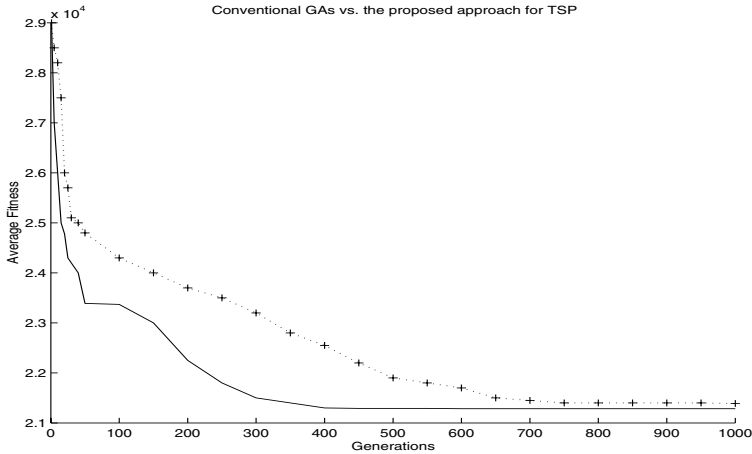


Fig. 4. Performance comparison of conventional genetic algorithms and chromosome reuse strategy in combinatorial

5 Conclusions and Future Work

In this paper a novel external memory-based genetic algorithms strategy based on the reuse of some potentially promising solutions from previous generations for the production of current offspring individuals is introduced as an alternative to the conventional implementation of GAs. The implemented strategy is used to solve difficult problems from numerical and combinatorial optimization areas and its performance is compared with the conventional GAs for representative problem instances. Each problem is solved exactly the same number of times with the employed strategies and the best and the average fitness results are analyzed for performance comparisons. All GA parameters are kept the same in the comparison of the two approaches.

From the results of case studies, for the same population size, it is concluded that the chromosome reuse strategy outperforms the conventional implementation in all trials. The performance of the chromosome reuse approach is the same for both numerical and combinatorial optimization problems. In fact, problems from these classes are purposely chosen to examine this side of the proposed strategy.

This work requires further investigation from following point of views: performance comparisons with other memory-based methods, performance evaluations for other problem classes, such as neural network design, speech processing, and face recognition; problem representations involving variable size chromosomes, particularly genetic programming; and mathematical analysis of chromosome reuse strategy.

References

1. Holland, J.H.: *Adaptation in Natural and Artificial Systems: An introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*, MIT Press, (1992).
2. Goldberg, D.E.: *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley Publishing Company, (1989).
3. Eshelman, L., Schaffer, J.: *Foundations of Genetic Algorithms 2*. In: L. Whitley (editor): pp. 187–202, Morgan Kaufmann Publishers, San Mateo, CA, (1993).
4. Back, T.: *Evolutionary Algorithms in Theory and Practice*, Oxford University Press, (1996).
5. Gen, M., Runwei, C.: *Genetic Algorithms in Engineering Design*, John Wiley & Sons, Inc., (1997).
6. Miettinen, K., Neittaanmaki, P., Makela, M.M., Periaux, J.: *Evolutionary Algorithms in Engineering and Computer Science*, John Wiley & Sons Ltd., (1999).
7. Cantu-Paz, E., Mejia-Olvera, M.: *Designing efficient master-slave parallel genetic algorithms*, IlliGAL Report No. 97004, Illinois Genetic Algorithm Laboratory, Urbana, IL, (1997).
8. Whitley, D., Starkweather, T.: *GenitorII: A distributed genetic algorithm*, *Journal of Experimental and Theoretical Artificial Intelligence*, (1990).
9. Eggermont, J., Lenaerts, T.: *Non-stationary function optimization using evolutionary algorithms with a case-based memory*, [url:http://citeseer.nj.nec.com/484021.html](http://citeseer.nj.nec.com/484021.html).
10. Goldberg, D. E., Smith, R. E.: *Non-stationary function optimization using genetic algorithms and with dominance and diploidy*, *Genetic Algorithms and their Applications: Proceedings of the Second International Conference on Genetic Algorithms*, p. 217–223, (1987).
11. Goldberg, D. E., Deb, K., Korb, B.: *Messy Genetic Algorithms: Motivation, analysis, and the first results*, *Complex Systems*, Vol. 3, No. 5, p. 493–530, (1989).
12. Lewis, J., Hart, E., Ritchie, G.: *A comparison of dominance mechanisms and simple mutation on non-stationary problems*, in Eiben, A. E., Back, T., Schoenauer, M., Schwefel, H. (Editors): *Parallel Problem Solving from Nature- PPSN V*, p. 139–148, Berlin, (1998).
13. Ryan, C., Collins, J. J.: *Polygenic inheritance- a haploid scheme that can outperform diploidy*, in Eiben, A. E., Back, T., Schoenauer, M., Schwefel, H. (Editors): *Parallel Problem Solving from Nature- PPSN V*, p. 178–187, Berlin, (1998).
14. Ryan, C.: *The degree of oneness*, *Firts Online Workshop on Soft Computing*, Aug. 19–30, (1996).
15. Ramsey, C.L., Grefenstette, J. J.: *Case-based initialization of GAs*, in Forest, S., (Editor): *Proceedings of the Fifth International Conference on Genetic Algorithms*, p. 84–91, San Mateo, CA, (1993).

16. Louis, S., Li, G.: Augmenting genetic algorithms with memory to solve travelling salesman problem, (1997).
17. Louis, S. J., Johnson, J.: Solving similar problems using genetic algorithms and case-based memory, in Back, T., (Editor): Proceedings of the Seventh International Conference on Genetic Algorithms, p. 84–91, San Fransisco, CA, (1997).
18. Luger, G.F.: Artificial Intelligence, 4th edition, Addison-Wesley, (2002).
19. S. Russel and P. Norvig, Artificial Intelligence: A Modern Approach, Prentice-Hall, (1995).
20. <http://www.f.utb.cz/people/zelinka/soma/func.html>.
21. Kim, H.S., Cho, S.B: An Efficient genetic algorithm with less fitness valuations by clustering, Proc. of the 2001 IEEE Congress on Evolutionary Computation, p.887–894, Seoul, Korea, May 27-30, (2001).

Real-Parameter Genetic Algorithms for Finding Multiple Optimal Solutions in Multi-modal Optimization

Pedro J. Ballester and Jonathan N. Carter

Imperial College London, Department of Earth Science and Engineering, London
SW7 2AZ, UK

Abstract. The aim of this paper is to identify Genetic Algorithms (GAs) which perform well over a range of continuous and smooth multi-modal real-variable functions. In our study, we focus on testing GAs combining three classes of genetic operators: selection, crossover and replacement. The approach followed is time-constrained and thus our stopping criterion is a fixed number of generations. Results show that GAs with random selection of parents and crowding replacement are robust optimizers. By contrast, GAs with tournament selection of parents and random replacement perform poorly in comparison.

1 Introduction

Genetic Algorithms (GAs) have proven to be a useful approach to address a wide variety of optimization problems. However, different problems usually require different combinations of GA parameters (population size, number of generations, choice of genetic operators, etc.). Unfortunately, GA parameters interact in complex ways, making the task of finding a suitable parameter set not always straightforward. In addition, a GA which excels with a given class of problems might yield poor results when applied to another class (Hart and Belew [11]).

This situation has given rise to the need for studies providing guidance in setting the GA parameters for different class of problems. As taking into account all the parameters is impractical, one usually focus on two or three parameters and try to understand their interactions. For instance, Deb and Agrawal [4], using a fixed set of binary genetic operators, studied the influence of population size, crossover probability and mutation probability over several classes of functions, which are known to be difficult to optimise.

The aim of this paper is to identify GAs which performs well over a specific class of functions: continuous, smooth, multi-modal real-variable functions. We say that a GA performs well if it has a good chance of finding the global minimum (we restrict without loss of generality to the minimization case). We are also interested in GAs that are able to find alternative good minima while reaching the global minimum too. There are two main reasons to set this secondary goal. First, real-world functions do not come without errors, which distort the fitness landscape. Therefore, a secondary minimum might actually be the global

one. Second, the optimal solution represented by the global minimum might be impossible to implement from the engineering point of view. In this case, a secondary good solution can be used as an alternative.

In our study, we focus on testing the performance of GAs combining three classes of genetic operators: selection, crossover and replacement. The approach followed is time-constrained and thus our stopping criterion is a fixed number of generations.

We arrange the rest of the paper as follows. In section 2, we describe the problem of multi-modal optimisation and present a set of test functions. Section 3 describes the structure of the GAs tested in this paper. In section 4, we explain the experimental setup introducing a performance measure. Results are discussed in section 5. Lastly, we present our conclusions in section 6.

2 GAs in Multi-modal Optimisation

Most of the traditional approaches to optimisation such as gradient-based methods are local optimisers. These sort of methods find the minimum of the basin of attraction in which the initial point is situated. Multi-modal optimisation is characterized by having a high number of attractors. Therefore, it is unlikely that the initial point lies within the basin containing the global minimum.

GAs are known to be a particularly good approach for multi-modal optimisation. Unlike local optimisers, GAs are stochastic approaches and thus are not strongly influenced by the initial population. The choice of GA parameters determines the balance of exploration of search space and the exploitation of information contained in the population. It is the adequacy of this balance that leads to the successful application of a GA.

Traditionally, the success of a GA run is verified by checking if there are individuals within a tight ϵ -neighbourhood of the minimum. By contrast, we are interested in finding the basin of attraction that contains the minimum. The motivation behind this definition of success is that of reducing the number of function evaluations needed. In practice, it is normally faster to find the minimum basin and then apply a fast local optimiser. This is a useful approach when the fitness evaluation carries a high computational cost.

Our main purpose is to find a robust optimiser for continuous, smooth, multi-modal real-variable functions. A particular GA is said to be robust if performs well on a given class of functions. As we cannot possibly consider every function in a class, we restrict our tests to a set of functions which represent the class. These test functions are defined in Figs. 1 to 6. The right-side plots corresponding to their two-dimensional views have been inverted in all cases to allow a better visualisation.

Each of these test functions has many local minima and one single global minimum. Functions 1, 2 and 3 are taken from previous studies [1] [8], while functions 4, 5 and 6 were constructed to be non-symmetric and composed by subfunctions which do not share minima with the whole function.

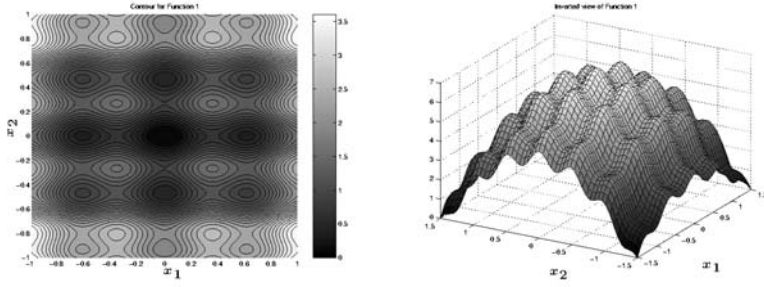


Fig. 1. $F_1(x_1, x_2) = x_1^2 + 2x_2^2 - 0.3 \cos(3\pi x_1) - 0.4 \cos(4\pi x_2) + 0.7$

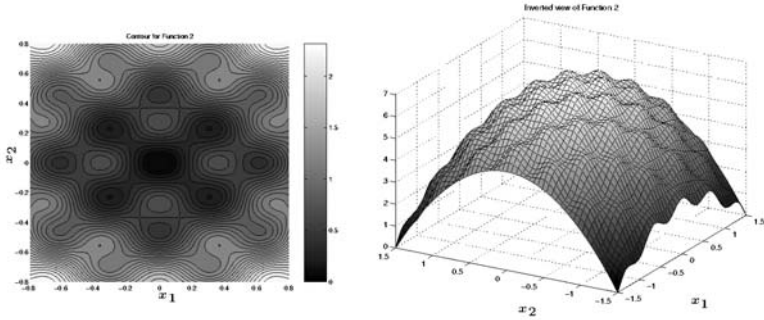


Fig. 2. $F_2(x_1, x_2) = x_1^2 + 2x_2^2 - 0.3(\cos(3\pi x_1) \cos(4\pi x_2)) + 0.3$

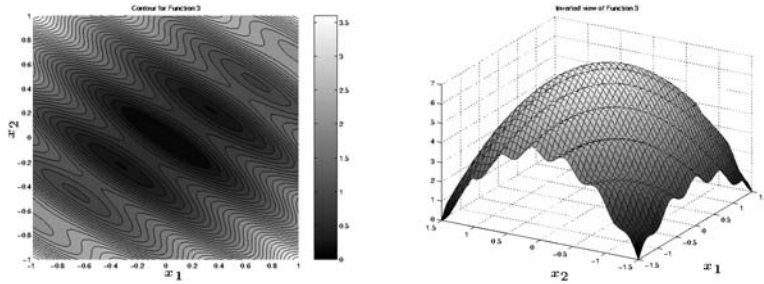


Fig. 3. $F_3(x_1, x_2) = x_1^2 + 2x_2^2 - 0.3 \cos(3\pi x_1 + 4\pi x_2) + 0.3$

3 Real-Parameter GAs

In this section, we describe the genetic operators used in our study. We aim to investigate the effect of three GA parameters (choice of selection scheme, replacement scheme and crossover operator) on the performance in multi-modal optimisation.

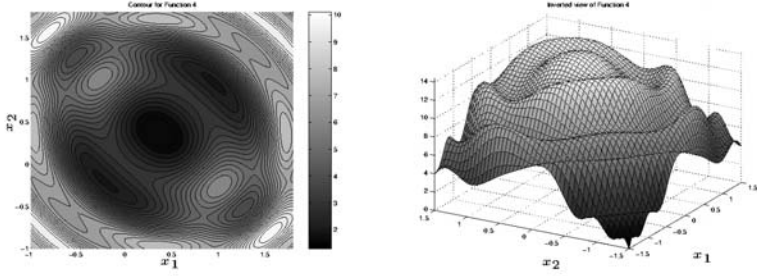


Fig. 4. $F_4(x_1, x_2) = \sum_{j=1}^2 (x_1 - d_j)^2 + (x_2 - d_j)^2 - \cos(\pi((x_1 - d_j)^2 + (x_2 - d_j)^2)) + 1$ with $d_1 = 0$ and $d_2 = 0.75$.

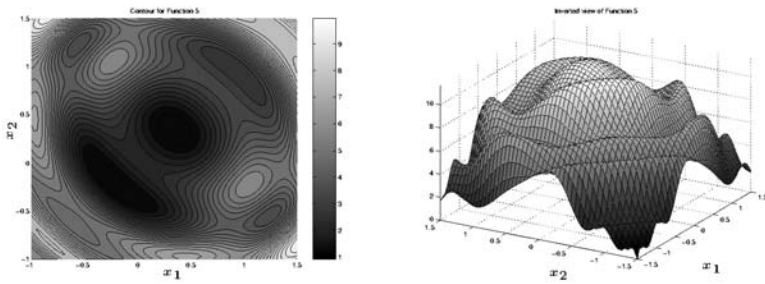


Fig. 5. $F_5(x_1, x_2) = \sum_{j=1}^2 (x_1 - \frac{d_j}{2})^2 + (x_2 - \frac{d_j}{2})^2 - \cos(\pi((x_1 - d_j)^2 + (x_2 - d_j)^2)) + 1$ with $d_1 = 0$ and $d_2 = 0.75$.

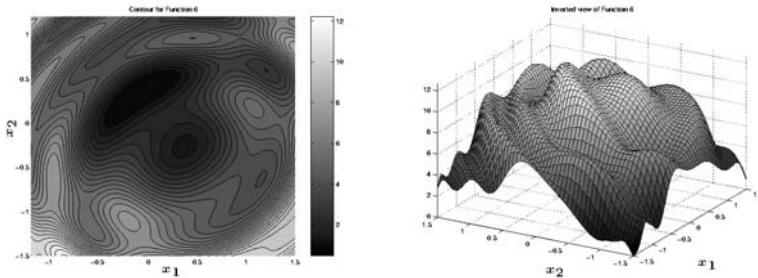


Fig. 6. $F_6(x_1, x_2) = \sum_{j=1}^2 (x_1 - \frac{d_j}{2})^2 + (x_2 - \frac{d_j}{2})^2 - \cos(\pi((x_1 - d_j)^2 + (x_2 + d_j)^2)) + 1$ with $d_1 = 0$ and $d_2 = 0.75$.

It has been discussed elsewhere [2] that real-parameter GAs are ideally suited to handle problems in a continuous search space. The encoding scheme is that the genome of each individual is composed of two real variables (x_1, x_2) , whose values represent a location on the search space. In order to avoid favouring a particular solution, the initial population is generated at random within the

range $-10 \leq x_1, x_2 \leq 10$. All the functions have their global minima within this range. An individual fitness is given by its function value.

In our GA architecture, a population of individuals enter the following loop for a fixed number of generations. First, three groups of parents are selected among the actual population. Second, each group of parents breeds two children, giving place to an offspring population of six new individuals. Lastly, offspring and parent populations are combined following a given replacement scheme to form a new population. We use a population size of 20, and terminate the process after 100 generations. This results in a total of 620 function evaluations.

We now present the different GA operators that we test in this paper:

3.1 Selection Schemes

We test two schemes: random (S_1) and tournament (S_2) selection [9]. In each scheme we generate three groups of parents. The parents within any of the groups are different from each other, although an individual may occur in more than one group. The number of parents within a group depends on the crossover operator used.

In random selection, the three groups of individuals are selected at random from the parent population. Whereas in tournament selection, two members of the parent population are picked at random and the fittest among them selected as a parent. The latter procedure is repeated until the required number of parents have been selected.

3.2 Crossover Schemes

Each of the three groups of parents produce two children, through the crossover operator, which are bred independently. This results in six new individuals per generation, which is an arbitrary choice and no claim about its optimality is made. Crossovers for real-parameter GAs have the interesting feature [2] of having tunable parameters that can be used to modify their exploration power. Thus, we use three different arbitrary parameter values for each crossover. The following four crossover operators are tested:

BLX. The Blend crossover (BLX) operator [7] randomly selects a value for each offspring gene y_i , using a uniform distribution within the range

$$[x_i^{(1)} - \alpha(x_i^{(2)} - x_i^{(1)}), x_i^{(2)} + \alpha(x_i^{(2)} - x_i^{(1)})]$$

where $x_i^{(1)}$ and $x_i^{(2)}$ are the parental genes, and α is the tunable parameter, we use $\alpha \in \{0.1, 0.5, 10\}$, the higher the value of α the more explorative the search.

SBX. We use the Simulated Binary crossover (SBX) operator [3] [5] with independently bred children. Therefore, for each children, we choose at random between the equally likely expressions $y_i = 0.5((1 + \beta_i)x_i^{(1)} + (1 - \beta_i)x_i^{(2)})$ and

$y_i = 0.5((1 - \beta_i)x_i^{(1)} + (1 + \beta_i)x_i^{(2)})$, where y_i is the offspring gene, $x_i^{(1)}$ and $x_i^{(2)}$ are the parental genes, and β_i a parameter. Then, we generate, for each coordinate, a second random number $u_i \in [0, 1]$ with β_i given by

$$\beta_i = \begin{cases} (2u_i)^{\frac{1}{\eta+1}} & 0 < u_i \leq 0.5 \\ (1/2(1 - u_i))^{\frac{1}{\eta+1}} & 0.5 < u_i \leq 1 \end{cases}$$

where η is the tunable parameter. A restricted search is achieved with a large value of η (we use $\eta \in \{0.1, 10, 150\}$). A plot of the distribution of children is given by Fig. 7.

vSBX. In SBX, children have zero probability of appearing in some regions of the parameter space, as shown in Fig. 7. A version of the latter crossover, called vSBX, is proposed that does not exclude any regions of the parameter space, while preserving the good SBX properties. This may allow a better exploration of the search space. In vSBX, you generate a random number $u \in [0, 1]$ to make the two following choices (only one value of u is needed per child). First, you choose between

$$y_i^{(1)} = \begin{cases} 0.5((1 + \beta_{1i})x_i^{(1)} + (1 - \beta_{1i})x_i^{(2)}) & 0 < u \leq 0.5 \\ 0.5((1 - \beta_{1i})x_i^{(1)} + (1 + \beta_{1i})x_i^{(2)}) & 0.5 < u \leq 1 \end{cases}$$

with $\beta_{1i} = (1/2u_i)^{\frac{1}{\eta+1}}$. Second, you choose between

$$y_i^{(2)} = \begin{cases} 0.5((3 - \beta_{2i})x_i^{(1)} - (1 - \beta_{2i})x_i^{(2)}) & 0 < u \leq 0.5 \\ 0.5(-(1 - \beta_{2i})x_i^{(1)} + (3 - \beta_{2i})x_i^{(2)}) & 0.5 < u \leq 1 \end{cases}$$

with $\beta_{2i} = (1/2(1 - u_i))^{\frac{1}{\eta+1}}$. u_i is a second random number that is used to choose between $y_i^{(1)}$ and $y_i^{(2)}$. The latter provides the offspring gene value y_i . A restricted search is achieved with a large value of η (we use $\eta \in \{0.1, 10, 150\}$).

UNDX. The Unimodal Normally Distributed crossover (UNDX) operator [13] changes every gene simultaneously by making use of a multivariate normal distribution. UNDX is unusual in that it needs three non-identical parents. The first two parents are used to define a dominant axis through the search space of real numbers. The crossover is carried out on a co-ordinate system that is rotated to coincide with this dominant axis. The distance between the two parents is used to define the standard deviation of a normal distribution along this dominant axis. The first parameter, a , is introduced as a non-negative real number multiplying the latter standard deviation. The distance of the third parent from this dominant axis is used to define the standard deviation used in that direction. The second parameter, b , is a non-negative real number multiplying the latter standard deviation. The reader is referred to the original paper (Ono and Kobayashi [13]) for details of the implementation.

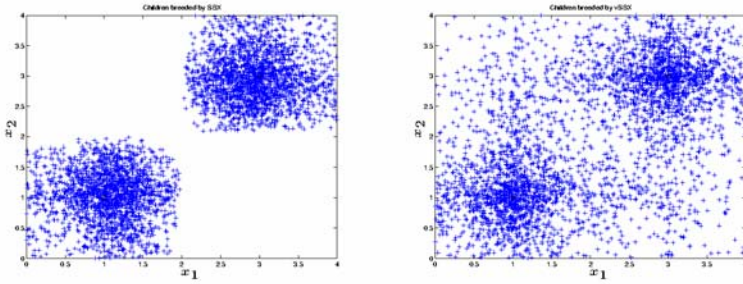


Fig. 7. Children bred from parents $\mathbf{x}^{(1)} = (1, 1)$ and $\mathbf{x}^{(2)} = (3, 3)$ for a) SBX ($\eta = 1$) and b) vSBX ($\eta = 1$)

The authors recommend $a = 1$ and $b = 0.5$. We keep the latter ratio between these two parameters, while changing the value of a . We test UNDX with parameter duples (a, b) equal to $(1, 0.5)$, $(0.5, 0.25)$ and $(0.25, 0.125)$, with the first $(1, 0.5)$ carrying the higher exploration and $(0.25, 0.125)$ the lower one.

3.3 Replacement Schemes

Five replacement schemes are tested: random (R_1), tournament (R_2), modified probabilistic crowding (R_3), classic probabilistic crowding (R_4) and deterministic crowding (R_5). In each of them, the offspring population is combined with the parent population to form a new one for the next generation. This new population is constructed to have the same size as the original population.

In Random replacement, six individuals from the current population are selected at random and completely replaced by the offspring population.

Tournament and crowding schemes operate in two steps. First, a part of the parent population is preselected at random. The size of this preselected group is taken as two in this work. Thereafter, the chosen scheme is followed to find out whether the offspring replaces one of the preselected individuals or not. The described operation is repeated for every offspring.

In the tournament scheme [10], the least fit of the preselected group competes with the offspring. Then, the fitter among them remains in the population. In the deterministic crowding [6], the preselected individual with the lower euclidean distance to the offspring competes with it. The fitter of them remains in the population.

The probabilistic crowding scheme [12] identifies the closest preselected individual ($\mathbf{x}^{cst}$) to enter a probabilistic tournament with the offspring ($\mathbf{x}^{ofp}$), with survival likelihoods given by $p(\mathbf{x}^{ofp}) = f(\mathbf{x}^{ofp}) / (f(\mathbf{x}^{ofp}) + f(\mathbf{x}^{cst}))$ and $p(\mathbf{x}^{cst}) = f(\mathbf{x}^{cst}) / (f(\mathbf{x}^{ofp}) + f(\mathbf{x}^{cst}))$.

Note that if the global minimum has a very high value with respect to the differences in fitness across the population, these likelihoods would be very similar in all cases. The modified probabilistic crowding is introduced to avoid this

situation. It operates with survival likelihoods $p(\mathbf{x}^{ofp}) = f(\mathbf{x}^{ofp}) / (f(\mathbf{x}^{ofp}) + f(\mathbf{x}^{cst}) - 2f_{worst})$ and $p(\mathbf{x}^{cst}) = f(\mathbf{x}^{cst}) / (f(\mathbf{x}^{ofp}) + f(\mathbf{x}^{cst}) - 2f_{worst})$, where f_{worst} is the fitness of the worst individual in the offspring and preselected group.

4 Experimental Setup

We define a ‘GA experiment’ by the i^{th} selection scheme S_i (see section 3.1), the j^{th} replacement scheme R_j (see section 3.3), the k^{th} crossover operator C_k (see section 3.2) and the l^{th} objective function F_l (see figures 1, 2, 3, 4, 5 and 6). As an experiment has some dependence on the initial conditions, we repeat each possible experiment with a different initial population (50 different sets of initial population are used).

We need to set boundaries to the main basins of attraction in order to evaluate the success of a run. This is done by adjusting an ellipse to the projection of the basin onto the search space. Thus, an individual lies within a basin defined by (c_1, c_2, θ, a, b) if

$$\frac{((x_1 - c_1) \cos \theta + (x_2 - c_2) \sin \theta)^2}{a^2} + \frac{((x_2 - c_2) \cos \theta - (x_1 - c_1) \sin \theta)^2}{b^2} \leq 1 \quad (1)$$

where a and b are the ellipse semiaxis, (c_1, c_2) is the ellipse center and θ is the angle between the ellipse and search space coordinate systems. Figure 8 shows the contour of function 1 with the global basin (represented by the central ellipse) and a set of basins containing the best alternative minima. In table 1 are the basins used for all of the test functions.

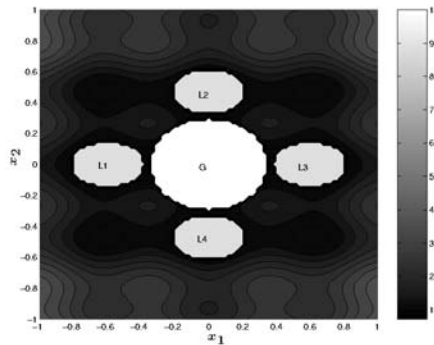


Fig. 8. Basins containing the best minima of function 1.

Finally, we define some performance indicators to study the success of each experiment. First, we define the primary success ratio (psr) for an experiment (S_i, R_j, C_k, F_l) as one, if at least one individual in the final population is within

Table 1. Defining values of each of the marked basins. The name of a basin has a G if includes the global minimum or L if includes a local minimum. θ is in radians.

	F_1					F_2					F_3			F_4			F_5		F_6	
	G	L1	L2	L3	L4	G	L1	L2	L3	L4	G	L1	L2	G	L1	L2	G	L1	G	L1
c_1	0	-.6	0	.6	0	0	-.3	-.3	.3	.3	0	-.35	.35	.37	-.15	.9	-.12	.35	-.03	.41
c_2	0	0	.47	0	-.47	0	-.2	.2	.2	-.2	0	-.25	.25	.37	-.15	.9	-.12	.35	.22	-.32
θ	0	0	0	0	0	0	0	0	0	0	.9	.9	.9	$\pi/4$	$\pi/4$	$\pi/4$	$\pi/4$	$\pi/4$	-.89	-.89
a^2	.35	.22	.22	.22	.22	.2	.15	.15	.15	.15	.2	.2	.2	.4	.3	.3	.3	.35	.3	.35
b^2	.3	.15	.15	.15	.15	.25	.1	.1	.1	.1	.9	.9	.9	.5	.8	.8	.8	.45	.85	.45

the basin containing the global minimum and zero otherwise. As we are also interested in multiple solutions, we introduce the secondary success ratio (ssr) for an experiment (S_i, R_j, C_k, F_l) as one, if $psr = 1$ and, in addition, at least one individual in the final population is within any of the marked alternative basins and zero otherwise.

5 Results

In this section, we present performance results of the several GA simulations carried out.

In order to investigate the performance of each crossover, let us take the case of the indicator primary success ratio (psr) as example and proceed as follows. First, for each of the 12 crossover operators C_k considered we sum over all possible S_i, R_j and F_l to dissipate the influence of selection, replacement and function over the crossover operator performance. Thereafter, we divide by the number of available schemes to normalise the resulting number (we do not include the case of random selection combined with random replacement). This procedure is contained in the following expression

$$psr(C_k) = \frac{1}{(N_S N_R - 1) N_F} \sum_i \sum_j \sum_l psr(S_i, R_j, C_k, F_l) \tag{2}$$

As each experiment was run 50 times, we can assess the uncertainty in equation 2 by providing an average and an standard deviation. Figure 9 shows the average of both psr and ssr (represented by ‘x’) and their corresponding estimation of uncertainty as twice the standard deviation (represented as a vertical interval). We also trace an horizontal line in each plot with the value of the highest average minus its standard deviation in order to allow an easier comparison. From Fig. 9, we see that those operators with the higher exploration power in their respective groups (ie. BLX-10, SBX-0.1, vSBX-0.1 and UNDX-(1,0.5)) outperforms the rest both in terms of psr and ssr. This result suggests that multi-modal optimisation requires extensive searches in order to find good minima.

Let us now compare the available combinations of selection and replacement schemes stated at the legend of Fig. 10. For each of these combinations (except

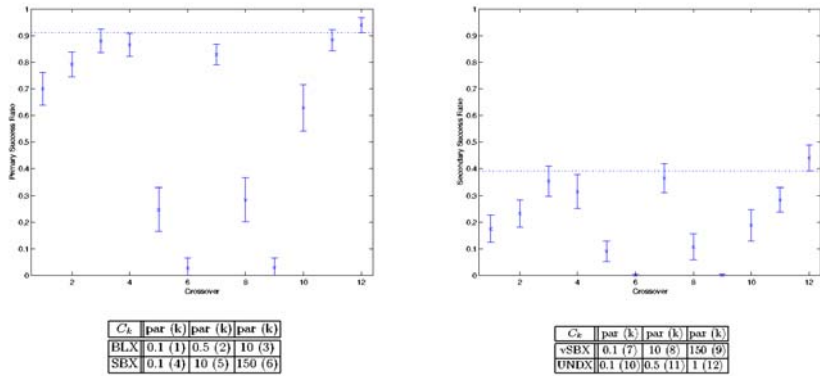


Fig. 9. Average and standard deviation of both performance indicators for each crossover.

again for combination one), we now dissipate the influence of crossover and function as follows

$$psr(S_i, R_j) = \frac{1}{N_{CNF}} \sum_k \sum_l psr(S_i, R_j, C_k, F_l) \quad (3)$$

Figure 10 shows the average and standard deviation over 50 different realisations of each available combination. It is observed that the GA with tournament selection and random replacement (combination 6) has the lowest probability of finding the global minimum. In addition, it does not perform well in finding more than one good minima, as we see from its ssr value. By contrast, random selection of parents combined with crowding replacement strategies (combinations 3, 4 and 5) seems to be more advisable approach for the class of problems we are dealing with. On the other hand, our proposed modified probabilistic crowding has a similar performance to the classical probabilistic crowding. Lastly, combinations 3, 4 and 5 perform better than 8, 9 and 10, which might be caused by too much exploitation in the latter group. Next, we want to study the performance of each GA implementation individually. Hence, for each combination of selection, crossover and replacement, we sum the indicator value over all possible functions as described in equation 4

$$psr(S_i, R_j, C_k) = \frac{1}{N_F} \sum_l psr(S_i, R_j, C_k, F_l) \quad (4)$$

Thereafter, we calculate the indicator average and standard deviation for each of these tripartite combinations. This operation leads to two remarkable findings. For all of the crossover operators, the combination tournament selection (S_2) and random replacement (R_1) gives poor results. On the other hand, the highest psr (0.997 ± 0.024) and ssr (0.847 ± 0.138) is achieved by S_1 , R_3 and C_{12} . Followed by S_1 , R_5 and C_{12} , with psr= 0.993 ± 0.033 and ssr= 0.810 ± 0.151 .

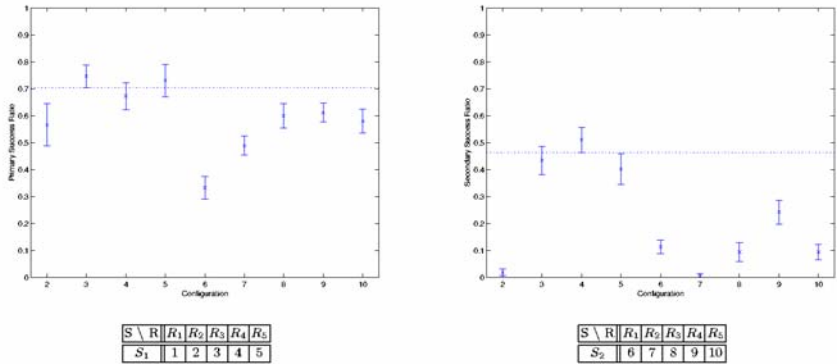


Fig. 10. Average and standard deviation of performance indicators for each combination of selection and replacement.

It is worth noting that all these results might depend on the number of generations. For instance, the GA with the best performance after 100 generations might be outperformed by another as the number of generations increase. This situation would be due to distinct convergence rates. Also, none of the implemented GAs is expected to indefinitely maintain subpopulations. Therefore, ssr would eventually reach the zero value, again at different rates for each GA.

In order to examine this question, we repeated our experiments with 300 generations (1820 function evaluations). Our main results are very similar to those with 100 generations. In general, all psr are marginally improved while ssr undergoes a large drop. Specifically, S_2 and R_1 has again the worst psr for most crossovers (all except C_{11} and C_{12}). The highest performance (psr=1 and ssr= 0.150 ± 0.118) corresponds still to S_1 , R_3 and C_{12} , but now is shared with S_1 , R_4 and C_{12} (psr=1 and ssr= 0.567 ± 0.165). These two followed by S_1 , R_3 and C_{12} (psr= 0.997 ± 0.024 and ssr= 0.233 ± 0.154), which previously obtained the second position.

6 Conclusions

Based on the set of experiments carried out, we arrive to the following conclusions for the multi-modal function optimisation problem:

- GAs using crossover operators with the highest search power obtain the best performance.
- GAs with tournament selection of parents and random replacement are not recommended because they have a poor performance.
- GAs with random selection of parents and crowding replacement are robust optimisers for this class of functions. In addition, they have a good chance of finding alternative good minima.

In this work, the number of new individuals per generation is fixed. We consider that future work analysing the influence of this variable is worthwhile. In addition, we plan to reproduce this benchmark for multi-modal functions in higher dimensions.

References

1. Bohachevsky, I.O., Johnson, M.E., and Stein, M.L. (1986): *Generalized Simulated Annealing for Function Optimization*. Technometrics 28(3), pp. 209–218.
2. Deb, K. (2001): *Multi-objective optimization using evolutionary algorithms*. 1st ed. (John Wiley & Sons, Ltd.).
3. Deb, K. and Agrawal, S. (1995): *Simulated binary crossover for continuous search space*. Complex Systems 9(2), pp. 115–148.
4. Deb, K. and Agrawal, S. (1999): *Understanding Interactions Among Genetic Algorithm Parameters*. In: Foundations of Genetic Algorithms 5, Ed. Wolfgang Banzhaf and Colin Reeves (Morgan Kaufmann), pp. 265–286.
5. Deb, K. and Kumar, A. (1995): *Real-coded genetic algorithms with simulated binary crossover: Studies on multi-modal and multi-objective problems*. Complex Systems 9(6), pp. 431–454.
6. DeJong, K.A. (1975): *An Analysis of the Behavior of a Class of Genetic Adaptive Systems*. Ph. D. Thesis (University of Michigan, Ann Arbor).
7. Eshelman, L.J. and Schaffer, J.D. (1993): *Real coded genetic algorithms and interval schemata*. In: Foundations of Genetic Algorithms 2, Ed. L. Darrell Whitley (Morgan Kaufmann), pp. 187–202.
8. Fogel, D.B. (1995): *Evolutionary computation: toward a new philosophy of machine intelligence* (Institute of Electrical and Electronics Engineers, Inc.).
9. Goldberg, D.E. and Deb, K. (1991): *A comparative analysis of selection schemes used in genetic algorithms*. In: Foundations of Genetic Algorithms, Ed. Gregory J.E. Rawlings (Morgan Kaufmann), pp. 69–93.
10. Harik, G.R. (1995): *Finding multi-modal solutions using restricted tournament selection*. In: Proceedings of the Sixth International Conference on Genetic Algorithms, Ed. L.J. Eshelman (Morgan Kaufmann), pp. 24–31.
11. Hart, W.E. and Belew, R.K. (1991): *Optimizing an Arbitrary Function is Hard for the Genetic Algorithm*. In: Proceedings of the Fourth International Conference on Genetic Algorithms, Ed. Richard K. Belew and Lashon B. Booker (Morgan Kaufmann), pp. 190–195.
12. Mengshoel, O.J. and Goldberg, D.E. (1999): *Probabilistic Crowding: Deterministic Crowding with Probabilistic Replacement*. In: Proceedings of the Genetic and Evolutionary Computation Conference, Ed. W. Banzhaf, J. Daida, A.E. Eiben, M.H. Garzon, V. Honavar, M. Jakiela and R.E. Smith (Morgan Kauffmann), pp. 409–416.
13. Ono, I. and Kobayashi, S. (1997): *A real-coded genetic algorithm for function optimization using unimodal normal distribution*. In: Proceedings of the Seventh International Conference on Genetic Algorithms, Ed. Thomas Bäck (Morgan Kauffmann), pp. 246–253.

An Adaptive Penalty Scheme for Steady-State Genetic Algorithms

Helio J.C. Barbosa¹ and Afonso C.C. Lemonge²

¹ LNCC/MCT

Rua Getulio Vargas 333
25651 070 Petropolis RJ, BRAZIL
hcbm@lncc.br

² Depto. de Estruturas, Faculdade de Engenharia
Universidade Federal de Juiz de Fora
36036 330 Juiz de Fora MG, BRAZIL
lemonge@numec.ufjf.br

Abstract. A parameter-less adaptive penalty scheme for steady-state genetic algorithms applied to constrained optimization problems is proposed. For each constraint, a penalty parameter is adaptively computed along the run according to information extracted from the current population such as the existence of feasible individuals and the level of violation of each constraint. Using real coding, rank-based selection, and operators available in the literature, very good results are obtained.

1 Introduction

Evolutionary algorithms (EAs) are weak search algorithms which can be directly applied to unconstrained optimization problems where one seeks for an element x belonging to the search space S , which minimizes (or maximizes) the real function f . Such EAs usually employ a fitness function closely related to f .

The straightforward application of EAs to constrained optimization problems (COPs) is not possible due to the additional requirement that a set of constraints must be satisfied. Several difficulties may arise: (i) the objective function may be undefined for some or all infeasible elements, (ii) the check for feasibility can be more expensive than the computation of the objective function value, and (iii) an informative measure of the degree of infeasibility of a given candidate solution is not easily defined. It is easy to see that even if both the objective function $f(x)$ and a measure of constraint violation $v(x)$ are defined for all $x \in S$ it is not possible to know in general which of two given infeasible solutions is closer to the optimum and thus should be operated upon or kept in the population. For minimization problems, for instance, one can have $f(x_1) > f(x_2)$ and $v(x_1) = v(x_2)$ or $f(x_1) = f(x_2)$ and $v(x_1) > v(x_2)$ and still have x_1 closer to the optimum.

It is also important to note that –for convenience and easier reproducibility– most comparisons between EAs in the literature have been conducted in problems with constraints which can be written as $g_i(x) \leq 0$, where each $g_i(x)$ is a given *explicit* function of the independent(design) variable $x \in \mathbb{R}^n$. Although

the available test problems attempt to represent different types of difficulties one is expected to encounter when dealing with practical situations, very often the constraints cannot be put explicitly in the form $g_i(x) \leq 0$. For instance, in structural engineering design most constraints (such as stress and deformation) are only known as implicit functions of the design variables. In order to check if a constraint has been violated, a whole computational simulation (carried out by a specific code expending considerable computational resources) is required.

The techniques for handling constraints within EAs can be classified either as *direct* (feasible or interior), when only feasible elements in S are considered or as *indirect* (exterior), when both feasible and infeasible elements are used during the search process. Direct techniques comprise the use of: a) *closed* genetic operators (in the sense that when applied to feasible parents they produce feasible offspring) which can be designed provided enough domain knowledge is available [1], b) special decoders [2] (which always generate feasible individuals from any given genotype) although no applications considering implicit constraints have been published, c) repair techniques [3,4] which use domain knowledge in order to move an infeasible offspring into the feasible set (a challenge when implicit constraints are present), and d) “the death penalty”, when any infeasible element is simply discarded irrespective of its potential information content.

Summarizing, direct techniques are problem dependent (with the exception of the “death penalty”) and actually of extremely reduced practical applicability.

Indirect techniques comprise the use of: a) Lagrange multipliers [5], which may also lead to a min-max problem defined for the associated Lagrangean $\mathcal{L}(x, \lambda)$ where the primal variables x and the multipliers λ are approximated by two different populations in a coevolutionary GA [6], b) fitness as well as constraint violation values in a multi-objective optimization setting [7], c) special selection techniques [8], and d) “lethalization”: any infeasible offspring is just assigned a given, very low, fitness value [9]. For other methods proposed in the evolutionary computation literature see [1,10,11,12,13] and references therein.

Methods to tackle COPs which require the knowledge of constraints in explicit form have thus limited practical applicability. This fact, together with simplicity of implementation are perhaps the main reasons why penalty techniques, in spite of their shortcomings, are the most popular ones.

In a previous paper [14] a penalty scheme which does not require the knowledge of the explicit form of the constraints as a function of the decision/design variables and is free of parameters to be set by the user was developed. In contrast with previous approaches where a single penalty parameter is used for all constraints, an adaptive scheme automatically sizes the penalty parameter corresponding to *each* constraint along the evolutionary process. However, the method was conceived for a *generational* genetic algorithm (GA), where the fitness of the whole population is computed at each generation.

In this paper, the procedure proposed in [14] is extended to the case of a steady-state GA where, in each “generation”, usually only one or two (in general just a few) new individuals are introduced in the population. Substantial

modifications were necessary in order to finally obtain a robust procedure capable of reaching very good results in a standard test-problem suite.

In the next section the penalty method and some of its implementations within EAs are presented. In Section 3 the proposed adaptive scheme for steady-state GAs is discussed, Section 4 presents numerical experiments with several test-problems from the literature and the paper closes with some conclusions.

2 Penalty Methods

A standard COP in R^n can be thought of as the minimization of a given objective function $f(x)$, where $x \in R^n$ is the vector of design/decision variables, subject to inequality constraints $g_p(x) \geq 0$, $p = 1, 2, \dots, \bar{p}$ as well as equality constraints $h_q(x) = 0$, $q = 1, 2, \dots, \bar{q}$. Additionally, the variables may be subject to bounds $x_i^L \leq x_i \leq x_i^U$ but this type of constraint is trivially enforced in a GA and need not be considered here.

Penalty techniques can be classified as *multiplicative* or *additive*. In the multiplicative case [15], a positive penalty factor $p(v(x), T)$ is introduced in order to amplify the value of the fitness function of an infeasible individual in a minimization problem. One would have $p(v(x), T) = 1$ for a feasible candidate solution x and $p(v(x), T) > 1$ otherwise. Also, $p(v(x), T)$ increases with the “temperature” T and with constraint violation. An initial value for the temperature is required as well as the definition of a function such that T grows with the generation number. This type of penalty has received much less attention in the evolutionary computation (EC) community than the additive type. In the additive case, a penalty functional is added to the objective function in order to define the fitness value of an infeasible element. They can be further divided into: (a) *interior* techniques¹ and (b) *exterior* techniques, where a penalty functional is introduced

$$F(x) = f(x) + kP(x) \quad (1)$$

such that $P(x) = 0$ if x is feasible and $P(x) > 0$ otherwise (for minimization problems). In both cases, as $k \rightarrow \infty$, the sequence of minimizers of the unconstrained problem converges to the solution of the original constrained one.

Defining the amount of violation of the j -th constraint by the candidate solution $x \in R^n$ as

$$v_j(x) = \begin{cases} |h_j(x)|, & \text{for an equality constraint,} \\ \max\{0, -g_j(x)\} & \text{otherwise} \end{cases}$$

it is common to design penalty functions that grow with the vector of violations $v(x) \in R^m$ where $m = \bar{p} + \bar{q}$ is the number of constraints to be penalized. The most popular penalty function is given by

$$P(x) = \sum_{j=1}^m (v_j(x))^\beta \quad (2)$$

¹ When a barrier functional, which grows rapidly as x approaches the boundary of the feasible domain, is added to the objective function.

where $\beta = 2$. Although it is easy to obtain the unconstrained problem, the definition of a good penalty parameter k is usually a time-consuming trial-and-error process.

Powell & Skolnick[16] proposed a method enforcing the superiority of any feasible solution over any infeasible one defining the fitness as

$$F(x) = f(x) + r \sum_{j=1}^m v_j(x) + \theta(t, x)$$

where $\theta(t, x)$ is conveniently defined and r is a constant.

A variant, (see Deb[17]) uses the fitness function:

$$F(x) = \begin{cases} f(x), & \text{if } x \text{ is feasible,} \\ f_{max} + \sum_{j=1}^m v_j(x), & \text{otherwise} \end{cases}$$

where f_{max} is the objective function value of the worst feasible solution.

Besides the widely used case of a single constant penalty parameter k , several other proposals are available [18,10,19] and some of them, more closely related to the work presented here, will be briefly discussed in the following.

2.1 Related Methods in the Literature

Two-level Penalties. Le Riche et al.[20] present a GA where two fixed penalty parameters k_1 and k_2 are used independently in two different populations. The idea is to create two sets of candidate solutions where one of them is evaluated with the parameter k_1 and the other with the parameter k_2 . With $k_1 \gg k_2$ there are two different levels of penalization and there is a higher chance of maintaining feasible as well as infeasible individuals in the population and to get offspring near the boundary between the feasible and infeasible regions.

Multiple Coefficients. Homaifar et al.[21] proposed different penalty coefficients for different levels of violation of each constraint. The fitness function is written as

$$F(x) = f(x) + \sum_{j=1}^m k_{ij}(v_j(x))^2$$

where i denotes one of the l levels of violation defined for the j -th constraint. This is an attractive strategy because, at least in principle, it allows for a good control of the penalization process. The weakness of this method is the large number, $m(2l+1)$, of parameters that must be set by the user for each problem.

Dynamic Coefficients. Joines & Houck[22] proposed that the penalty parameters should vary dynamically along the search according to an exogenous schedule. The fitness function $F(x)$ was written as in (1) and (2) with the penalty parameter, given by $k = (C \times t)^\alpha$, increasing with the generation number t .

Adaptive Penalties. A procedure where the penalty parameters change according to information gathered during the evolution process was proposed by Bean & Hadj-Alouane[23]. The fitness function is again given by (1) and (2) but with the penalty parameter $k = \lambda(t)$ adapted at each generation by the rules:

$$\lambda(t+1) = \begin{cases} (\frac{1}{\beta_1})\lambda(t), & \text{if } b^i \in \mathcal{F} \text{ for all } t-g+1 \leq i \leq t \\ \beta_2\lambda(t), & \text{if } b^i \notin \mathcal{F} \text{ for all } t-g+1 \leq i \leq t \\ \lambda(t) & \text{otherwise} \end{cases}$$

where b^i is the best element at generation i , $\mathcal{F}$ is the feasible set, $\beta_1 \neq \beta_2$ and $\beta_1, \beta_2 > 1$. In this method the penalty parameter of the next generation $\lambda(t+1)$ decreases when all best elements in the last g generations were feasible, increases if all best elements were infeasible and otherwise remains without change.

The method proposed by Coit et al.[24], uses the fitness function:

$$F(x) = f(x) + (F_{feas}(t) - F_{all}(t)) \sum_{j=1}^m (v_j(x)/v_j(t))^\alpha$$

where $F_{all}(t)$ corresponds to the best solution, until the generation t (without penalty), F_{feas} corresponds to the best feasible solution and α is a constant.

Schoenauer & Xanthakis[25] presented a strategy that handles constrained problems in stages: (i) initially, a randomly generated population is evolved considering only the first constraint until a certain percentage of the population is feasible with respect to that constraint; (ii) the final population of the first stage of the process is used in order to optimize with respect to the second constraint. During this stage, the elements that had violated the previous constraint are removed from the population, (iii) the process is repeated until all the constraints are processed. This strategy becomes less attractive as the number of constraints grows and is potentially dependent on the order in which the constraints are processed.

Recently, Hamida & Schoenauer[26] proposed an adaptive scheme using a niching technique with adaptive radius to handle multimodal functions.

Other Techniques. Runarsson & Yao[8] presented a novel approach where a good balance between the objective and the penalty function values is sought by means of a stochastic ranking scheme. However, there is a parameter, P_f , (the probability of using only the objective function for ranking infeasible individuals) that must be set by the user.

Later, Wright & Farmani[27] proposed a method that requires no parameters and aggregates all constraint violations in a single infeasibility measure.

For constraint satisfaction problems, adaptive EAs have been developed successfully by Eiben and co-workers (see [28]).

3 The Proposed Method

In a previous paper[14] a penalty scheme was proposed which adaptively sizes the penalty coefficient of *each* constraint using information from the population

such as the average of the objective function and the level of violation of each constraint. The fitness function was written as:

$$F(x) = \begin{cases} f(x), & \text{if } x \text{ is feasible,} \\ \mathbf{h}(x) + \sum_{j=1}^m k_j v_j(x) & \text{otherwise} \end{cases} \quad (3)$$

where

$$\mathbf{h}(x) = \begin{cases} f(x), & \text{if } f(x) > \langle f(x) \rangle, \\ \langle f(x) \rangle & \text{otherwise} \end{cases} \quad (4)$$

and $\langle f(x) \rangle$ is the average of the objective function values in the current population. The penalty parameter was defined at each *generation* by:

$$k_j = |\langle f(x) \rangle| \frac{\langle v_j(x) \rangle}{\sum_{l=1}^m [\langle v_l(x) \rangle]^2} \quad (5)$$

and $\langle v_l(x) \rangle$ is the violation of the l -th constraint averaged over the current population. The idea is that the penalty coefficients should be distributed in such a way that those constraints which are more difficult to be satisfied should have a relatively higher penalty coefficient. It is also clear that the notion of the superiority of any feasible over any infeasible solution[16] is not enforced here.

It must be observed that in all procedures where a penalty coefficient varies along the run one must ensure that the fitness value of all elements is computed with the same penalty coefficient(s) so that standard selection schemes remain valid. For a generational GA, one can simply update the coefficient(s) every, say g , generations. As the concept of generation does not hold for a steady-state GA extra care must be taken in order to ensure that selection (for reproduction as well as for replacement) works properly.

A straightforward extension of that penalty procedure[14] to the steady-state case would be to periodically update the penalty coefficients and the fitness function values for the population. However, in spite of using a real-coding, the results obtained were inferior to those of the binary-coded generational case[14].

Further modifications are then proposed here for the steady-state version of that penalty scheme. The fitness function is still computed according to (3). However, $\mathbf{h}$ and the penalty coefficients are redefined respectively as

$$\mathbf{h} = \begin{cases} f(x_{worst}) & \text{if there is no feasible element in the population,} \\ f(x_{bestfeasible}) & \text{otherwise} \end{cases} \quad (6)$$

$$k_j = \mathbf{h} \frac{\langle v_j(x) \rangle}{\sum_{l=1}^m [\langle v_l(x) \rangle]^2} \quad (7)$$

Also, every time a better feasible element is found (or the number of new elements inserted into the population reaches a certain level) $\mathbf{h}$ is redefined and all fitness values are recomputed using the updated penalty coefficients. The updating of each penalty coefficient is performed in such a way that no reduction in its value is allowed. For convenience one should keep, for each individual in the population, the objective function value and all constraint violations. The fitness function value is then computed using (6), (7), and (3).

It is clear from the definition of h in (6) that if no feasible element is present in the population one is actually minimizing a measure of the distance of the individuals to the feasible set since the actual value of the objective function is not taken into account. However, when a feasible element is found then it immediately enters the population since, after updating all fitness values using (6), (7), and (3), it becomes the element with the best fitness value.

A pseudo-code for the proposed adaptive penalty scheme for a steady-state GA can be written as shown in Figure 1. Numerical experiments are then presented in the following section.

```

Begin
Initialize population
Compute objective function and constraint violation values
if there is no feasible element then
    h = worst objective function value
else
    h = objective function value of best feasible individual
endif
Compute penalty coefficients
Compute fitness values
ninser = 0
repeat
    Select operator
    Select parent(s)
    Generate offspring
    Evaluate offspring
    Keep best offspring
    if offspring is the new best feasible element then
        update penalty coefficients and fitness values
        ninser = 0
    endif
    if offspring is better than the worst in the population then
        worst is removed
        offspring is inserted
        ninser = ninser + 1
    endif
    if (ninser/popsize >= r) then
        update penalty coefficients and fitness values
        ninser = 0
    endif
until maximum number of evaluations is reached
End

```

Fig. 1. Pseudo-code for the steady-state GA with adaptive penalty scheme. (**ninser** is a counter for the number of offspring inserted in the population, **popsize** is the population size and **r** is a fixed constant that was set to 3 in all cases)

4 Numerical Experiments

In order to investigate the performance of the proposed penalty procedure, the 11 well known G1-G11 test-functions presented by Koziel & Michalewicz[2] are considered. The G-Suite is made up of different kinds of functions and involves constraints given by linear inequalities, nonlinear equalities, and nonlinear inequalities. An extended discussion involving each one of these problems and other techniques from the evolutionary computation literature can be found in [29].

A simple real-coded steady-state GA with a linear ranking selection scheme was implemented. The operators used were: (i) random mutation (which modifies a randomly chosen variable of the selected parent to a random value uniformly distributed between the lower and upper bounds of the corresponding variable), (ii) non-uniform mutation (as proposed by Michalewicz[30]), (iii) Muhlenbein's mutation (as described in [31]), (iv) multi-parent discrete crossover (which generates an offspring by randomly taking each allele from one of the n_p selected parents), and (v) Deb's SBX crossover as described in [32].

No parameter tuning was attempted. The same probability of application (namely 0.2) was assigned to all operators above, n_p was set to 4, and η was set to 2 in SBX. This set of values was applied to *all* test-problems in order to demonstrate the robustness of the procedure. Each equality constraint was converted into one inequality constraint of the form $|h(x)| \leq 0.0001$. Enlarging the set of operators, changing the relative probabilities of application, population size, or parameters associated with operators in each case could of course lead to local performance gains.

The Tables 1, 2, 3, and 4 show the results obtained for the G1-G11 test-functions, in 20 independent runs, using a population containing 800 individuals and a maximum number of function evaluations n_{eval} set to 320000, 640000, 1120000, and 1440000, respectively. It is clear that good results were found for all test-functions and at all levels of n_{eval} .

The Table 5 displays a comparison of results found in the Experiment 3 (Table 3) –where $n_{eval} = 1120000$ – and the results found in the Experiment #2 of [14] where a generational binary-coded GA –with **popsize** = 70 and $n_{eval} = 1400000$ – was used in 20 independent runs.

The Table 6 compares the results from Experiment 3 with those presented by Hamida & Shoenauer[26] using a (100 + 300)–ES segregational selection scheme with an adaptive penalty and a niching strategy. They performed 31 independent runs comprising 5000 generations ($n_{eval} = 1500000$) each.

The Tables 5 and 6 show that better results are obtained with the proposed adaptive steady-state GA using less function evaluations. The interested reader can find additional results in [2,33,27,29], and verify that they are not superior to those presented here.

Finally, in Table 7 we compare the results obtained with the parameter-less scheme proposed here, using **popsize** = 700, with those of Runarsson & Yao[8], both with $n_{eval} = 350000$. It must be observed that the results in Table 7 are the best in [8] (and probably the best in the evolutionary computation literature) and correspond to the choice $P_f = 0.45$. However, one can see in [8] that slightly

Table 1. Exp. 1: $n_{eval} = 320000$.

$f(x)$	Experiment 1		
	worst	best	average
G1	-15.00	-15.00	-15.00
G2	0.7701039	0.7980134	0.7894922
G3	0.7468729	0.9970834	0.8733876
G4	-30665.54	-30665.54	-30665.54
G5	5667.431	5126.484	5829.603
G6	-6961.811	-6961.811	-6961.811
G7	27.05797	24.31103	24.86856
G8	0.0958250	0.0958250	0.0958250
G9	680.7184	680.6303	680.64824
G10	10864.27	7139.031	7679.41880
G11	0.749	0.749	0.74899

Table 2. Exp. 2: $n_{eval} = 640000$.

$f(x)$	Experiment 2		
	worst	best	average
G1	-13.00	-15.00	-14.90
G2	0.7624246	0.8036177	0.7904785
G3	0.9318285	1.000491	0.9890722
G4	-30665.54	-30665.54	-30665.54
G5	5632.585	5126.484	5257.531
G6	-6961.811	-6961.811	-6961.811
G7	25.77410	24.32803	24.70925
G8	0.0958250	0.0958250	0.0958250
G9	680.6932	680.6305	680.6385
G10	7786.534	7098.464	7413.0185
G11	0.749	0.749	0.74899

Table 3. Exp. 3: $n_{eval} = 1120000$.

$f(x)$	Experiment 3		
	worst	best	average
G1	-15.00	-15.00	-15.00
G2	0.7778333	0.8036125	0.7900538
G3	0.9593665	1.000498	0.9981693
G4	-30665.54	-30665.54	-30665.54
G5	5639.265	5126.484	5205.561
G6	-6961.811	-6961.811	-6961.811
G7	25.24219	24.31465	24.58272
G8	0.0958250	0.0958250	0.0958250
G9	680.6494	680.6301	680.6333
G10	8361.596	7049.360	7339.957
G11	0.749	0.749	0.74899

Table 4. Exp. 4: $n_{eval} = 1440000$.

$f(x)$	Experiment 4		
	worst	best	average
G1	-15.00	-15.00	-15.00
G2	0.7778334	0.8036024	0.7908203
G3	1.000340	1.000499	1.000460
G4	-30665.54	-30665.54	-30665.54
G5	5672.701	5126.484	5206.389
G6	-6961.811	-6961.811	-6961.811
G7	25.51170	24.30771	24.52875
G8	0.0958250	0.0958250	0.0958250
G9	680.7122	680.6301	680.6363
G10	7942.683	7072.100	7300.013
G11	0.749	0.749	0.74899

Table 5. Results from this study (SSGA) and the generational GA (GGA) of [14].

$f(x)$	optimum	best values		average values		worst values	
		SSGA	GGA	SSGA	GGA	SSGA	GGA
G1	-15.0	-15.00	-15.00	-15.00	-15.00	-15.00	-15.00
G2	0.803619	0.8036125	0.7918570	0.7900538	0.7514353	0.7778333	0.6499022
G3	1.0	1.000498	1.000307	0.9981693	0.9997680	0.9593665	0.9983935
G4	-30655.539	-30665.54	-30665.51	-30665.54	-30665.29	-30665.54	-30664.91
G5	5126.4981	5126.484	5126.571	5205.561	5389.347	5639.265	6040.595
G6	-6961.814	-6961.811	-6961.796	-6961.811	-6961.796	-6961.811	-6961.796
G7	24.306	24.31465	24.85224	24.58272	27.90973	25.24219	33.07581
G8	0.0958250	0.0958250	0.0958250	0.0958250	0.0942582	0.0958250	0.0795763
G9	680.630	680.6301	680.6678	680.6333	680.9640	680.6494	681.6396
G10	7049.33	7049.360	7080.107	7339.957	8018.938	8361.596	9977.767
G11	0.75	0.749	0.75	0.74899	0.75	0.749	0.75

changing that parameter to $P_f = 0.475$ produces changes in the second most relevant digit of the best values found for functions G6 and G10, and severely degrades the mean value for functions G1, G6 and G10. It is clear that our first results presented in this paper are very competitive.

Table 6. Comparison between this study (SSGA) and Hamida & Schoenauer[26]. Average values for this study were computed with feasible and infeasible final solutions. Those in [26] considered only feasible solutions. Worst values were not given in [26].

$f(x)$	optimum	best values		average values	
		SSGA	H&S	SSGA	H&S
G1	-15.0	-15.00	-15.00	-15.00	-14.84
G2	0.803619	0.8036125	0.785	0.7900538	0.59
G3	1.0	1.000498	1.0	0.9981693	0.99989
G4	-30655.539	-30665.54	-30665.5	-30665.54	-30665.5
G5	5126.4981	5126.484	5126.5	5205.561	5141.65
G6	-6961.814	-6961.811	-6961.81	-6961.811	-6961.81
G7	24.306	24.31465	24.3323	24.58272	24.6636
G8	0.0958250	0.0958250	0.095825	0.0958250	0.095825
G9	680.630	680.6301	680.630	680.6333	680.641
G10	7049.33	7049.360	7061.13	7339.957	7497.434
G11	0.75	0.749	0.75	0.74899	0.75

Table 7. Comparison of results between this study (SSGA) and Runarsson & Yao[8].

$f(x)$	optimum	best values		worst values	
		SSGA	R&Y	SSGA	R&Y
G1	-15.0	-15.00	-15.00	-15.00	-15.00
G2	0.803619	0.8035839	0.803515	0.7777818	0.726288
G3	1.0	0.9960645	1.0	0.6716288	1.00
G4	-30655.539	-30665.54	-30665.539	-30644.32	-30665.539
G5	5126.4981	5126.484	5126.497	5624.208	5142.472
G6	-6961.814	-6961.811	-6961.814	-6961.811	-6350.262
G7	24.306	24.32190	24.307	29.82257	24.642
G8	0.0958250	0.0958250	0.095825	0.0958250	0.095825
G9	680.630	680.6304	680.630	680.6886	680.763
G10	7049.33	7102.265	7054.316	7229.3908	8835.655
G11	0.75	0.749	0.75	0.749	0.75

5 Conclusions

A new adaptive parameter-less penalty scheme which is suitable for implementation within steady-state genetic algorithms has been proposed in order to tackle constrained optimization problems. Its main feature, besides being adaptive and not requiring any parameter, is to automatically define a different penalty coefficient for each constraint. The scheme was introduced in a real-coded steady-state

GA and, using available operators from the literature, produced results competitive with the best available in the EC literature, besides alleviating the user from the delicate and time consuming task of setting penalty parameters.

Acknowledgements. The authors acknowledge the support received from CNPq and FAPEMIG. The authors would also like to thank the reviewers for the corrections and suggestions which helped improve the quality of the paper.

References

1. M. Schoenauer and Z. Michalewicz. Evolutionary computation at the edge of feasibility. In *Parallel Problem Solving from Nature - PPSN IV*, volume 1141, pages 245–254. Springer-Verlag, 1996. LNCS.
2. S. Koziel and Z. Michalewicz. Evolutionary algorithms, homomorphous mappings, and constrained parameter optimization. *Evolutionary Computation*, 7(1):19–44, 1999.
3. G.E. Liepins and W.D. Potter. A genetic algorithm approach to multiple-fault diagnosis. In Lawrence Davis, editor, *Handbook of Genetic Algorithms*, chapter 17, pages 237–250. Van Nostrand Reinhold, New York, New York, 1991.
4. D. Orvosh and L. Davis. Using a genetic algorithm to optimize problems with feasibility constraints. In *Proc. of the First IEEE Conf. on Evolutionary Computation*, pages 548–553, 1994.
5. H. Adeli and N-T. Cheng. Augmented lagrangian genetic algorithm for structural optimization. *Journal of Aerospace Engineering*, 7(1):104–118, January 1994.
6. H.J.C. Barbosa. A coevolutionary genetic algorithm for constrained optimization problems. In *Proc. of the Congress on Evolutionary Computation*, pages 1605–1611, Washington, DC, USA, 1999.
7. P.D. Surry and N.J. Radcliffe. The COMOGA method: Constrained optimisation by multiobjective genetic algorithms. *Control and Cybernetics*, 26(3), 1997.
8. T.P. Runarsson and X. Yao. Stochastic ranking for constrained evolutionary optimization. *IEEE Trans. on Evolutionary Computation*, 4(3):284–294, 2000.
9. A.H.C. van Kampen, C.S. Strom, and L.M.C. Buydens. Lethalization, penalty and repair functions for constraint handling in the genetic algorithm methodology. *Chemometrics and Intelligent Laboratory Systems*, 34:55–68, 1996.
10. Z. Michalewicz and M. Schoenauer. Evolutionary algorithms for constrained parameter optimization problems. *Evolutionary Computation*, 4(1):1–32, 1996.
11. R. Hinterding and Z. Michalewicz. Your brains and my beauty: Parent matching for constrained optimization. In *Proc. of the Fifty Int. Conf. on Evolutionary Computation*, pages 810–815, Alaska, May 4-9 1998.
12. S. Koziel and Z. Michalewicz. A decoder-based evolutionary algorithm for constrained optimization problems. In *Proc. of the Fifth Parallel Problem Solving from Nature*. Springer-Verlag, 1998. Lecture Notes in Computer Science.
13. J.-H. Kim and H. Myung. Evolutionary programming techniques for constrained optimization problems. *IEEE Trans. on Evolutionary Computation*, 2(1):129–140, 1997.
14. H.J.C. Barbosa and A.C.C. Lemonge. An adaptive penalty scheme in genetic algorithms for constrained optimization problems. In *Proc. of the Genetic and Evolutionary Computation Conference*, pages 287–294. Morgan Kaufmann Publishers, 2002.

15. S.E. Carlson and R. Shonkwiler. Annealing a genetic algorithm over constraints. In *Proc. of the IEEE Int. Conf. on Systems, Man and Cybernetics*, pages 3931–3936, 1998.
16. D. Powell and M.M. Skolnick. Using genetic algorithms in engineering design optimization with non-linear constraints. In *Proc. of the Fifth Int. Conf. on Genetic Algorithms*, pages 424–430. Morgan Kaufmann, 1993.
17. K. Deb. An efficient constraint handling method for genetic algorithms. *Computer Methods in Applied Mechanics and Engineering*, 186(2-4):311–338, June 2000.
18. Z. Michalewicz. A survey of constraint handling techniques in evolutionary computation. In *Proc. of the 4th Int. Conf. on Evolutionary Programming*, pages 135–155, Cambridge, MA, 1995. MIT Press.
19. Z. Michalewicz, D. Dasgupta, R.G. Le Riche, and M. Schoenauer. Evolutionary algorithms for constrained engineering problems. *Computers & Industrial Engineering Journal*, 30(2):851–870, 1996.
20. R.G. Le Riche, C. Knopf-Lenoir, and R.T. Haftka. A segregated genetic algorithm for constrained structural optimization. In *Proc. of the Sixth Int. Conf. on Genetic Algorithms*, pages 558–565, 1995.
21. H. Homaifar, S.H.-Y. Lai, and X. Qi. Constrained optimization via genetic algorithms. *Simulation*, 62(4):242–254, 1994.
22. J.A. Joines and C.R. Houck. On the use of non-stationary penalty functions to solve nonlinear constrained optimization problems with GAs. In *Proc. of the First IEEE Int. Conf. on Evolutionary Computation*, pages 579–584, June 19–23 1994.
23. J.C. Bean and A.B. Alouane. A dual genetic algorithm for bounded integer programs. Dept. of Industrial and Operations Engineering, The University of Michigan, Tech. Rep. 92-53 1992.
24. D.W. Coit, A.E. Smith, and D.M. Tate. Adaptive penalty methods for genetic optimization of constrained combinatorial problems. *INFORMS Journal on Computing*, 6(2):173–182, 1996.
25. M. Schoenauer and S. Xanthakis. Constrained GA optimization. In *Proc. of the Fifth Int. Conf. on Genetic Algorithms*, pages 573–580. Morgan Kaufmann Publishers, 1993.
26. S. Ben Hamida and M. Schoenauer. ASCHEA: new results using adaptive segregational constraint handling. In *Proc. of the 2002 Congress on Evolutionary Computation*, volume 1, pages 884–889, May 2002.
27. J.A. Wright and R. Farmani. Genetic algorithms: A fitness formulation for constrained minimization. In *GECCO 2001: Proc. of the Genetic and Evolutionary Computation Conference*, pages 725–732. Morgan Kaufmann, 2001.
28. A.E. Eiben and J. I. van Hemert. Saw-ing EAs: adapting the fitness function for solving constrained problems. In D. Corne, M. Dorigo, and F. Glover, editors, *New ideas in optimization, chapter 26*, pages 389–402. McGraw-Hill, London, 1999.
29. Z. Michalewicz and D.B. Fogel. *How to Solve It: Modern Heuristics*. Springer-Verlag, 1999.
30. Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer-Verlag, New York, 1992.
31. H. Muhlenbein, M. Schomisch, and J. Born. The parallel genetic algorithm as function optimizer. *Parallel Computing*, 17(6-7):619–632, Sep 1991.
32. K. Deb and H.G. Beyer. Self-adaptive genetic algorithms with simulated binary crossover. *Evolutionary Computation Journal*, 9(2):197–221, 2001.
33. S.B. Hamida and M. Schoenauer. An adaptive algorithm for constrained optimization problems. In *PPSN VI – LNCS*, volume 1917, pages 529–538. Springer-Verlag, 2000.

Asynchronous Genetic Algorithms for Heterogeneous Networks Using Coarse-Grained Dataflow

John W. Baugh Jr.¹ and Sujay V. Kumar²

¹ North Carolina State University, Raleigh, NC 27695 USA
`john.baugh@ncsu.edu`

² NASA Goddard Space Flight Center, Greenbelt, MD 20771 USA
`sujay@hsb.gsfc.nasa.gov`

Abstract. Genetic algorithms (GAs) are an attractive class of techniques for solving a variety of complex search and optimization problems. Their implementation on a distributed platform can provide the necessary computing power to address large-scale problems of practical importance. On heterogeneous networks, however, the performance of a global parallel GA can be limited by synchronization points during the computation, particularly those between generations. We present a new approach for implementing asynchronous GAs based on the dataflow model of computation — an approach that retains the functional properties of a global parallel GA. Experiments conducted with an air quality optimization problem and others show that the performance of GAs can be substantially improved through dataflow-based asynchrony.

1 Introduction

Numerous studies have sought to exploit the inherent parallelism in GAs to achieve better performance. A recent report by Cantu-Paz [4] surveys the extensive research in this area and categorizes techniques for parallelization. One of the more straightforward techniques is global parallelization, in which the evaluation of individuals is performed in parallel [3]. Certain variations on global parallel GAs, such as evolving independent subpopulations [8] and hierarchically evolving populations [7], have also been developed. These and other global parallel GAs are *synchronous* in the sense that computations involving subsequent generations may not proceed until those of the current generation are complete. The speedup lost as a result of these synchronization points can be significant, particularly in a heterogeneous, networked environment, since the presence of a single slow processor can impede the overall progress of the GA.

The limitations of global parallel GAs due to end-of-generation synchronization points have been studied by a number of researchers. Most of the reported approaches use localized evolution strategies such as island-based approaches [5, 9] to achieve asynchrony. However, approaches other than global parallelization introduce fundamental changes in the structure of a GA [3]. For example, island-based GAs work with multiple interacting subpopulations whose parameters for

interaction require additional, problem-specific tuning. Poor settings can result in either convergence to an inferior solution or suboptimal parallel performance. Steady-state GAs [10], which work with a single evolving population, are another means of eliminating end-of-generation synchronization points. Instead of placing offspring in subsequent populations, such GAs return them to the original population by an operator that selects individuals to be replaced. In addition to suffering in some cases from problems of premature convergence, steady-state GAs, like island-based approaches, introduce fundamental changes in the GA.

In this paper, we present a new approach for implementing asynchronous GAs that is functionally equivalent to a global parallel GA, and hence to a sequential GA as well. By *functionally equivalent* we mean that the outputs are determined by precisely the same numerical operations and are likewise identical. Equivalence is achieved by “unrolling” the main loop of a global parallel GA, i.e., the loop responsible for advancing from one generation to the next. Inter-generational data dependencies are then captured formally using dataflow graphs, which enable the concurrent processing of multiple generations to the extent allowed by those dependencies. The benefits of functional equivalence between sequential and parallel implementations are substantial. Numerical results obtained from either implementation can be compared one-to-one with assurance that artifacts have not been introduced via parallelization. Further, the additional parameter tuning required when moving from sequential to parallel runs of a GA need not be repeated.

While applicable in other contexts, our approach targets GAs on heterogeneous workstation networks that may need hours, days, or even weeks to complete. In such a scenario participating computers may vary over time in their availability and in the resources that are committed to a given GA run. This type of variability imposes severe performance penalties when extraneous synchronization points are encountered. For all its benefits with compute-intensive runs, though, it is equally appealing that the approach adds very little computational overhead: it is lightweight enough to be imperceptible on runs taking well under a minute to complete.

2 Dataflow Principles

Dataflow [6] is a term that refers to algorithms or machines whose order of execution is based on the availability and forwarding of data. A dataflow program is a directed graph with nodes that represent operators and directed arcs that represent data dependencies. Nodes are computational tasks, and may be primitive machine-level instructions or arbitrarily complex functions. As a result, the dataflow model is applicable to fine- or coarse-grained parallelism. In addition to supporting varying levels of parallelism, the dataflow model also supports various types of parallelism. For instance, *vectorizing* and *pipelining* are simply special cases of standard flow graphs.

In the dataflow model, data values are carried on tokens, which travel along arcs, which we model as one-place buffers. The status of nodes can be determined

by a simple firing rule: A node is said to be *firable* when the data it needs are available. When a node is fired, its input tokens are absorbed. The computation is performed and the result is sent to its output arcs for other nodes to use. There is no communication between tasks — each task simply receives and outputs data.

The dataflow model has the following properties [1]:

- *parallelism*: nodes may execute in parallel unless there is an explicit data dependence between them;
- *determinacy*: results do not depend on the relative ordering in which nodes execute.

The natural parallelism in the dataflow model occurs because it does not force over-specification of an algorithm. The firing rule only says when a node *can* fire. It does not require that it be executed at any particular time.

3 Using Dataflow for Asynchrony

A synchronous distributed GA (SDGA) based on global parallelism begins with an initial population from which subsequent ones are obtained through a selection process. Here we assume the use of a binary tournament scheme, which selects two individuals at random, evaluates their fitnesses remotely, and produces a single winner. To generate a new population of size P this process is performed P times. Processor loads are dynamically balanced by placing evaluation requests in a task pool. Crossover and mutation operators are then applied and the entire process is repeated until convergence.

The “repeat until convergence” part of the above algorithm forces synchronization at the end of each generation since individuals in subsequent generations cannot be evaluated until all of their individuals are in place. An asynchronous distributed GA (ADGA) is obtained by “unrolling” this loop and building dataflow graphs that capture the algorithm’s inter-generational data dependencies. Intuitively, once a sufficient number of individuals have been evaluated in one generation, some of their offspring can be produced and undergo evaluation, even before the prior generation is complete. The extent to which generations are processed concurrently is limited only by the data dependencies derived from the synchronous implementation. Typically a “band” of 2 to 4 generations is active at any one time as the computation unfolds.

Pseudo-code for an ADGA using dataflow is shown in Figure 1. Populations are constructed and named using the *new population* procedure, which initiates enough *dataflow* threads (or “lightweight” processes) to carry out the genetic operations necessary for that generation. As each *dataflow* thread completes its task, the resulting offspring are placed in the subsequent generation. The *succ* function finds and returns the subsequent (or “successor”) generation: if it does not exist it is created via *new population*, which has the side effect of forking a new round of *dataflow* threads for the next generation unless a termination condition is met.

```

 $P_{final} = \text{empty}$ 
main
  new population ( $P_0$ )
  while  $P_{final}$  is empty do wait
  return fittest from  $P_{final}$ 

procedure new population ( $P_t$ )
  if termination condition met
  then  $P_{final} = P_t$ 
  else start  $n/2$  threads: dataflow ( $P_t$ )

thread dataflow ( $P_t$ )
  place 4 random individuals from  $P_t$  in graph
    (evaluate remotely, compete, mate)
  write 2 offspring into succ ( $P_t$ )

function succ ( $P_t$ )
  if  $P_{t+1}$  is empty
  then new population ( $P_{t+1}$ )
  return  $P_{t+1}$ 

```

Fig. 1. Pseudo-code for an Asynchronous GA

An illustration of a running ADGA program is shown in Figure 2. The figure depicts three active generations, each with a population of 10 individuals. Unshaded circles in each population denote empty token positions — a place to put an individual once it is produced. The initial population, $G1$, begins with randomly generated individuals so all of its circles are filled with tokens. The figure shows that some processing has already occurred. Dataflow graphs $D11$, $D12$, $D14$, and $D15$ have completed, as indicated by their dashed outlines and the fact that they have produced offspring (shaded circles) in generation $G2$. Dataflow graph $D13$, on the other hand, is still working: it has a solid outline and has yet to produce its offspring in generation $G2$. There is a mix of working and completed dataflow graphs in generation $G2$ as well. In generation $G3$, however, no dataflow graphs have completed, and some are still waiting for input. No space will be allocated for generation $G4$ until one of the graphs in $G3$ is ready to produce its offspring.

The inputs to each dataflow graph are the randomly selected individuals that will be used in the genetic operations. For instance, dataflow graph $D13$ takes individuals 7, 2, 5, and 0 from generation $G1$ and produces its offspring in positions 4 and 5 of generation $G2$. This behavior is more clearly seen in Figure 3, which provides a detailed view of dataflow graph $D13$. As shown in the figure, individuals 7 and 2 compete for position 4, and individuals 5 and 0 compete for position 5. This processing is performed by nodes in the graph, each

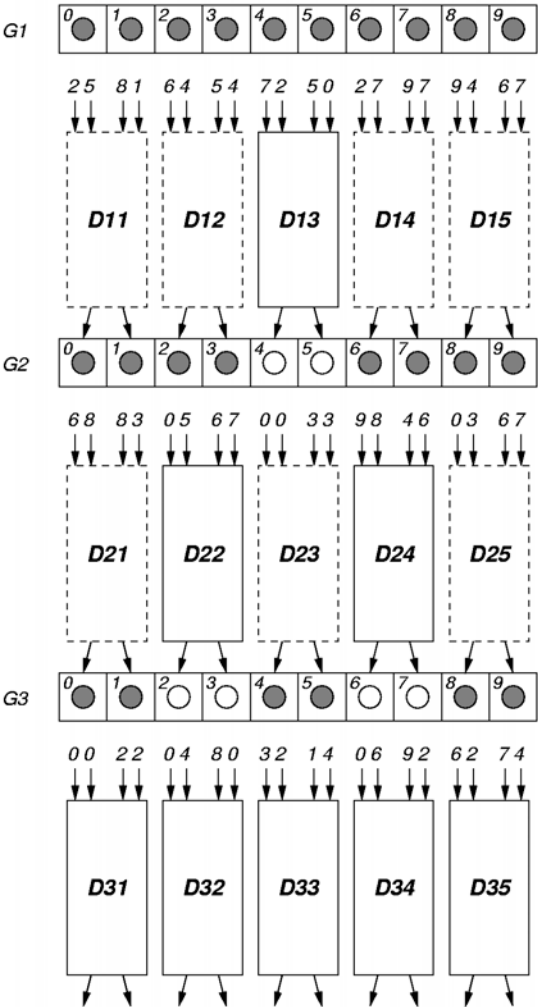


Fig. 2. Dataflow Graphs Dynamically Unfolding

being implemented by concurrent threads that block until their requisite inputs are available.

The *Copy* nodes ensure that an individual can be selected and processed simultaneously by other dataflow graphs. The need to copy is a result of having data flow through the model via tokens instead of being referenced as variables — a fundamental requirement of the dataflow model. Pointer copying is sufficient here, ensuring implementation efficiency. *Compare* nodes are used to keep track of an incumbent organism — the fittest seen during the GA run. Other nodes in the graph — *Evaluate*, *Compete*, and *Mate* — perform the usual genetic

operations. True parallelism is obtained in the implementation of the *Evaluate* nodes, which place in a task pool a request to evaluate the individual's fitness on a remote processor; each blocks until the result becomes available.

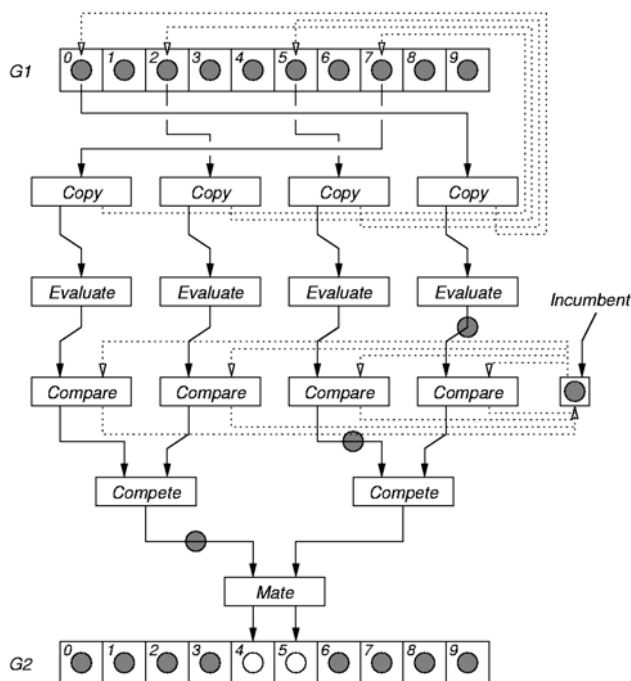


Fig. 3. Details of Dataflow Graph *D13*

4 Analysis and Results

Realizations of the SDGA and ADGA approaches, as described above, have been conveniently implemented in the Java programming language using its multi-threading capabilities and socket libraries for network communication. The implementations have been shown to be both efficient and portable across multiple platforms and operating systems — even within a single GA run. Experiments have been conducted with homogeneous as well as heterogeneous systems of processors, and simple empirical models have been developed to predict execution times. We begin by describing these models and then comparing predicted results with those obtained on a simple 0/1 knapsack problem and on a more complex air quality management problem.

4.1 Homogeneous System of Processors

Consider a homogeneous network of computers consisting of N identical processors. For a single generation of a GA to complete, P organisms must be evaluated. It is assumed that all of the N processors start simultaneously, and that each takes time t_{comp} to execute a fitness evaluation and time t_{comm} for communication with the client. The tasks associated with the GA can then be laid out in blocks, with each block representing the tasks performed by N processors in time $t_{comp} + t_{comm}$, as shown in Figure 4.

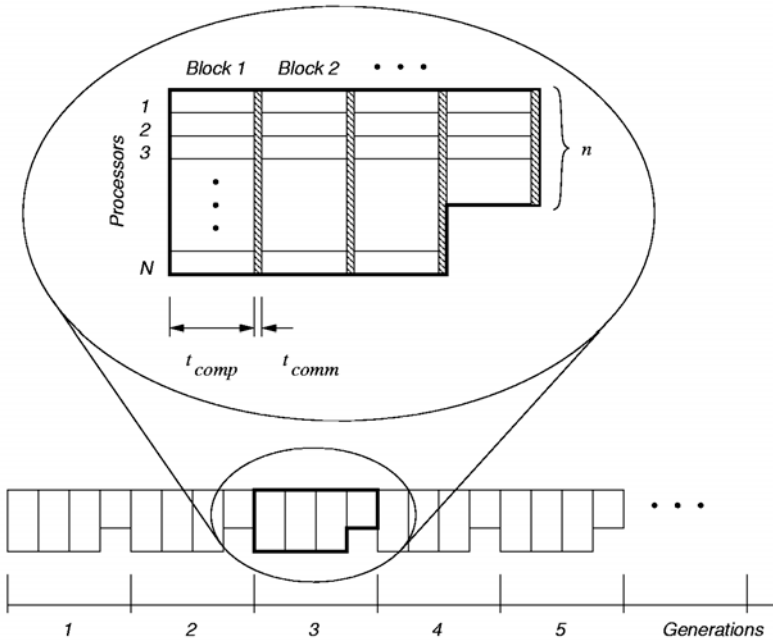


Fig. 4. GA Tasks Executing on N Homogeneous Processors

The pattern of blocks repeats itself until the end of a generation, at which point some number of evaluations n remain to be performed. Since N individuals are evaluated in each block, the total number of blocks in a generation is equal to $\lceil \frac{P}{N} \rceil$. From the figure, the time taken for a single generation (T_g) and the total time taken by an SDGA (T_{sync}) can be estimated as:

$$T_g = \lceil \frac{P}{N} \rceil (t_{comp} + t_{comm}) \quad (1)$$

$$\begin{aligned}
 T_{sync} &= T_g G \\
 &= \left\lceil \frac{P}{N} \right\rceil (t_{comp} + t_{comm}) G
 \end{aligned} \tag{2}$$

In the case of an ADGA, the processors are not constrained by the lack of available tasks at the end of a generation since, in practice, a sufficient number are available from subsequent generations to avoid idling. The total number of tasks in an ADGA evaluation is PG . Since there are N processors the total time taken by an ADGA (T_{async}) can be estimated as:

$$T_{async} = \frac{PG}{N} (t_{comp} + t_{comm}) \tag{3}$$

4.2 Heterogeneous System of Processors

To model a heterogeneous system, n_s identically slow processors are introduced into the system of N processors. Each of these slow processors is assumed to require a factor of f more processing time to evaluate an individual. The quantities t and t_{slow} are defined to be the sum of t_{comp} and t_{comm} for fast and slow processors, respectively. As with the homogeneous case, the tasks on a heterogeneous system can be laid out in blocks, where in this case each block is of width t_{slow} . Figure 5 shows GA tasks on a heterogeneous system with a single slow processor and f equal to 4. As depicted in the figure, for an SDGA, the presence of a slow processor clearly leaves idle a large number of faster processors.

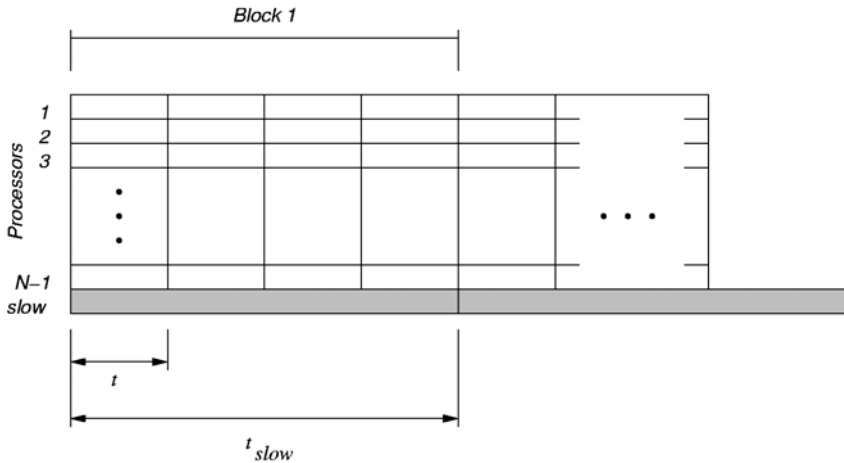


Fig. 5. GA Tasks Executing on Heterogeneous Processors

The number of blocks in a generation can be estimated as:

$$n_b = \lceil \frac{P}{f(N - n_s) + n_s} \rceil \quad (4)$$

Depending on the ordering of tasks, the number of tasks that remain at the end of generation becomes important. The number present in the final block of a generation (δ_1) can be estimated as:

$$\delta_1 = P - (n_b - 1)(f(N - n) + n) \quad (5)$$

If there are more tasks in the last block than fast processors, the slow processors will receive tasks to evaluate. Taking these factors into account, the total time taken by an SDGA can be estimated as:

$$T_{sync} = \begin{cases} (n_b - 1)ftG + tG & \text{if } \delta_1 \leq (N - n_s) \\ n_b t f G & \text{otherwise} \end{cases} \quad (6)$$

Since end-of-generation synchronizations are eliminated in an ADGA, the overall GA execution can be thought of as an ordering of PG tasks among processors. The number of blocks is estimated as:

$$n_b = \lceil \frac{PG}{f(N - n_s) + n_s} \rceil \quad (7)$$

At the end of the GA execution, if the last block contains more tasks than the number of fast processors, the slow processors will be involved in the final computations. The number of tasks present in the final block of GA execution (δ_2) can be estimated as:

$$\delta_2 = PG - (n_b - 1)(f(N - n_s) + n_s) \quad (8)$$

The estimated time taken by an ADGA is:

$$T_{async} = \begin{cases} (n_b - 1)ft + t & \text{if } \delta_2 \leq (N - n_s) \\ n_b t f & \text{otherwise} \end{cases} \quad (9)$$

4.3 0/1 Knapsack Problem

The 0/1 knapsack problem is representative of the large class of problems known as combinatorial optimization problems. Informally stated, the objective of the knapsack problem is to select items that maximize profit without exceeding capacity. As such, the problem is fine grained since fitness evaluation is typically inexpensive.

Both SDGA and ADGA implementations are applied to the 0/1 knapsack problem with anywhere from 3 to 30 processors. To assess their scalability with increased problem size, fitness evaluation times are artificially varied to achieve four different levels of granularity based on the ratio of t_{comp} to t_{comm} . Since

t_{comm} is approximately 250 milliseconds in our set up, t_{comp} times are artificially set to 250, 500, 750 and 1000 milliseconds, resulting in granularity factors of 1 through 4. To simulate a heterogeneous system, a slow processor is introduced with f set to 5. GA runs conducted with a population size of 100 for 200 generations yield the results shown in Figure 6. Although t_{comp} and t_{comm} are underpredicted in the model, the trends are as expected, with execution times increasing with problem granularity, and the ADGA scaling better than the SDGA.

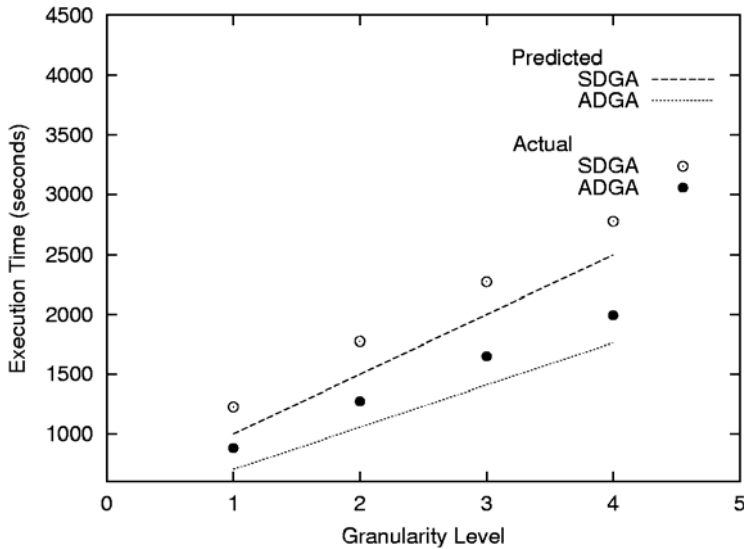


Fig. 6. Execution Time vs. Granularity using 15 Processors: 0/1 Knapsack Problem

4.4 Air Quality Optimization

Tropospheric ozone formed from the emissions of vehicles and industrial sources is considered a major pollutant. As a result, air quality management strategies may be necessary for geographic regions containing hundreds of sources, with each in turn having thousands of processes. Formal search strategies using GAs can be applied to find cost-effective ways of reducing ozone formation. For instance, an ambient least cost (ALC) model [2] is an optimization approach that incorporates source marginal control costs and emission dispersion characteristics to compute the source emissions at the least cost. A number of modeling techniques can be used to determine dispersion characteristics, such as the Empirical Kinetic Modeling Approach (EKMA), a Lagrangian box model that is used in this study. Because of the execution times typically required for EKMA, this GA formulation is somewhat coarse grained.

Experiments for an air quality management study around Charlotte, NC, were conducted on a network of workstations with as many as 19 processors. To simulate a heterogeneous system, a slow processor with an f factor of 5 is used. In each case, the GA was run for 50 generations using a population size of 50. The execution times are found to be in close agreement with the values predicted by the empirical model, as shown in Figure 7. Better agreement here than in the knapsack problem is likely due to increased problem granularity. Similar to earlier trends, the SDGA is outperformed by the ADGA; the execution times of the SDGA follow a step function pattern implying that, in between each step, there is no marginal benefit in using additional processors.

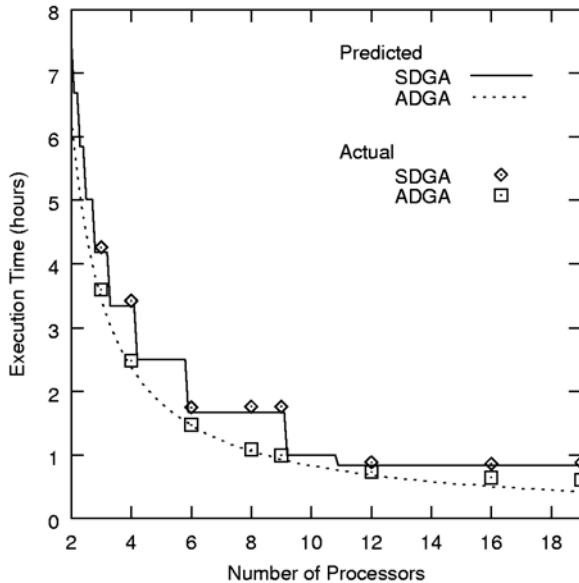


Fig. 7. Execution Time vs. Processors: Air Quality Optimization

5 Final Remarks

The growing acceptance of GAs has led to widespread use and attempts at solving larger and more challenging problems. A practical approach for doing so may rest on the ability to use available computer resources efficiently. Motivating the algorithmic developments in this paper is the expectation that a heterogeneous collection of personal computers, workstations, and laptops should be able to contribute their cycles to the solution of substantial problems without inadvertently detracting from overall performance. Removing the end-of-generation synchronization points from global parallel GAs is necessary to meet this expectation. The application of loop unrolling and dataflow modeling described herein

has been shown to be effective in keeping available processors from idling even when substantial variations exist in the processors' capabilities.

Although other asynchronous approaches might be used, one that is functionally equivalent to a simple, sequential GA offers real benefits with respect to parameter tuning. In a significant study on air quality management [references temporarily withheld for blind review process], our research team was able to move with little effort between atmospheric models that varied widely in their computational demands — from simple ones that can be solved using sequential GAs, to ones that require 20 minutes to evaluate a single individual on a high-end workstation: the same basic algorithm and parameters could be (and were) used in either case.

The GA implementations described in this paper are part of *Vitri*, an object-oriented framework implemented in Java for high-performance distributed computing [references temporarily withheld for blind review process]. Among its features are basic support for distributed computing and communication, as well as visual tools for evaluating run-time performance, and modules for heuristic optimization. It balances loads dynamically using a client-side task pool, allows the addition or removal of servers during a run, and provides fault tolerance transparently for servers and networks.

References

1. Arvind and D. E. Culler. Dataflow architectures. *Annual Reviews in Computer Science*, 1:225–253, 1986.
2. S. E. Atkinson and D. H. Lewis. A cost-effective analysis of alternative air quality control strategies. *Journal of Environmental Economics*, pages 237–250, 1974.
3. E. Cantu-Paz. Designing efficient master-slave parallel genetic algorithms. Technical report, University of Illinois at Urbana-Champaign, Urbana, IL, 1997.
4. E. Cantu-Paz. A survey of parallel genetic algorithms. Technical Report 97003, University of Illinois at Urbana Champaign, May 1997.
5. V. Coleman. The DEMO mode: An asynchronous genetic algorithm. Technical Report UM-CS-1989-033, University of Massachusetts, May 1989.
6. Computer. Special issue on data flow systems. 15(2), 1982.
7. J. Kim and P. Zeigler. A framework for multiresolution optimization in a parallel/distributed environment: Simulation of hierarchical GAs. *Journal of Parallel and Distributed Computing*, 32:90–102, 1996.
8. Yu-Kwong Kwok and Ahmad Ishfaq. Efficient scheduling of arbitrary task graphs to multiprocessors using a parallel genetic algorithm. *Journal of Parallel and Distributed Computing*, 47:58–77, 1997.
9. M. G. Schleuter. ASPARAGAS: An asynchronous parallel genetic optimization strategy. *Proceedings of the Third International Conference on Genetic Algorithms*, pages 422–427, 1989.
10. J. E. Smith and T. C. Fogarty. Self adaptation of mutation rates in a steady state genetic algorithm. In *Proceedings of IEEE International Conference on Evolutionary Computing*, volume 72, pages 318–323, 1999.

A Generalized Feedforward Neural Network Architecture and Its Training Using Two Stochastic Search Methods

Abdesselam Bouzerdoun¹ and Rainer Mueller^{2*}

¹ School of Engineering and Mathematics
Edith Cowan University, Perth, WA, Australia
a.bouzerdoun@ieee.org

² University of Ulm, Ulm, Germany

Abstract. *Shunting Inhibitory Artificial Neural Networks (SIANNs)* are biologically inspired networks in which the synaptic interactions are mediated via a nonlinear mechanism called *shunting inhibition*, which allows neurons to operate as adaptive nonlinear filters. In this article, The architecture of SIANNs is extended to form a *generalized feedforward neural network* (GFNN) classifier. Two training algorithms are developed based on stochastic search methods, namely *genetic algorithms* (GAs) and a *randomized search method*. The combination of stochastic training with the GFNN is applied to four benchmark classification problems: the XOR problem, the 3-bit even parity problem, a diabetes dataset and a heart disease dataset. Experimental results prove the potential of the proposed combination of GFNN and stochastic search training methods. The GFNN can learn difficult classification tasks with few hidden neurons; it solves perfectly the 3-bit parity problem using only one neuron.

1 Introduction

Computing has historically been dominated by the concept of *programmed computing*, in which algorithms are designed and subsequently implemented using the dominant architecture at the time. An alternative paradigm is *intelligent computing*, in which the computation is distributed and massively parallel and learning replaces *a priori* program development. This new, biologically inspired, intelligent computing paradigm is called *Artificial Neural Networks* (ANNs) [1]. ANNs have been used in many applications where the conventional programmed computing has immense difficulties, such as understanding speech and handwritten text, recognizing objects, etc. However, an ANN needs to learn the task at hand before it can be operated in practice to solve the real problem. Learning is accomplished by a training algorithm. To this end, a number of different training methods have been proposed and used in practice.

* R. Mueller was a visiting student at ECU for the period July 2001 to June 2002.

Another biologically inspired computing paradigm is *genetic* and *evolutionary algorithms* [2],[3]. Evolutionary algorithms are stochastic search methods that mimic the metaphor of natural biological evolution. They operate on population of potential solutions applying the principle of survival of the fittest. The combination of these two biologically inspired computing paradigms is a powerful instrument for solving problems in pattern recognition, signal and image processing, machine vision, control, etc.. The aim in this article is to combine a *Generalized Feedforward Neural Network* (GFNN) architecture with genetic algorithms to design a new class of artificial neural networks that has the potential to learn complex problems more efficiently.

In the next section, the *generalized shunting neuron* and the GFNN architecture are introduced. Two training methods for the GFNN architecture are presented in section 3. First the randomized search method is presented in Subsection 3.1, then the GA technique in Subsection 3.2. The developed training algorithms are tested with some common benchmark problems in Section 4, followed by concluding remarks and future work in Section 5.

2 The Generalized Feedforward Neural Network Architecture

In [4] Bouzerdoun introduced the class of *shunting inhibitory artificial neural networks* (SIANNs) and used them for classification and function approximation. In this section, we extend SIANNs to form a generalized feedforward neural network architecture. But before describing the generalized architecture, we first introduce the elementary building block of the architecture, namely the *generalized shunting inhibitory neuron*.

2.1 Generalized Shunting Inhibitory Neuron

The output of a generalized shunting inhibitory neuron is given by

$$x_j = \frac{f(\sum_i w_{ji}I_i + w_{j0})}{a_j + g(\sum_i c_{ji}I_i + c_{j0})} = \frac{f(\mathbf{w}_j \cdot \mathbf{I} + w_{j0})}{a_j + g(\mathbf{c}_j \cdot \mathbf{I} + c_{j0})} \quad (1)$$

where x_j is the activity (output) of neuron j ; I_i is the i th input; c_{ji} is the “shunting inhibitory” connection weight from input i to neuron j ; w_{ji} is the connection weight from input i to neuron j ; w_{j0} and c_{j0} are bias constants; a_j is a constant preventing the division by zero, by keeping the denominator always positive; f and g are activation functions. The name shunting inhibition comes from the fact that a high term in the denominator tends to suppress (or inhibit in a shunting fashion) the activity caused by the term in the numerator of (1).

2.2 The Network Architecture

The architecture of the generalized feedforward neural network is similar to that of a *Multilayer Perceptron Network* [1], and is shown in Fig. 1. The network

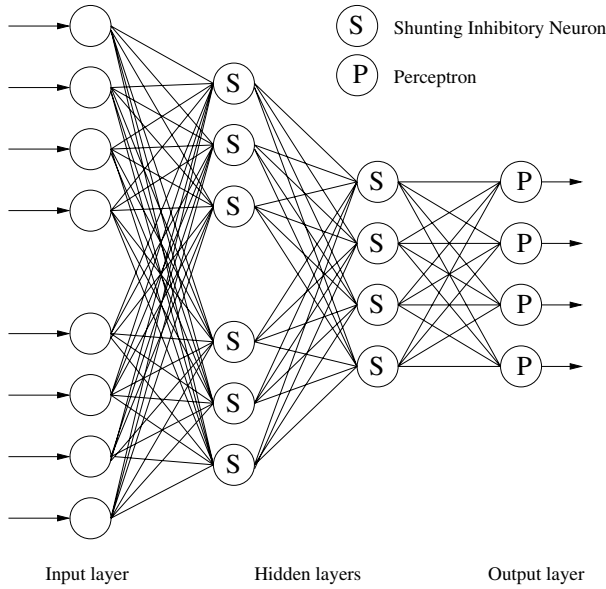


Fig. 1. Generalized Feedforward Neural Network architecture (GFNN).

consists of many layers, each of which has a number of neurons. The input layer only acts as a receptor that receives inputs from the environment and broadcasts them to the next layer; therefore, no processing is done in the input layer. The processing in the network is done by the hidden and output layers. Neurons in each layer receive inputs from the previous layer, process them and then pass their outputs to the next layer. Hidden layers are so named because they have no direct connection with the environment. In the GFNN architecture, the hidden layers consist of only generalized shunting inhibitory neurons. The role of the shunting inhibitory layers is to perform a nonlinear transformation on the input data so that the results can easily be combined by the output neurons to form the correct decision. The output layer, which may be a linear or sigmoidal type (i.e., perceptron), is different from the hidden layers; each output neuron basically calculates the weighted sum of its inputs followed by an appropriate activation function. The response, y , of an output neuron is given by

$$y = h(\mathbf{w}_o \cdot \mathbf{x} + b) \quad (2)$$

where $\mathbf{x}$ is the input vector $\mathbf{w}_o$ is the weight vector, b is the bias constant, and h is the activation function, which may be a linear or a sigmoid function.

3 Training Methods

An artificial neural network needs to be trained instead of being *a priori* programmed. Supervised learning is a form of learning in which the target values are

included part of the training data. During the training phase, the set of training data is repeatedly applied to the network and the weights of the network are adjusted until the difference between the target values and the network output values is within the desired tolerance.

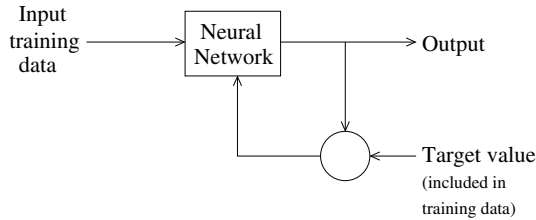


Fig. 2. Supervised learning: the weights are adjusted until the target values are reached.

In this section, two different methods for training the GFNN are described: the Random Optimization Method (ROM) and the GA based method. Since GAs are known for being able to find good solutions for many complex optimization problems, this training method is of particular interest to us.

3.1 Random Optimization Method (ROM)

The ROM is employed because it is a simple method to implement and intuitively appealing. It is used to test the network structure before the GA is applied, and serves as a benchmark for comparing the GA based training method. The ROM searches the weight space by generating randomized vectors in the weight space and testing them. The basic ROM procedure is as follows [1]:

1. Randomly choose a weight vector $\mathbf{W}$ and a small vector $\mathbf{R}$.
2. If the output of the net $\mathbf{Y}(\mathbf{W} + \mathbf{R})$ is better than $\mathbf{Y}(\mathbf{W})$ then $\mathbf{W} = \mathbf{W} + \mathbf{R}$.
3. Check for termination criteria, end the algorithm when one of the termination criteria is achieved.
4. Randomly choose a new $\mathbf{R}$ and go to step (2)

There are some obvious extensions to the above algorithm which we have implemented. The first one implements reverse side checking. This means instead of checking only $\mathbf{W} + \mathbf{R}$ we check $\mathbf{W} - \mathbf{R}$ as well. Furthermore, an orthogonal vector $\mathbf{R}^*$ is also checked in both directions. That alone wouldn't improve the algorithm much, but there is another extension. If there is an improvement in any of the four previous directions, simply extend the search in the same direction, instead of just generating another value of $\mathbf{R}$. The idea is that if $\mathbf{W} + \mathbf{R}$ gives an improved output $\mathbf{Y}$, then another scaled step $k \cdot \mathbf{R}$ in the same direction might be in a "downhill" direction, and hence a successful direction. All these extensions have been implemented to train the GFNN.

3.2 Genetic Algorithms (GAs)

The GAs are used as a training method because they are known for their ability to perform well on complex optimization problems. Furthermore, they are less likely to get trapped in local minima, a problem suffered by traditional gradient based training algorithms. GAs are stochastic search methods that mimic the metaphor of natural biological evolution. They operate on a population of potential solutions applying the principle of survival of the fittest to produce an improved approximation to a solution. At each generation, a new set of approximations is created by the process of selecting individuals according to their level of fitness in the problem domain and breeding them together using operators borrowed from natural evolution. This process leads to the evolution of populations of individuals that are better suited to their environment than the individuals they were created from, just as in natural adaptation.

GAs model natural evolutionary processes, such as selection, recombination, mutation, migration, locality and neighborhood. They work on populations of individuals instead of single solutions. Furthermore, simple GAs can be extended to multipopulation GAs. In multipopulation GAs several subpopulations are introduced, which evolve independently over few generations before one or more individuals are exchanged between the subpopulations. Figure 3 shows the structure of an extended multipopulation genetic algorithm.

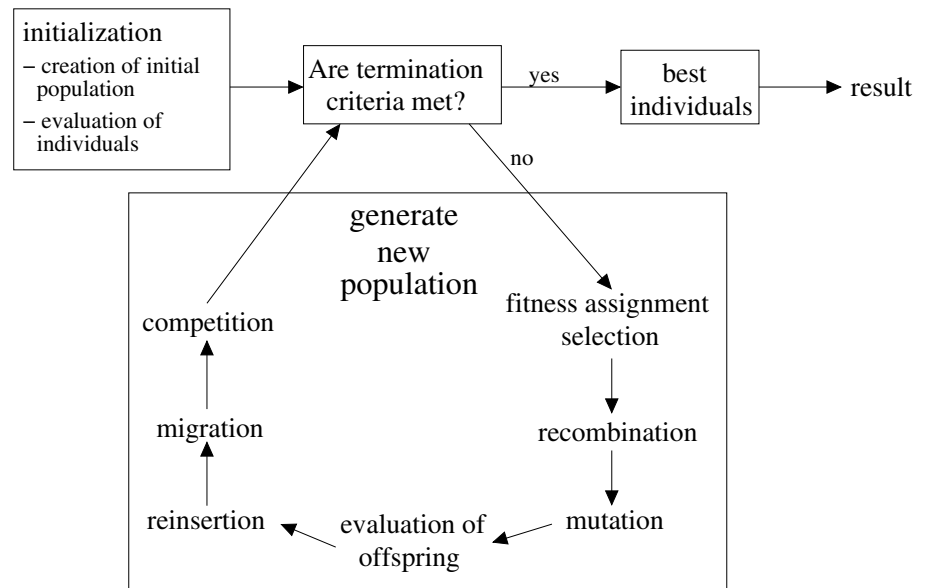


Fig. 3. Structure of an extended multipopulation genetic algorithm (adapted from [5]).

The genetic operators that can be applied to evolve the population depend on the variable representation in the GA: binary, integer, floating point (real), or symbolic. In this research, we employed the real variable representation because it is the most natural representation for weights and biases of neural networks. Furthermore, it has been shown that the real-valued GA is more efficient than the binary GA [3]. Some of the most common GA operators are described below.

Selection. Selection determines the individuals which are chosen for mating (recombination) and how many offsprings each selected individual produces. Each individual in the selection pool receives a reproduction probability depending on its own objective value and the objective values of all other individuals in the population. There are two fitness-based assignment methods: proportional fitness assignment and rank-based fitness assignment. The proportional fitness assignment assigns a fitness value proportional to the objective value, whereas the fitness value in a rank-based assignment depends only on the rank of the individual in a list sorted according to the objective values.

Roulette-wheel selection, also called “stochastic sampling with replacement” [6], maps the individuals to contiguous segments of a line, such that each individual’s segment is equal in size to its fitness [5]. The individual whose segment spans a generated random number, is selected.

In *stochastic universal sampling*, the individuals are mapped to N contiguous segments of a line (N being the number of individuals), each segment having a length proportional to its fitness. Then N equally spaced Pointers are placed above the line and the position of the first pointer is given by a randomly generated number in the range $[0, 1/N]$. Every pointer indicates a selected individual.

In *local selection* every individual interacts only with individual residing in its local neighborhood [5].

In *truncation selection* individuals are sorted according to their fitness and only the best individuals are selected as parents.

The *tournament selection* chooses randomly a number of individuals from the population and the best individual from this group is selected as parent. The process is repeated until enough mating individuals are found.

Recombination. The process of recombination produces new individuals by combining the information contained in the parents. There are different recombination methods depending on the variable representation. Discrete recombination can be used with all representations. In addition, there are two specific methods for real valued recombination, the *intermediate recombination* and the *line recombination*. In intermediate recombination the variables of the offspring are chosen somewhere around and between the variable values of the parents. Line recombination, on the other hand, generates the offspring on a line defined by the variable values of the parents.

Mutation. After recombination, every offspring undergoes a mutation, like in nature. Small perturbations mutate the offspring variables with low probability. Mutation of real variables means that randomly generated values are added to

the offspring variables with low probability. Thus, the probability of mutating a variable (mutation rate) and the size of change for each mutated variable (mutation step) must be defined. In our simulations, the mutaion rate is inversely proportional to the number of variables; the more variables an individual has, the smaller is the mutation rate.

Reinsertion. After an offspring is produced it must be inserted into the population. There are two different situations. First, the size of the offspring population produced is less than the size of the original population. In this case, the whole offspring population has to be inserted to maintain the size of the original population. Second more offsprings are generated than there are individuals in the original population. In this case, the reinsertion scheme determines which individuals should be reinserted into the new population and which individuals should be replaced by the offsprings. There are different schemes for reinsertion. *Pure reinsertion* produces as many offsprings as parents and replaces all parents by the offspings. *Uniform reinsertion* produces fewer offsprings than parents and replaces parents uniformly at random. *Elitist reinsertion* produces fewer offsprings than parents and replaces the worst parents. *Fitness based reinsertion* produces more offsprings than needed and reinserts only the best offsprings.

After reinsertion, one needs to verify if a termination criteria is met. If a criteria is met, then the cycle can be stopped; otherwise, the cycle will be repeated until a termination criteria is met. The GA parameters used in the simulations are presented in Table 1 below.

Table 1. Evolutionary algorithm parameters used in the simulations.

subpopulations	individuals	50 30 20 20 10
variable format	real values	
selection	function	selsus (stochastic universal sampling)
	pressure	1.7
	gen. gap	0.9
reinsertion	rate	1
recombination	name	discrete and line recombination
	rate	1
mutation	name	mutreal (real-valued mutation)
	rate	0.00826
	range	0.1 0.03 0.01 0.003 0.001
	precision	12
regional model		
migration	rate	0.1
competition	rate	0.1

The objective function to be minimized here is the mean squared error.

$$MSE = \frac{1}{N_p} \sum_{j=1}^{N_p} (y_j - d_j)^2 \tag{3}$$

where y_j is the output of the GFNN, d_j the desired output for input pattern $\mathbf{x}_j$, and N_p is the number of training patterns.

4 Experimental Results

Experiments were conducted to assess the ability of the proposed NN architecture to learn some difficult classification tasks. Four benchmark problems were selected to test the network architecture: two Boolean functions, the Exclusive-OR (XOR) and the 3-bit parity, and two medical diagnosis problems, the heart disease and diabetes. The heart disease and diabetes data sets were obtained from UCI Machine Learning Repository [7].

4.1 The XOR and 3-Bit Parity Problems

A two-layer network architecture consisting of two inputs, one or two hidden units, and an output unit is trained with XOR problem. For every network configuration, ten training runs, with different initializations, were performed using both the GA- and the ROM-based training algorithms. If during the training a network reaches an error of zero, training is halted. Table 2 summarizes the results: the first column indicates the f/g combination of activation functions (see Eq. (1)), along with the training algorithm. In all the simulations f was *hyperbolic tangent sigmoid* activation function, **tansig**, and g was either the *exponential* function, **exp**, or the *logarithmic sigmoid* activation function, **logsig**. The GA uses a search space ranging from -128 to 128, and hence is labeled GA128. The second column shows the number of training runs that achieved zero error. The “Best case error” column shows the lowest test error of trained networks. Note that even when an error of zero is not reached during training, the network can still learn the desired function after thresholding its output.

Table 2. Training with the XOR problem.

	Runs w. E=0	Aver. generation to reach E=0	Aver. time to reach E=0	Best case error	Mean error	Std
No. of neurons: 1 (hidden layer), 9 weights						
tansig/logsig GA128	1	620	15.89	0.00	25.50	790
tansig/logsig ROM	4	4423	4.56	0.00	15.00	1290
tansig/exp GA128	10	21	0.51	0.00	0.00	000
tansig/exp ROM	6	488	0.47	0.00	10.00	1290
No. of neurons: 2 (hidden layer), 17 weights						
tansig/logsig GA128	8	68	2.02	0.00	5.00	1054
tansig/logsig ROM	10	393	0.52	0.00	0.00	000
tansig/exp GA128	10	13	0.37	0.00	0.00	000
tansig/exp ROM	10	845	1.05	0.00	0.00	000

The best results were obtained using two neurons in the hidden layer with the exponential activation function, **exp**, in the denominator. Note that both training algorithms, GA and ROM, reached an error of zero at least once during

training. The GA was slightly faster with 0.37 minutes average time to reach an error of zero than the ROM, which needed 1.05 minutes. Figure 4 displays the percentage mean error vs. training time for the best combination of activation functions (tansig/exp). More importantly, however, is the fact that even with one hidden neuron and tansig/exp combination, ten out of ten runs reached an error of zero, with the GA as training algorithm. However, the time to reach an error of 0 was 0.51 minutes slightly longer than the time of the two neuron network. Also, we can observe that both the ROM and GA perform well in the sense of reaching runs with error zero. Furthermore, all trained were able to classify the XOR problem correctly.

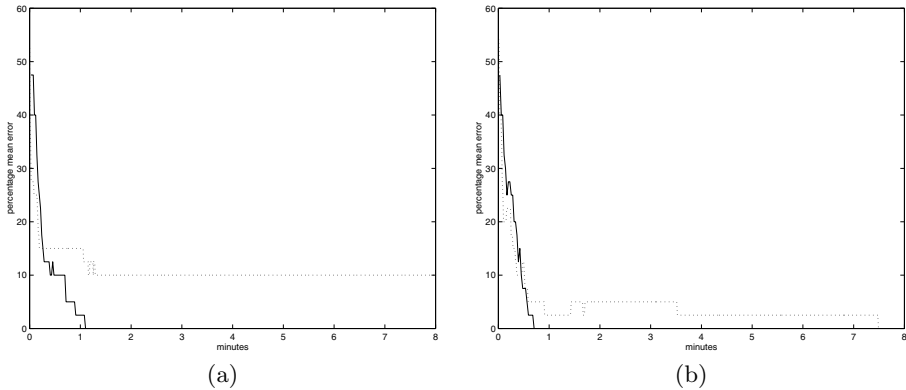


Fig. 4. Percentage mean error over time with tansig/exp as activation functions: (a) 1 hidden unit, (b) 2 hidden units. The dotted line is the result of the ROM and the solid line is the result of the GA.

For the 3-bit parity problem, the network architecture consists of three inputs, one hidden layer and one output unit of the perceptron type; the hidden layer comprises one, two or three shunting neurons. The same experiments as with the XOR problem were conducted with 3-bit parity; that is, ten runs for each architecture are performed with tansig/logsig or tansig/exp activation functions. Table 3 presents the result of the ten runs. None of the networks with **logsig** activation function in the denominator reach an error of zero during training. However, using the exponential activation function in the denominator, some networks with one hidden unit reach zero error during training and most networks, even those that do not reach zero error during training, learn to classify the even-parity correctly.

4.2 Diabetes Problem

The diabetes dataset has 768 samples with 8 input parameters and two output classes: presence (1) or absence (0) of diabetes. The dataset was partitioned into two sets: 50% of the data points were used for training and the other 50% for testing. The network architecture consisted of 8 input units, one hidden layer

Table 3. Training with the 3-bit even parity.

	Runs w. E=0	Aver. generation to reach E=0	Aver. time to reach E=0	Best case error	Mean error	Std
No. of neurons: 1 (hidden layer), 11 weights						
tansig/logsig GA128	0	NaN	NaN	12.50	20.00	6.45
tansig/logsig ROM	0	NaN	NaN	12.50	28.75	11.86
tansig/exp GA128	2	629	7.13	0.00	17.50	12.08
tansig/exp ROM	0	2720	1.36	0.00	20.00	10.54
No. of neurons: 2 (hidden layer), 21 weights						
tansig/logsig GA128	0	NaN	NaN	12.50	22.50	5.27
tansig/logsig ROM	0	7320	4.99	0.00	18.75	8.84
tansig/exp GA128	6	243	3.33	0.00	6.25	8.84
tansig/exp ROM	4	11180	6.56	0.00	7.50	6.45
No. of neurons: 3 (hidden layer), 31 weights						
tansig/logsig GA128	3	753	12.58	0.00	12.50	10.21
tansig/logsig ROM	3	4770	6.59	0.00	13.75	10.94
tansig/exp GA128	8	57	0.92	0.00	2.50	5.27
tansig/exp ROM	7	9083	12.04	0.00	3.75	6.04

of shunting neurons, and one output unit. The number of hidden units varied from one to eight. The size of the search space is also varied: $[-64, 64]$ (GA64), $[-128, 128]$ (GA128), $[-512, 512]$ (GA512). Again ten training runs for each architecture and each algorithm, GA and ROM, were performed. The network GA128 was also trained on a reduced data set (a quarter of the total data); this network is denoted GA128q. After training is completed, the generalization ability of each network is tested by evaluating its performance on the test set.

Figure 5 presents the percentage mean error of the training dataset. It can be observed that the tansig/exp activation function combination performs slightly better than the tansig/logsig. The ROM gets worse with increasing number of neurons, what we expected. The reason is that the one hidden-neuron configuration has 21 weights/biases whereas the 8 hidden-neuron configuration has 161

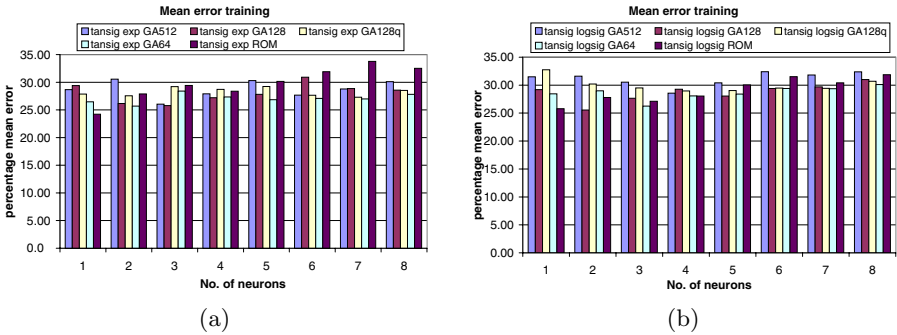


Fig. 5. Percentage mean error (train dataset) of the 10 runs: (a) tansig/exp, (b) tansig/logsig configuration.

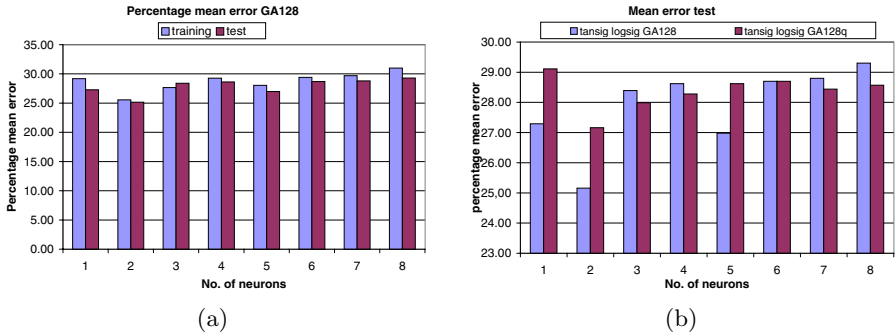


Fig. 6. (a) Percentage mean error of the GA128 on the training and test sets. (b) Generalization performance of GA128 and GA128q on the test set.

weights/biases. With increasing number of weights/biases the dimension of the search space increases, which leads to worse performance by the ROM.

In Fig. 6 the percentage mean error of the training dataset is compared with the percentage mean error of the test set; both are almost equal for all the different number of neurons. This shows that overfitting is not a serious problem.

4.3 Heart Disease Problem

The experimental procedure was the same as for the diabetes diagnoses problem, except that the data set has only 270 samples with 13 input parameters. This increases the number of parameters of the network and slows down the training process. To avoid being bogged down by the training process, only GA128 was trained on the Heart dataset. Figure 7(a) presents the mean error rates on the training set. Not surprising, the mean error rate of the ROM increases with increasing number of neurons. Figure 7(b) compares the performances of the GA on the training and test sets. The results of the heart disease problem are

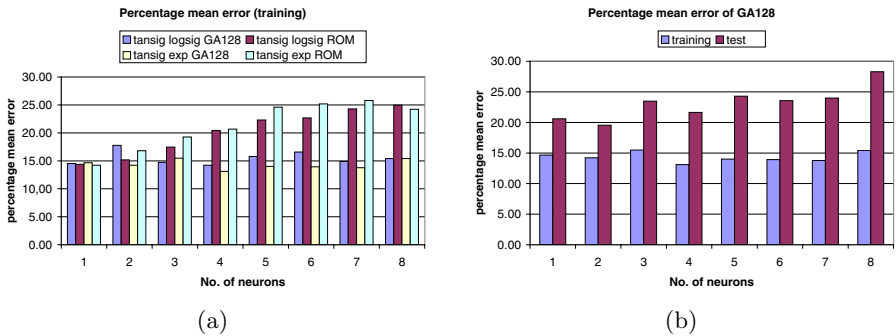


Fig. 7. Percentage mean error: (a) training set, (b) training set compared totest set.

similar to those of the diabetes diagnoses problem, except the errors are much lower; it is well known that the Diabetes problem is harder to learn than the Heart Disease problem.

5 Conclusions and Future Work

In this article we presented a new class of neural networks and two training methods: the ROM and the GA algorithms. As expected, the ROM works well for a small number of weights/biases but becomes worse as the number of parameters increases. The experimental results show that the presented network architecture, with the proposed learning schemes, can be a powerful tool for solving problems in prediction, forecasting and classification. It was shown that the proposed architecture can learn a Boolean function perfectly with a small number of hidden units. The tests on the two medical diagnosis problems, diabetes and heart disease, proved that the proposed architecture can learn complex tasks with good generalization ability and hardly any overfitting.

Some further work needs to be considered to improve the learning performance of the proposed architecture. Firstly, a suitable termination criteria must be found to stop the algorithm, which could be the classification error on a validation set. Secondly, the settings of the GA should be optimized. In this project only different sizes of the search space were used. To get better results other settings, e.g. size of population, mutation methods, should be optimized. Finally a combination of the GA and, e.g., gradient descent method can improve the results further. GAs are known for their global search and gradient methods for their local search; by combining the two, we should expect better results.

References

1. Schalkoff, R. J.: *Artificial Neural Networks*. McGraw-Hill 1997.
2. Goldberg, D. E.: *Genetic Algorithms in search, Optimization and Machine Learning*. Addison-Wesley, 1989.
3. Michalewicz, Z.: *Genetic Algorithms + Data Structures = Evolution Programs*, (2nd edition). Berlin, Heidelberg, New York: Springer-Verlag, 1994.
4. Bouzerdoum, A.: "Classification and function approximation using feed-forward shunting inhibitory artificial neural networks," *Proc. IEEE/INNS Int. Joint Conf. Neural networks (IJCNN-2000)*, Vol. VI, pp. 613–618, 24–27 July 2000, Como, Italy.
5. Pohlheim, H.: *Genetic and Evolutionary Algorithms: Principles, Methods and Algorithms*, 1999. <http://www.geatbx.com>.
6. Baker, J. E.: "Reducing bias and inefficiency in the selection algorithms," *Proc. Second Int. Conf. on Genetic Algorithms*, pp. 14–21, 1987.
7. Blake, C. L., Merz, C. J.: "UCI Repository of Machine Learning Databases," Dept. Information and Computer Science, University of California, Irvine, 1998.
8. Rooij, Jain and Johnson: *Neural Network Training using Genetic Algorithms*. World Scientific, 1996.

Ant-Based Crossover for Permutation Problems

Jürgen Branke, Christiane Barz, and Ivesa Behrens

Institute AIFB, University of Karlsruhe, 76128 Karlsruhe, Germany

`branke@aifb.uni-karlsruhe.de`

Abstract. Crossover for evolutionary algorithms applied to permutation problems is a difficult and widely discussed topic. In this paper we use ideas from ant colony optimization to design a new permutation crossover operator. One of the advantages of the new crossover operator is the ease to introduce problem specific heuristic knowledge. Empirical tests on a travelling salesperson problem show that the new crossover operator yields excellent results and significantly outperforms evolutionary algorithms with edge recombination operator as well as pure ant colony optimization.

1 Introduction

Crossover for evolutionary algorithms (EAs) applied to permutation problems is notoriously difficult, and many different crossover operators have been suggested in the literature. Ant colony optimization (ACO), however, seems particularly well suited for permutation problems. In this paper, we propose to hybridize these two approaches in a way that performs better than either of the original approaches. In particular, we design a new crossover operator, called ant-based crossover (ABX), which uses ideas from ACO within an EA framework.

In ACO, new solutions are constructed step by step based on a pheromone matrix which contains information about which decisions have been successful in the past. Furthermore, problem specific heuristic knowledge is usually used to influence decisions. In ABX, a temporary pheromone matrix is constructed based on the parents selected for mating. This temporary pheromone matrix is then used to create one or several children in the standard way employed by ACO. This has several interesting implications: First of all, it is now as easy as in ACO to incorporate problem-specific heuristic knowledge. Furthermore, we gain additional flexibility. For example, it is natural to extend ABX to construct children from more than two parents, or to integrate ACO as local optimizer. Finally, the use of a population allows us to explicitly maintain several different good solutions, which is not possible in pure ACO approaches.

While we do not see any reason why the proposed approach should not be successful on a wide range of permutation problems, in this paper we concentrate on the travelling salesperson problem (TSP). We empirically compare our approach with an evolutionary algorithm with edge recombination as well as a pure ACO algorithm.

The paper is structured as follows: the next section surveys related work and provides a brief overview on recombination operators for permutation problems as well as on ant colony optimization. In Section 3 we introduce the new ant-based crossover operator. The approach is evaluated empirically in Section 4. The paper concludes in Section 5 with a summary and ideas for future work.

2 Related Work

2.1 Permutation Crossover

Crossover for permutation problems is difficult, and has been discussed in the literature for a long time. Generally, a crossover operator should create feasible offspring by combining parental information in a sensible way. What is to be considered sensible also depends on the application at hand. For example, with regard to a TSP, it seems more important to preserve edges from the parents (i.e. direct adjacencies in the permutation), while for a scheduling problem, it is more important to preserve the general precedence relations (cf. [2]).

Standard one-point or multi-point crossover does not work for permutations, as it would generate infeasible offspring. The crossover operators suggested in the literature are numerous and range from simple approaches such as order crossover [5] or partially mapped crossover [8] to more complicated ones such as distance preserving crossover [7], edge assembly crossover [15], inner-over crossover [18], natural crossover [12], or edge recombination crossover [20]. The difficulty of designing a proper permutation crossover even led some researchers to abandon a permutation representation, and to use e.g. random keys encoding [1] instead.

For TSPs, edge recombination crossover (ERX) seems to be a very effective crossover operator as it is able to preserve more than 95% of the parental edges [20]. We will use it later for comparison with ABX and therefore discuss it here in slightly more detail: Starting from a random city, ERX iteratively constructs a tour. In each step, it first considers the (up to 4) cities that are neighbors (i.e. connected) to the current location in either of the two parents. If at least one of those has not been visited so far, it selects the city which has the fewest yet unvisited other cities as neighbors in the parents. Otherwise, a random successor is selected. For details, see [20].

There have also been attempts to incorporate problem-specific knowledge into the crossover operator. For example, Grefenstette [9] and Tang and Leung [17], propose variants of ERX, which, when they have to choose between parental edges, prefer the short ones. Julstrom and Raidl [11] compare several ways for preferring short edges within an ERX framework, for decisions between parental edges as well as for decisions when all parental edges are inadmissible. In effect, the latter approach comes quite close to the simplest form of ABX proposed here. Despite of this similarity, it still differs in the way the parental information and the heuristic information are combined. Furthermore, it lacks the whole general ACO framework, which allows us to e.g. additionally use ACO as local optimizer.

2.2 Ant Colony Optimization

Standard ACO: ACO is an iterative probabilistic optimization heuristic inspired by the way real ants find short paths between their nest and a food source. The fundamental principle used by ants for communication is stigmergy, i.e. ants use pheromones to mark their trails. A higher pheromone intensity suggests a better path and consequently inclines more ants to take a similar path.

Transferring these ideas to the artificial scenario of a TSP with n cities, an ACO approach works as follows (cf. [3,6]): In every iteration, a number of m (artificial) ants construct one solution each through all the given n cities. Starting at a random city, an ant iteratively selects the next city based on heuristic information as well as pheromone information. The heuristic information, denoted by η_{ij} , represents a priori heuristic knowledge w.r.t. how good it is to go from city i to city j . For TSPs, $\eta_{ij} = 1/d_{ij}$ where d_{ij} is the distance between city i and j . The pheromone values, denoted by τ_{ij} , are dynamically changed by the ACO algorithm and serve as a kind of memory, indicating which choices were good in the past.

When having inserted city i in the previous step, the next city j is chosen probabilistically according to the following probabilities:

$$p_{ij} = \frac{\tau_{ij}^\alpha \cdot \eta_{ij}^\beta}{\sum_{h \in S} \tau_{ih}^\alpha \eta_{ih}^\beta}, \quad (1)$$

where S is the set of cities that have not been visited yet, and α and β are constants that determine the relative influence of the heuristic and the pheromone values on the ant's decision.

After each of the m ants have constructed a solution, the pheromone information is updated. First, some of the old pheromone is evaporated on all edges according to $\tau_{ij} \mapsto (1 - \rho) \cdot \tau_{ij}$, where parameter $\rho \in (0, 1)$ specifies the evaporation rate. Afterwards, a fixed amount Δ of additional pheromone is 'deposited' along all tour edges of the best ant in the iteration. Often, the elitist ant (representing the best solution found so far) is also allowed to deposit pheromone along its path. Each of these positive updates has the form $\tau_{ij} \mapsto \tau_{ij} + \Delta$ for all cities i and j connected by an edge of the respective tour. Initially $\tau_{ij} = \tau_0$ for each edge e_{ij} .

Population-Based Ant Colony Optimization (PACO): The population-based ACO (PACO), which has been proposed by Guntsch [10], is a modification of the standard ACO. The main difference is that the pheromone matrix no longer accumulates the information from all the updates over time, but instead only contains information about a small number k of solutions explicitly maintained in a population. Solution construction is performed probabilistically as in the standard ACO described above. The main change is the pheromone update, which is described in more detail in the next paragraph.

In the beginning, the pheromone matrix is initialized with a constant value τ_0 , the solution population with a maximal size of k is empty. Then, in each

of the k first iterations, the iteration's best ant is allowed to lay pheromone ($\tau_{ij} \mapsto \tau_{ij} + \Delta$) on all edges of its tour in the pheromone matrix. Furthermore, the tour is added to the solution population. No pheromone evaporates during the first k iterations. In all subsequent iterations $(k + 1), (k + 2), \dots$, the best ant updates as before and is added to the solution population. To keep the population size constant, another solution of the population (usually the worst or the oldest) is deleted, and the respective amount of pheromone is subtracted from the elements of the pheromone matrix corresponding to the deleted solution ($\tau_{ij} \mapsto \tau_{ij} - \Delta$). The information of the deleted ant completely disappears in one iteration. Consequently, the pheromone matrix only preserves information about the k ants currently in the solution population. Observe that in PACO, pheromone values never fall below the initial amount of pheromone τ_0 and never exceed $\tau_0 + k\Delta$.

The fact that the pheromone matrix used in PACO represents only a small number of solutions inspired us to design ABX which shall be described in Section 3.

2.3 Hybrids

A couple of authors have suggested to combine the ideas of ACO and EAs in several ways. Bonabeau et al. [4], for example, propose to optimize ACO parameters using an EA, and Miaghiikh and Punch [13,14] design a hybrid which uses a pheromone matrix as well as a complete solution as part of each individual's representation. To the authors' knowledge, no one has ever proposed to use an ACO algorithm to replace crossover in the way presented in this paper.

Many approaches combine metaheuristics with local search for best results [7]. But here we are interested in the workings of the specific crossover operator proposed. Since we were afraid that local search might blur the effects of crossover, we decided to concentrate on crossover alone.

3 Ant-Based Crossover

The fundamental idea of ABX is as follows: In each generation of the EA the parents are regarded as a solution population in the sense of a PACO. Their tour information is used to generate temporary pheromone matrices. These temporary pheromone matrices are then used by ants to generate new solutions. The generated set of solutions is the candidate set for the children returned to the EA. This creates a number of design options which are discussed in the following:

Number of parents: In principle, the temporary pheromone matrix can be created from an arbitrary number of parents, ranging from 1 to the population size p . We denote this parameter $\#parents$.

Pheromone matrix initialization: It is important how much influence is given to the parents relative to the basic initialization value $\tau_0 = 1/n$. We tested two basic possibilities:

- *Uniform update:* each parent deposits a pheromone value of $1/\#parents$ on each of the edges along its tour.
- *Rank-based update:* The amount of pheromone a parent is allowed to deposit depends on its rank within the set of parent individuals. The individual with rank i ($i = 1 \dots \#parents$) is allowed to deposit

$$\Delta_i = \frac{b}{\#parents} - \left(\frac{2b-2}{\#parents} \right) \left(\frac{i-1}{\#parents-1} \right)$$

with $b = 1.5$, which results in a linear weighting from best to worst.

In both cases, the total amount of pheromone in each row of the pheromone matrix is equal to 2. Half of it results from the initialization τ_0 and half of it from the parents' updates.

ACO run: Given a temporary pheromone matrix, we have to decide on the number of iterations $\#iter$ we would like to run the ACO, and the number of solutions m that are constructed in each iteration. In case we decide to run the ACO for more than one iteration, a pheromone update strategy has to be chosen as well. We used the standard evaporation strategy in combination with an elite ant for pheromone update, the update value was set to $\Delta = 1/\#parents$.

Number of children: The general scheme allows us to create any number of children from a single crossover operation, ranging from one to $m \cdot \#iter$. The number of children is henceforth denoted $\#children$, and the best $\#children$ from the $m \cdot \#iter$ generated solutions are returned as children.

4 Empirical Evaluation

For empirical evaluation, we proceed as follows: first, we try to find a reasonable set of the basic EA parameter settings. Parameters are tuned independently for an EA with ERX and an EA with ABX. Then, in a second step we will examine the effect of the parameters and design choices specific to the ant-based crossover. Finally, we will compare our approach to the standard algorithms ACO and EA with ERX on different TSP test instances.

4.1 Test Setup

For the initial parameter tuning, we use the eil101 TSP instance from TSPLIB [16] which has an optimal tour length of 629. Our basic EA uses a $(\mu + \lambda)$ -reproduction scheme¹ with tournament selection and tournament size of 2. To keep the number of free parameters small, we fix μ to 50 and only vary λ .

¹ λ children are created in every generation, and then compete with the μ individuals from the last generation's population for survival into the next generation

Mutation swaps the subtour between two randomly selected cities. The first city is selected at random, and the second city is selected in its neighborhood. More specifically, if c_1 is the position of the first city in the current tour, the second city's position is determined using a gaussian distribution with expected value of c_1 and standard deviation σ (result modulo n). The mutation operator is called with probability *mutprob*. If an individual is mutated, at least one swap is performed. Additional swaps are performed with probability *repeatSwap*, which results in a geometric distribution of the number of swaps with mean $1/(1 - \text{repeatSwap})$. All children are created by crossover, i.e. crossover probability is equal to 1.0. Specifically for ABX, parameters α and β are fixed to standard values 1 and 5 respectively.

Each algorithm terminates after a fixed number of 50,000 evaluations. Note that the EA with ERX always generates one child per crossover and performs λ evaluations per generation of the EA, i.e. the EA runs for $50,000/\lambda$ generations. With ABX, each solution generated by an ant counts as one evaluation, i.e. there are $(\lambda/\#children)(m \cdot \#iter)$ evaluations per generation of the EA, which can be significantly larger than λ . The number of EA generations is reduced accordingly. Recalculating the fitness after mutation is not counted towards the number of evaluations, since this can be done very efficiently in constant time for the given mutation operator. A comparison based on a fixed number of evaluations implicitly assumes that evaluation is much more time consuming than the crossover operation. This is true for many problems but not for a TSP. On the other hand, fixing the runtime makes the result very much dependent on implementation issues. In our experiments with up to 198 cities, the actual runtime differences between the different examined approaches were negligible. We therefore decided to use a fixed number of evaluations as stopping criterion.

In the results reported below, the performance of each parameter set is averaged over 20 runs with different random seeds. T-tests with significance level of 0.99 are used to analyze significance.

4.2 Basic EA Parameters

The basic EA parameters tuned first are the number of offspring per generation λ , the mutation probability *mutprob*, the expected length of the swapped tour σ , and the mutation frequency *repeatSwap*. With regard to ABX, for the test reported here, we use rank-based update of the parents, two parents per crossover, and a single ant producing a single child based on the temporary pheromone matrix ($\#children = 1, m = 1, \#iter = 1$).

We test all possible combinations of the parameter settings listed in Table 1. The settings that perform best for ERX are $\lambda = 50$, *mutprob* = 0.8, $\sigma = 15$ and *repeatSwap* = 0.1 which yield a solution quality of 691.8.

For the EA with ABX, $\lambda = 1$ performs slightly (but not significantly) better than $\lambda = 24$. Nevertheless, we chose $\lambda = 24$ for further testing, since $\lambda = 1$ restricts the testing of child-parent combinations too much. The effect of the mutation parameters seem to be relatively small. We select the following parameters for future tests: *mutprob* = 0.25, $\sigma = 1$ and *repeatSwap* = 0.1. It is

Table 1. Tested parameter values for reproduction and mutation, settings chosen for future tests are bold.

	ERX	ABX
λ	1, 25, 50	1, 24 , 50
<i>mutprob</i>	0.25, 0.6, 0.8 , 1.0	0.0, 0.25 , 0.5, 0.75
σ	3, 10, 15	1 , 3, 10
<i>repeatSwap</i>	0.1 , 0.4, 0.5, 0.6	0.0, 0.1 , 0.5

interesting to note that the results without mutation (*mutprob* = 0) are almost as good. The fact that mutation plays a minor role in ant-based crossover is not really surprising, because variation is introduced implicitly as part of crossover by the way ants construct their tours probabilistically.

4.3 Parameters for Ant-Based Crossover

In this section, we analyze the influence of the parameters and design choices specific to ABX. For that purpose, we test all feasible combinations of the parameters specified in Table 2. Evaporation rate ρ is set to 0.1 where needed. Additionally, we test a large number of combinations with $\#children = 8$, $\#parents = 1$, $\#parents = 50$ as well as $\#iter = 15$.

Table 2. Tested parameter values for ABX

parameter	values tested
$\#parents$	2, 4, 8
<i>parentalUpdate</i>	constant, rank-based
$\#children$	1, 2, 24
<i>m</i>	1, 2, 12, 24
$\#iter$	1, 2 or 5

Overall, the approach seems to be rather robust with respect to the parameter settings chosen. The following paragraphs outline the main results for the five examined parameters. Results with respect to a specific parameter are averaged over all settings of the other parameters (as long as they existed for all settings of the examined parameter).

Number of parents: Table 3 shows the best tour length over all performed test runs classified according to the number of parents and the parental update strategy. As can be seen, using two or four parents for crossover is better than only one or more than eight. The differences are statistically significant. Looking at the convergence graphs (not shown), it becomes apparent that increasing the number of parents slows down convergence.

Table 3. Test results depending on the number of parents and the parental update

	parental update			
	all		constant	rank-based
$\#parents$	mean	std. error	mean	mean
1	639.61	0.2234	639.61	639.61
2	636.38	0.2095	636.36	636.72
4	636.38	0.1820	636.89	636.33
8	637.68	0.2596	637.91	637.50
50	641.27	0.4559	642.60	639.94
all combinations	637.93	0.1736	638.30	637.56

Parental update: Unsurprisingly, rank-based parental update leads to faster convergence than uniform parental update, due to the additional influence of good parents (convergence curves are not shown due to space limitations). As can be seen in Table 3, the difference of the two update strategies w.r.t. the obtained tour length is rather small, but becomes more pronounced in combination with a large number of parents. As has been noted in the previous paragraph, increasing the number of parents slows down convergence. This effect should be counterbalanced to some degree e.g. by using the rank-based parental update.

Number of children per crossover: The 24 children generated per generation of the EA can be produced by calling the ABX once with $\#iter \cdot m > 24$. Alternatively, one may call the ABX several times, thereby splitting the total of 24 children to be generated evenly among the ABXs. Our test results suggest that it is significantly better to generate only a few children per crossover and rather call the ABX more than once with a smaller number of children each. In other words, it seems to be important that the children are generated based on the information from different sets of parents. The reason may be that if all 24 children are based on one temporary pheromone matrix, they might be so similar that they lead to early convergence of the EA. Overall, test runs converge slower with decreasing $\#children$, but to a better solution (cf. Figure 1). This effect is strengthened with increasing $\#iter$ (see below).

Number of ants per iteration: Increasing the number of ants m per ACO iteration implicitly leads to better children. On the other hand, the number of fitness evaluations required per generated child is increased, meaning that the EA can only run for fewer generations. Our tests show that the parameter has little influence on the final results, although convergence is slowed down a bit with increasing m . Apparently, the effect of improved children is not able to outweigh the reduction of EA generations, at least not given the limit of 50,000 evaluations (cf. Table 4). For our test environment, between two and twelve ants per iteration seem to perform best.

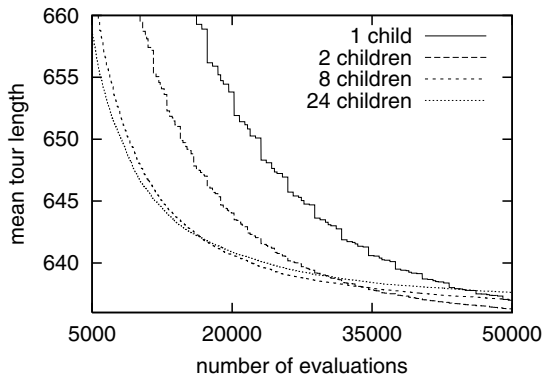


Fig. 1. Convergence behavior of runs with different numbers of children per crossover.

Number of ACO iterations: Similar to increasing the number of ants per iteration, increasing the number of iterations per ACO improves the quality of the generated children at the expense of requiring a larger number of fitness evaluations. Although the additional search should be more structured, when comparing Tables 4 and 5, little difference can be observed regarding the effect of these two parameters. According to our test results, two or five iterations of ants yield the shortest tours. These two settings are significantly better than only a single iteration (cf. Table 5).

Note that the standard error of the results for 15 iterations is relatively high. As can be seen in Figure 2, this high variance can be traced back to two different effects. First of all, in case all children of one generation are generated from a single ACO run, 15 generations lead to premature convergence after only 15,000 – 20,000 evaluations and very poor results. The effect of many children generated from a single temporary pheromone matrix, as has been described above, is emphasized by running many ACO iterations, since the pheromone matrix converges and thus the children become even more similar. If few children are generated, two cases can be distinguished: If m is large, the number of evaluations per child becomes so high that the runs are far from convergence given the maximum of 50,000 evaluations, and consequently the results are rather poor. On the opposite, the algorithm converges and the results are very good if m

Table 4. Test results depending on the number of ants per ACO iteration

m	mean	std. error
1	636.85	0.3200
2	636.33	0.2619
12	636.14	0.2816
24	637.79	0.4289

Table 5. Test results depending on the number of ACO iterations

#iter	mean	std. error
1	637.27	0.2458
2	636.54	0.2292
5	636.62	0.3122
15	637.53	0.6761

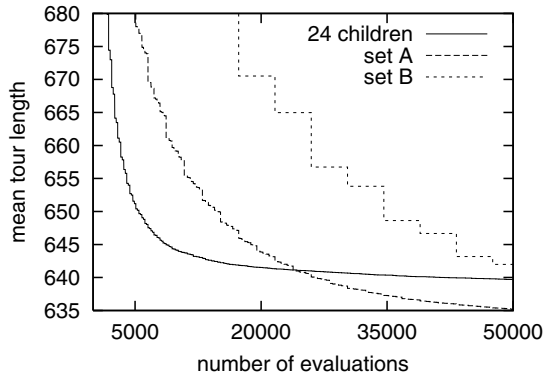


Fig. 2. Convergence behavior of runs with 15 generations of ants. The first line has 24 children per ABX. Sets A and B are averages over runs with ≤ 12 children per operator, set A over those with less than 4000 evaluations per crossover, set B over those with more than 4000 evaluations per crossover.

is sufficiently small. On the whole, increasing the number of ACO iterations leads to promising solutions given that the algorithm has sufficient time to converge and the number of children per population is small.

Summary: To sum up, the EA with ABX is quite robust with respect to the examined parameter settings. As is often the case, the ideal parameter settings probably depend on the time available for computation. We have demonstrated that the number of evaluations per crossover operator ($m \cdot \#iter$) plays an important role. If this number is too large, the algorithm will not converge in the given time frame. Apparently, in most cases the effect of local optimization due to the larger number of tours evaluated cannot outweigh the reduction of generations performed by the EA. This stresses the importance of the EA heuristic and clarifies that ABX avails itself of both algorithms and is more than a splitted ACO. For the tests reported in the next section, we use two parents per ABX with uniform update and allow 12 ants to run for 5 iterations to produce one child.

4.4 Comparison of ABX with ERX, and ACO

To compare the performance of our ABX with the other heuristics, we carry out test runs on the following three benchmark problems from the TSPlib [19]: *eil101* with 50,000 evaluations, *kroA150* with 75,000 evaluations and *d198* with 100,000 evaluations (linearly increasing the maximum allowed number of evaluations with the number of cities in the problem). Since in practice, it is not possible to perform extensive parameter tuning when solving a new problem instance, for all heuristics we use the same parameter settings that have proven successful for *eil101* respectively. The results are summarized in Table 6.

Table 6. Comparison of the ant-based crossover with other approaches

Heuristic	Problem Instance		
	eil101	kroA150	d198
ERX	691.8	32985.85	18671.8
Standard ACO	638.5	27090.76	16123.36
Ant-Based Crossover	632.5	26807.8	16080.8
Optimum	629	26524	15780

As can be seen, our EA with ABX clearly outperforms the EA with ERX in all tested problem instances. It also performs significantly better than pure ACO². In addition, we can compare ABX to the relatively similar weight-biased edge-crossover reported in [11]. For the tested kroA150 problem, Julstrom and Raidl report an average result of 27081 for their best strategy after 150,000 evaluations, which is clearly inferior to our result of 26807.8 after 75,000 evaluations (at least when ignoring other factors influencing computational complexity).

5 Conclusion and Future Work

In this paper we introduced a new crossover operator for permutation problems which draws on ideas from ant colony optimization (ACO). With the suggested ant-based crossover (ABX), it is straightforward to integrate problem-specific heuristic knowledge and local fine-tuning into the crossover operation. First empirical tests on the TSP have shown that the approach is rather robust with respect to parameter settings, and that it significantly outperforms an EA with edge recombination crossover, as well as pure ACO.

Given these excellent results, the performance of the ABX should also be tested on other permutation problems such as scheduling or the quadratic assignment problem. A more thorough comparison of the computational complexities of the different approaches would also be desirable. Finally, for best results, a hybridization of our approach with local optimizers like Lin-Kernighan should be tested.

References

1. J. C. Bean. Genetic algorithms and random keys for sequencing and optimization. *ORSA Journal on Computing*, 6(2):154–160, 1994.

2. C. Bierwirth, D.C. Mattfeld, and H. Kopfer. On permutation representations for scheduling problems. In H.-M. Voigt, editor, *Parallel Problem Solving from Nature*, volume 1141 of *LNCS*, pages 310–318. Springer, Berlin, 1996.

3. E. Bonabeau, M. Dorigo, and G. Theraulaz. *Swarm intelligence: from natural to artificial systems*. Oxford University Press, 1999.

² $m = 15, \alpha = 1, \beta = 5, \rho = 0.01, \tau_0 = 0.5$, fix update of $\Delta = 0.05$ for best ant of iteration and elite ant, and minimal pheromone value of $\tau_{min} = 0.001$.

4. H. M. Botee and E. Bonabeau. Evolving ant colonies. *Advanced Complex Systems*, 1:149–159, 1998.
5. L. Davis. Applying adaptive algorithms to epistatic domains. In *International Joint Conference on Artificial Intelligence*, pages 162–164, 1985.
6. M. Dorigo and G. Di Caro. The ant colony optimization meta-heuristic. In D. Corne, M. Dorigo, and F. Glover, editors, *New Ideas in Optimization*, pages 11–32. McGraw-Hill, 1999.
7. B. Freisleben and P. Merz. New genetic local search operators for the traveling salesman problem. In Hans-Michael Voigt, Werner Ebeling, Ingo Rechenberg, and Hans-Paul Schwefel, editors, *Parallel Problem Solving from Nature*, volume 1141, pages 890–899, Berlin, 1996. Springer.
8. D. E. Goldberg and R. Lingle. Alleles, loci, and the TSP. In J. J. Grefenstette, editor, *First International Conference on Genetic Algorithms*, pages 154–159. Lawrence Erlbaum Associates, 1985.
9. J. J. Grefenstette. Incorporating problem specific knowledge into genetic algorithms. In *Genetic Algorithms and Simulated Annealing*, pages 42–60. Morgan Kaufmann, 1987.
10. M. Guntsch and M. Middendorf. A population based approach for ACO. In *European Workshop on Evolutionary Computation in Combinatorial Optimization*, volume 2279 of *LNCS*, pages 72–81. Springer, 2002.
11. B. A. Julstrom and G. R. Raidl. Weight-biased edge-crossover in evolutionary algorithms for two graph problems. In G. Lamont, J. Carroll, H. Haddad, D. Morton, G. Papadopoulos, R. Sincovec, and A. Yfantis, editors, *16th ACM Symposium on Applied Computing*, pages 321–326. ACM Press, 2001.
12. S. Jung and B.-R. Moon. Toward minimal restriction of genetic encoding and crossovers for the two-dimensional Euclidean TSP. *IEEE Transactions on Evolutionary Computation*, 6(6):557–565, 2002.
13. V. V. Miagkikh and W. F. Punch. An approach to solving combinatorial optimization problems using a population of reinforcement learning agents. In *Genetic and Evolutionary Computation Conference*, pages 1358–1365, 1999.
14. V. V. Miagkikh and W. F. Punch. A generalized approach to handling parameter interdependencies in probabilistic modeling and reinforcement learning optimization algorithms. In *Workshop on Frontiers in Evolutionary Algorithms*, 2000.
15. Y. Nagata and S. Kobayashi. Edge assembly crossover: A high-power genetic algorithm for the traveling salesman problem. In T. Bäck, editor, *International Conference on Genetic Algorithms*, pages 450–457. Morgan Kaufmann, 1997.
16. G. Reinelt. TSPLIB - a travelling salesman problem library. *ORSA Journal on Computing*, 3:376–384, 1991.
17. A.Y.-C. Tang and K.-S. Leung. A modified edge recombination operator for the travelling salesman problem. In *Parallel Problem Solving from Nature II*, volume 866 of *LNCS*, pages 180–188, Berlin, 1994. Springer.
18. G. Tao and Z. Michalewicz. Evolutionary algorithms for the TSP. In A. E. Eiben, T. Bäck, M. Schoenauer, and H.-P. Schwefel, editors, *Parallel Problem Solving from Nature*, volume 1498 of *LNCS*, pages 803–812. Springer, 1998.
19. <http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/index.html>.
20. D. Whitley, T. Starkweather, and D'A. Fuquay. Scheduling problems and traveling salesman: The genetic edge recombination operator. In J. Schaffer, editor, *International Conference on Genetic Algorithms*, pages 133–140. Morgan Kaufmann, 1989.

Selection in the Presence of Noise

Jürgen Branke and Christian Schmidt

Institute AIFB, University of Karlsruhe, 76128 Karlsruhe, Germany
{branke|csc}@aifb.uni-karlsruhe.de

Abstract. For noisy optimization problems, there is generally a trade-off between the effort spent to reduce the noise (in order to allow the optimization algorithm to run properly), and the number of solutions evaluated during optimization. However, for stochastic search algorithms like evolutionary optimization, noise is not always a bad thing. On the contrary, in many cases, noise has a very similar effect to the randomness which is purposefully and deliberately introduced e.g. during selection. Using the example of stochastic tournament selection, we show that the noise inherent in the optimization problem should be taken into account by the selection operator, and that one should not reduce noise further than necessary.

Keywords: Noise, tournament selection, stochastic fitness

1 Introduction

Many real-world optimization problems are noisy, i.e. a solution's quality (and thus the fitness function) is a random variable. Examples include all applications where the fitness is determined by a stochastic computer simulation, or where fitness is measured physically and prone to measuring error. Researchers have long argued that evolutionary algorithms (EAs) should be relatively robust against noise (see e.g. [FG88]), and recently a number of publications have appeared which support that claim at least partially [MG96, AB00a, AB00b, AB03].

For most noisy optimization problems, the uncertainty in fitness evaluation can be reduced by sampling an individual's fitness several times and using the average as estimate for the true mean fitness. Sampling n times reduces a random variable's standard deviation by a factor of $\sqrt{n}$, but on the other hand increases the computation time by a factor of n . Thus, there is a generally perceived trade-off: either one can use relatively exact estimations but only evaluate a small number of individuals (because a single estimation requires many evaluations), or one can let the algorithm work with relatively crude fitness estimations, but allow for more evaluations (as each estimation requires less effort). Generally, noise is considered harmful, as it may mislead the optimization algorithm. The main issue is probably the selection step: If due to the noise, a bad individual is evaluated better than it actually is, and/or a good individual is evaluated worse than its true fitness, the EA may wrongly select the worse individual although (according to the algorithmic design) it should have selected the better individual. Clearly, if such errors happen too frequently, optimization stagnates.

However, noise is not always a bad thing, on the contrary. EAs are randomized search algorithms, which use deliberate randomness to purposefully introduce errors into the selection process, primarily in order to get out of local minima. Therefore, in this paper we argue that it should be possible to accept the noise inherent in the optimization problem and to use it to (at least partially) replace the randomness in the optimization algorithm. As a result, it is possible to get the optimization algorithm to behave closer to its behavior on deterministic problems, even without excessive sampling. Furthermore, we will demonstrate that, depending on the fitness values and variances, noise affects some tournaments much stronger than others. As a consequence, we suggest a simple but effective resampling strategy to adapt the sample size to the specific tournament, allowing us to again get closer to the algorithm's behavior in a deterministic setting, while drastically reducing the number of samples required.

The paper is structured as follows: In Section 2, we survey some related work on EAs applied to noisy optimization problems, followed by a brief description of stochastic tournament selection. Section 4 demonstrates the effect noise has on tournament selection, and describes two ways to integrate a possible sampling error into the selection procedure. The idea of adapting not only the selection probability but also the sample size is discussed in Section 5. The paper concludes with a summary and some ideas for future work.

2 Related Work

The application of EAs in noisy environments has been the focus of many research papers. There are several papers that have looked at the trade-off between population size and sample size to estimate an individual's fitness, with sometimes conflicting results. Fitzpatrick and Grefenstette [FG88] conclude that for the genetic algorithm studied, it is better to increase the population size than the sample size. On the other hand, Beyer [Bey93] shows that for a $(1, \lambda)$ evolution strategy on a simple sphere, one should increase the sample size rather than λ . Hammel and Bäck [HB94] confirm these results and empirically show that it also doesn't help to increase the parent population size μ . Finally, Arnold and Beyer [AB00a, AB00b] show analytically that for the simple sphere, increasing the parent population size μ is helpful in combination with intermediate multirecombination. Miller [Mil97, MG96] has developed some simplified theoretical models which allow to simultaneously optimize the population size and the sample size. A good overview of theoretical work on EAs applied to noisy optimization problems can be found in [Bey00] or [Arn02].

All papers mentioned so far assume that the sample size is fixed for all individuals. Aizawa and Wah [AW94] were probably the first to suggest that the sample size could be adapted during the run, and suggested two adaptation schemes: increasing with the generation number, and higher sample size for individuals with higher estimated variance. Albert and Goldberg [AG01] look at a slightly different problem, but also conclude that the sample size should increase over the run. For (μ, λ) or $(\mu + \lambda)$ selection, Stagge [Sta98] has suggested basing

the sample size on an individual's probability to be among the μ best (and thus to survive to the next generation).

Branke et al. [Bra98,BSS01] and Sano and Kita [SK00,SKKY00] propose taking the fitness estimations of neighboring individuals into account when estimating an individual's fitness. This improves the estimation without requiring additional samples.

Finally, another related subject is that of searching for robust solutions, where instead of a noisy fitness function the decision variables are perturbed (cf. [TG97, Bra98,Bra01]).

3 Stochastic Tournament Selection

Stochastic tournament selection (STS) [GD91] is a rather simple selection scheme where two individuals are randomly chosen from the population, and then the better is selected with probability $(1 - \gamma)$. If individuals are sorted from rank 1 (best) to rank m (worst), this results in a linearly decreasing selection probability for an individual on rank i , with the slope of the line being determined by the selection probability $(1 - \gamma)$.

4 Selection Based on a Fixed Sample Size

Selecting the better of two individuals with probability $(1 - \gamma)$ in a noisy environment can be achieved in two fundamental ways: The standard way would be to eliminate the noise as much as possible by using a large number of samples, and then selecting the better individual with probability $(1 - \gamma)$. The noise-adapted selection proposed here has a different philosophy: instead of eliminating the noise and then artificially introducing randomness, we propose accepting a higher level of noise, and only add a little bit of randomness to achieve the desired behavior.

In the following, we will start with the standard STS, demonstrate the consequences in a noisy environment, and then develop a simple and a more complex model to get closer to the ideal noise-adapted selection.

4.1 Basic Notations

Let us denote the two individuals to be compared as x and y . If the fitness is noisy, the fitness of individual x (y) is a random variable F_x (F_y) with $F_x \sim \mathcal{N}(\mu_x, \sigma_x^2)$ ($F_y \sim \mathcal{N}(\mu_y, \sigma_y^2)$)¹. If $\mu_x > \mu_y$, we would like to select individual x with probability $(1 - \gamma)$ and vice versa. However, μ_x and μ_y are unknown, we can only estimate them by sampling each individual's fitness a number of n times

¹ Note that it will be sufficient to assume that the average difference obtained from sampling the individuals' fitnesses n times is normally distributed. This is certainly valid if each individual's fitness is normally distributed, but also independent of the actual fitness distributions for large enough n (central limit theorem).

and using the averages $\bar{f}_x$ and $\bar{f}_y$ as estimators for the fitnesses, and the sample variances s_x^2 and s_y^2 as estimators for the true variances.

If the actual fitness difference between the two individuals is denoted as $\delta = \mu_x - \mu_y$, the observed fitness difference $D = \bar{f}_x - \bar{f}_y$ is again a random variable $D \sim \mathcal{N}(\delta, \sigma_d^2)$. The variance of D depends on the number of samples drawn from each individual, n , and can be calculated as $\sigma_d^2 = (\sigma_x^2 + \sigma_y^2)/n$. A specific realization of the observed fitness difference is named d . Furthermore, we will need a standardized observed fitness which we define as $d^* = d/\sqrt{s_d^2}$ where $s_d^2 = (s_x^2 + s_y^2)/n$ is the unbiased estimated standard deviation of the fitness difference. The corresponding true counterpart is $\delta^* = \delta/\sigma_d$. Note that non-linear transformations of unbiased estimators are no longer unbiased, therefore d^* is a biased estimator for δ^* .

While γ is the *desired* selection probability for the *truly* worse individual, we denote with β the *implemented* probability for choosing the worse individual based on the estimated standardized fitness difference d^* , and $\xi(\delta^*, \beta)$ the actual selection probability for the *better* individual given a true standardized fitness difference of δ^* .

4.2 Standard Stochastic Tournament Selection

The simplest (and standard) way to apply STS would be to ignore the uncertainty in evaluation by making the following assumption:

Assumption: The observed fitness difference is equal to the actual fitness difference, i.e. $d = \delta$.

As a consequence, individual x is selected with probability $(1 - \beta) = (1 - \gamma)$ if $d \geq 0$ and with probability $\beta = \gamma$ if $d < 0$.

However, there can be two sources of error: Either we observe a fitness difference $d > 0$ when actually $\delta < 0$, or vice versa. The corresponding error probability α can be calculated as

$$\begin{aligned} \alpha &= \begin{cases} P(D > 0) = 1 - \Phi\left(\frac{-\delta}{\sigma_d}\right) = \Phi\left(\frac{\delta}{\sigma_d}\right) & : \delta \leq 0 \\ P(D < 0) = \Phi\left(\frac{-\delta}{\sigma_d}\right) & : \delta > 0 \end{cases} \\ &= \Phi\left(\frac{-|\delta|}{\sigma_d}\right) = \Phi(-|\delta^*|) \end{aligned} \quad (1)$$

with Φ being the cumulative distribution function for a standard gaussian.

The overall selection probability for individual x can then be calculated as

$$\begin{aligned} \xi &= P(D > 0)(1 - \beta) + P(D < 0)\beta \\ &= (1 - \alpha)(1 - \beta) + \alpha\beta \end{aligned} \quad (2)$$

Example: To visualize the effect of the error probability on the actual selection probability ξ , let us consider an example with $\sigma_x^2 = \sigma_y^2 = 10$, $n = 20$ and $\gamma = 0.2$. The actual selection probability for individual x depending on δ^* can be determined by a Monte Carlo simulation. We did this in the following way: For

a given δ^* , we generated 100,000 realizations of d^* according to $d^* = \frac{\bar{f}_x - \bar{f}_y}{\sqrt{(s_x^2 + s_y^2)/n}}$ based on $F_x \sim \mathcal{N}(0, \sigma_x^2)$, $F_y \sim \mathcal{N}(-\delta^* \sigma_d, \sigma_y^2)$. For each observed d^* , we select x with probability $(1 - \beta)$ if $d^* > 0$ and with probability β otherwise. The actual selection probability $\xi(\delta^*, \beta)$ is then the fraction of times x has been selected.

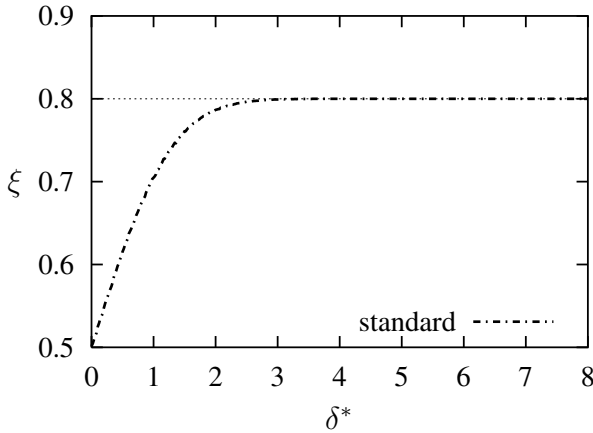


Fig. 1. True selection probability of individual x depending on the actual standardized fitness difference δ^* . The dotted line represents the desired selection probability $(1 - \gamma)$.

Figure 1 depicts the resulting true selection probability of individual x depending on the actual standardized fitness difference δ^* . The dotted line corresponds to the desired behavior in the deterministic case, the bold line labeled “standard” is the actual selection probability due to the noise. As can be seen, the actual selection probability for the better individual largely depends on the ratio δ^* of the fitness difference δ and the amount of noise measured as σ_d . While it corresponds to the desired selection probability of $(1 - \gamma)$ for $\delta^* > 3$, it approaches 0.5 for $\delta^* \rightarrow 0$. The latter fact is unavoidable, since for $\delta^* \rightarrow 0$ it becomes basically impossible to determine the better of the two individuals. The interesting question is how quickly ξ approaches $1 - \gamma$, and whether this behavior can be improved. Note that we only show the curves for $\delta^* \geq 0$ (assuming without loss of generality that $\mu_x > \mu_y$). For $\delta^* < 0$ the curve would be symmetric to $(0, 0.5)$.

In previous papers, it has been noted that the effect of noise on EAs is similar to a smaller selection pressure (e.g. [Mil97]). Figure 1 demonstrates that this is not entirely true for STS. A lower selection pressure in form of a higher γ would change the level of the dotted line, but it would still be horizontal, i.e. the selection probability for the better individual would be independent of the actual fitness difference. With noise, only the tournaments between individuals

of similar fitness are affected. Hence, a dependence on the actual fitness values is introduced which somehow contradicts the idea of rank-based selection.

4.3 A Simple Correction

If we know that our conclusion about which of the two individuals has a better fitness is prone to some error, it seems straightforward to take this error probability into account when deciding which individual to select. Instead of always selecting the better individual with probability $(1 - \gamma)$, we could try to replace γ by a function $\beta(d^*)$ which depends on the standardized observed difference d^* .

Let us make the following assumption:

Assumption: It is possible to accurately estimate the error probability α .

Then, since we would like to have an overall true selection probability of $(1 - \gamma)$, an appropriate β -function could be derived as

$$(1 - \alpha)(1 - \beta) + \alpha\beta \stackrel{!}{=} (1 - \gamma) \quad (3)$$

$$1 - \beta - \alpha + \alpha\beta + \alpha\beta = (1 - \gamma)$$

$$\beta(-1 + 2\alpha) = (1 - \gamma) - 1 + \alpha$$

$$\beta = \frac{\gamma - \alpha}{1 - 2\alpha}. \quad (4)$$

β is a probability and can not be smaller than 0, i.e. the above equation assumes $\alpha \leq \gamma < 0.5$. For $\alpha > \gamma$ we set $\beta = 0$.

Unfortunately, α can not be calculated using Equation 1, because we don't know either δ nor σ_d .

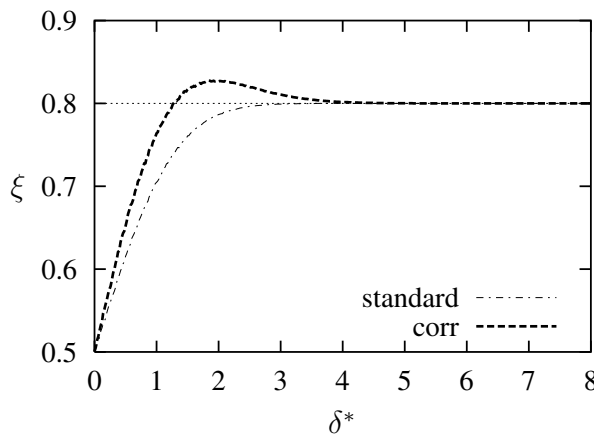


Fig. 2. True selection probability of individual x depending on the actual standardized fitness difference δ^* . The dotted line represents the desired selection probability $(1 - \gamma)$.

It seems straightforward then to estimate δ by the observed difference d , and σ_d^2 by the observed variance s_d^2 . Then, α is estimated as $\hat{\alpha} = \Phi(-|d|/s_d) = \Phi(-|d^*|)$, which is only a biased estimator due to the non-linear transformations. Nevertheless, this may serve as a reasonable first approximation of an optimal β -function. Figure 3 visualizes this β -function (labeled as “corr”). As can be seen, the probability to select the worse individuals decreases when the standardized difference d^* becomes small, and is 0 for $|d^*| < -\Phi^{-1}(\gamma)$ (i.e. the observed better individual is always selected if the observed standardized fitness difference d^* is small). Assuming the same parameters as in the example above, the resulting true selection probabilities $\xi(\delta^*, \beta(\cdot))$ are depicted in Figure 2 (labeled as “corr”). The true selection probability approaches the desired selection probability faster than with the standard approach, but then it overshoots before it converges towards $(1 - \gamma)$. Nevertheless, the approximation is already much better than the standard approach (assuming a uniform distribution of δ^*).

4.4 Bootstrapping

The β -function proposed above can be further improved by bootstrapping [Efr90]. This method compares the observed selection probabilities p given the current β function with the desired selection probabilities, and then reduces β where the selection probability is too low, and increases β where the selection probability is too high. The observed selection probabilities $\xi(\delta^*, \beta(\cdot))$ have to be estimated by Monte Carlo simulation, generating realisations of d^* and then selecting according to $\beta(d^*)$. Unfortunately, the distribution of d^* depends on the variance σ_d^2 of the observed fitness difference which is unknown. Therefore, in this approach we make the following simplifying assumption:

Assumption: The estimated variance of the difference corresponds to the true variance of the difference, i.e. $s_d^2 = \sigma_d^2$.

From that it follows that d^* is normally distributed according to $\mathcal{N}(\delta^*, 1)$.

More specifically, our bootstrapping approach starts with an initial $\beta_0(z)$ which corresponds to the β function defined in the section above. Then, it iteratively adapts beta according to

$$\beta_{t+1}(z) = \beta_t(z) + \xi(z, \beta_t(\cdot)) - (1 - \gamma). \quad (5)$$

This procedure can be iterated until one is satisfied with the outcome.

The resulting β -function is depicted in Figure 3. At first sight, the strong fluctuations seems surprising. However, a steeper ascent of the true selection probability can only be achieved by keeping $\beta(d^*) = 0$ for as long as possible. The resulting overshoot then has to be compensated by a very high β etc. such that in the end, an oscillating acceptance pattern emerges as optimal. The corresponding true selection probabilities $\xi(\delta^*)$ are shown in Figure 4. As can be seen, despite the oscillating β -function, this curve is very smooth, and much closer to the actually desired selection probability of γ resp. $(1 - \gamma)$ than either the standard approach of ignoring the noise, or the first approximation of an appropriate β -function presented in the previous section.

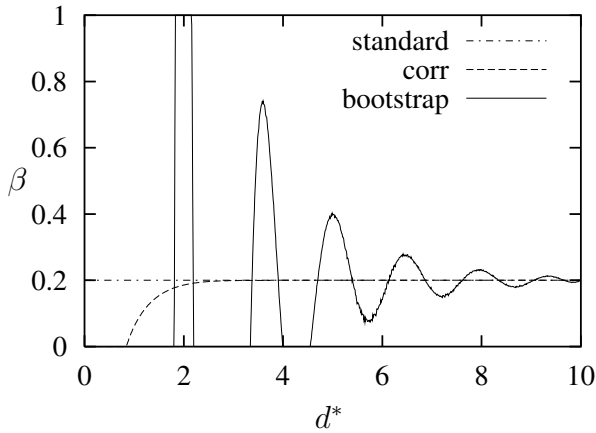


Fig. 3. The probability to select the worse individual (β -function), depending on the observed standardized fitness difference d^* . Results of the different approaches.

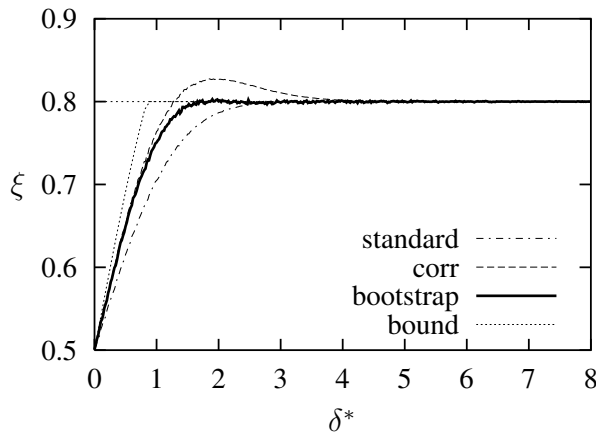


Fig. 4. True selection probability of individual x depending on the actual standardized fitness difference δ^* . The line denoted by “bound” is an idealized curve which depicts a limit to how close one can get to the desired selection probability. The dotted line represents the desired selection probability $(1 - \gamma)$.

Even though the bootstrapping method yields a much better approximation to the desired selection probability than the other two approaches, it could perhaps be further improved by basing it not only on d^* but on all three observed variables, namely d , σ_x^2 , and σ_y^2 . However, we expect that the additional improvement would be rather small. Furthermore, there is a bound to how close one can get to the desired selection probability: the steepest possible ascent of the true selection probability is clearly obtained if the individual with the higher

observed fitness is *always* selected. However, as long as α exceeds γ , the resulting true selection probability would still be below the desired selection probability. The corresponding steepest ascent curve is also shown in Figure 4 and denoted as “bound”.

Instead of trying to further improve the estimation, we will now turn to the idea of drawing additional samples if the probability for a selection error is high.

5 Resampling

From the above discussion, it is clear that the deviation from actual selection probability to desired selection probability is only severe for small values of δ/σ_d , i.e. if the individuals have similar fitness and/or the noise is large.

Therefore, we now attempt to counteract that problem by adapting the number of samples to the expected error probability, i.e. by drawing a large number of samples whenever we assume that the selection error would be high and vice versa. We propose to do that in the following way: Starting with a reduced number of 10 samples for every individual, we calculate d^* . If $|d^*| \geq \epsilon$ where ϵ is a constant, we stop and use d^* to decide which individual to select. Otherwise, we repeatedly draw another sample for each of the two individuals until either $|d^*| \geq \epsilon$ or the total number of samples exceeds a maximum number N . For our experiments, we set $N = 100$ and $\epsilon = 1.33$, which approximately yields an error probability of 1% if $\delta^* = 1$ assuming that d^* is normally distributed as $d^* \sim \mathcal{N}(\delta^*, 1)$, i.e. if $\delta^* = 1$, there is only a 1% chance that we will observe a distance $d < 0$.

For our standard example with $\sigma_x^2 = \sigma_y^2 = 10$ and $\gamma = 0.2$, the above sampling scheme results in an average number of samples depending on δ^* as depicted in Figure 5. For small standardized distances d^* , the average number of samples is quite high, but it drops quickly and approaches the lower limit of 20 for $\delta^* > 3$. Depending on the distribution of δ^* in a real EA, this sampling scheme is thus able to achieve tremendous savings compared to the fixed sampling rate of 20 samples per individual (40 samples in total). Furthermore, the actual selection probabilities using this sampling scheme are much closer to the desired selection probability than if a fixed number of samples is used. The two sampling schemes in combination with standard STS are compared in Figure 6.

Just as for the fixed sample size, we can apply bootstrapping also to the adaptive sampling scheme. The resulting β -function and selection probabilities are depicted in Figures 7 and 8. The resulting beta-function is much smoother than the one obtained for the fixed sampling scheme. Also, although there is still a clear benefit of bootstrapping with respect to the deviation of ξ from the desired $(1 - \gamma)$, the improvement over standard STS is significantly smaller than with a fixed sample size. This is probably because due to the smaller initial sample size in combination with the resampling scheme used, our assumption that D^* is normally distributed may be less appropriate.

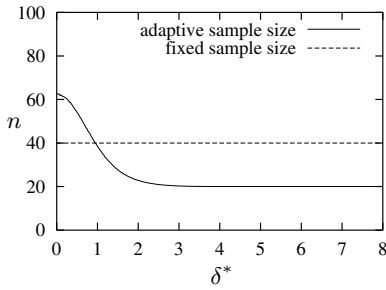


Fig. 5. Average sample size depending on the actual standardized fitness difference δ^* , with the fixed sampling scheme (dashed line) and the adaptive sampling scheme (solid line).

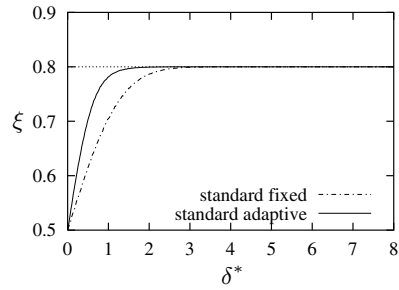


Fig. 6. Actual sampling probability depending on the actual standardized fitness difference δ^* , for the standard stochastic tournament selection with fixed and with adaptive sampling scheme.

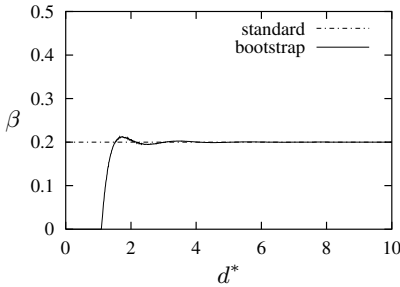


Fig. 7. β -function derived by bootstrapping for the case of an adaptive sample size.

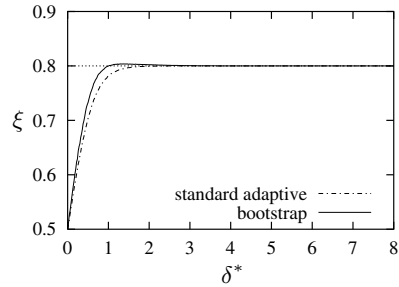


Fig. 8. Comparison of the actual sampling probability depending on the actual standardized fitness difference δ^* for the standard STS and the bootstrapping approach, when using the adaptive sampling scheme.

6 Conclusion

In this paper, we have argued that the error probability due to a noisy fitness function should be taken into account in the selection step. At the example of stochastic tournament selection, we have demonstrated that it is possible to obtain a much better match between actual and desired selection probability for an individual.

In a first step, we have derived two models which determine the selection probability for the better individual depending on the observed fitness difference. The simple model was based on some simplifying assumptions regarding

the distribution of the error probability; the second model was based on bootstrapping.

In a second step, we looked at a different sampling scheme, namely adapting the number of samples to the expected error probability. That way, a pair of similar individuals is sampled much more often than a pair of individuals with very different fitness values. This approach also greatly improves the accuracy of the actual selection probability. Additionally, depending on the distribution of fitness differences in an actual EA run, it will significantly reduce the number of samples required.

We are currently exploring a number of different extensions. For one, it should be relatively straightforward to extend our framework to other selection schemes and even to other heuristics like simulated annealing. Furthermore, we intend to improve the adaptive sampling scheme by using statistical test theory.

Acknowledgements. We would like to thank David Jones for pointing us to the bootstrapping methodology, and the anonymous reviewers for their helpful comments.

References

- [AB00a] D. V. Arnold and H.-G. Beyer. Efficiency and mutation strength adaptation of the $(\mu/\mu_i, \lambda)$ -es in a noisy environment. In Schoenauer et al. [SDR⁺00], pages 39–48.
- [AB00b] D. V. Arnold and H.-G. Beyer. Local performance of the $(\mu/\mu_i, \lambda)$ -es in a noisy environment. In W. Martin and W. Spears, editors, *Foundations of Genetic Algorithms*, pages 127–142. Morgan Kaufmann, 2000.
- [AB03] D. V. Arnold and H.-G. Beyer. A comparison of evolution strategies with other direct search methods in the presence of noise. *Computational Optimization and Applications*, 24:135–159, 2003.
- [AG01] L. A. Albert and D. E. Goldberg. Efficient evaluation genetic algorithms under integrated fitness functions. Technical Report 2001024, Illinois Genetic Algorithms Laboratory, Urbana-Champaign, USA, 2001.
- [Arn02] D. V. Arnold. *Noisy Optimization with Evolution Strategies*. Kluwer, 2002.
- [AW94] A. N. Aizawa and B. W. Wah. Scheduling of genetic algorithms in a noisy environment. *Evolutionary Computation*, pages 97–122, 1994.
- [Bey93] H.-G. Beyer. Toward a theory of evolution strategies: Some asymptotical results from the $(1 \nmid \lambda)$ -theory. *Evolutionary Computation*, 1(2):165–188, 1993.
- [Bey00] H.-G. Beyer. Evolutionary algorithms in noisy environments: Theoretical issues and guidelines for practice. *Computer methods in applied mechanics and engineering*, 186:239–267, 2000.
- [Bra98] J. Branke. Creating robust solutions by means of an evolutionary algorithm. In A. E. Eiben, T. Bäck, M. Schoenauer, and H.-P. Schwefel, editors, *Parallel Problem Solving from Nature*, volume 1498 of *LNCIS*, pages 119–128. Springer, 1998.
- [Bra01] J. Branke. *Evolutionary Optimization in Dynamic Environments*. Kluwer, 2001.

- [BSS01] J. Branke, C. Schmidt, and H. Schmeck. Efficient fitness estimation in noisy environments. In L. Spector, E. D. Goodman, A. Wu, W. B. Langdon, H.-M. Voigt, M. Gen, S. Sen, M. Dorigo, S. Pezeshk, M. H. Garzon, and E. Burke, editors, *Genetic and Evolutionary Computation Conference*, pages 243–250. Morgan Kaufmann, 2001.
- [Efr90] B. Efron. *The Jackknife, the Bootstrap and Other Resampling Plans*. SIAM, 1990.
- [FG88] J. M. Fitzpatrick and J. J. Grefenstette. Genetic algorithms in noisy environments. *Machine Learning*, 3:101–120, 1988.
- [GD91] D. E. Goldberg and K. Deb. A comparative analysis of selection schemes used in genetic algorithms. In G. Rawlins, editor, *Foundations of Genetic Algorithms*, San Mateo, CA, USA, 1991. Morgan Kaufmann.
- [HB94] U. Hammel and T. Bäck. Evolution strategies on noisy functions, how to improve convergence properties. In Y. Davidor, H. P. Schwefel, and R. Männer, editors, *Parallel Problem Solving from Nature*, volume 866 of *LNCS*. Springer, 1994.
- [MG96] B. L. Miller and D. E. Goldberg. Genetic algorithms, selection schemes, and the varying effects of noise. *Evolutionary Computation*, 4(2):113–131, 1996.
- [Mil97] Brad L. Miller. *Noise, Sampling, and Efficient Genetic Algorithms*. PhD thesis, Dept. of Computer Science, University of Illinois at Urbana-Champaign, 1997. available as TR 97001.
- [SDR⁺00] M. Schoenauer, K. Deb, G. Rudolph, X. Yao, E. Lutton, J. J. Merelo, and H.-P. Schwefel, editors. *Parallel Problem Solving from Nature*, volume 1917 of *LNCS*. Springer, 2000.
- [SK00] Y. Sano and H. Kita. Optimization of noisy fitness functions by means of genetic algorithms using history of search. In Schoenauer et al. [SDR⁺00], pages 571–580.
- [SKKY00] Y. Sano, H. Kita, I. Kamihira, and M. Yamaguchi. Online optimization of an engine controller by means of a genetic algorithm using history of search. In *Asia-Pacific Conference on Simulated Evolution and Learning*. Springer, 2000.
- [Sta98] P. Stagge. Averaging efficiently in the presence of noise. In A. E. Eiben, T. Bäck, M. Schoenauer, and H.-P. Schwefel, editors, *Parallel Problem Solving from Nature V*, volume 1498 of *LNCS*, pages 188–197. Springer, 1998.
- [TG97] S. Tsutsui and A. Ghosh. Genetic algorithms with a robust solution searching scheme. *IEEE Transactions on Evolutionary Computation*, 1(3):201–208, 1997.

Effective Use of Directional Information in Multi-objective Evolutionary Computation

Martin Brown¹ and Robert E. Smith²

¹Department of Computing and Mathematics
Manchester Metropolitan University, Manchester, UK,
m.brown@mmu.ac.uk

²The Intelligent Computer Systems Centre
The University of The West of England, Bristol, UK.
robert.smith@uwe.ac.uk

Abstract. While genetically inspired approaches to multi-objective optimization have many advantages over conventional approaches, they do not explicitly exploit directional/gradient information. This paper describes how steepest-descent, multi-objective optimization theory can be combined with EC concepts to produce improved algorithms. It shows how approximate directional information can be efficiently extracted from parent individuals, and how a multi-objective gradient can be calculated, such that children individuals can be placed in appropriate, **dominating** search directions. The paper describes and introduces the basic theoretical concepts as well as demonstrating some of the concepts on a simple test problem.

1 Introduction

Multi-objective optimization is a challenging problem in many disciplines, from product design to planning [2][3][7][9][10]. Evolutionary computation (EC) approaches to multi-objective problems have had many successes in recent years. In the realm of real-valued, single-objective optimization, recent results with EC algorithms that more explicitly exploit gradient information have shown distinct performance advantages [4]. However, as will be shown in this paper, the rationale employed in these EC algorithms must be adjusted for multi-objective EC. This paper provides a theoretical framework and some empirical evidence for these adjustments.

This paper describes how evolutionary multi-objective optimization can efficiently utilize approximate, local directional (gradient) information. The local gradients associated with each point in the population can be combined to produce a **multi-objective gradient** (MOG). The MOG indicates whether the design is locally Pareto optimal, or if the design can be improved further by altering the parameters along the direction defined by the negative MOG.

The main problem associated with the conventional approach to steepest-descent optimization is the need to estimate the local gradient for each design at each iteration. Therefore, viewing the problem from an EC perspective (where a population of de-

signs is maintained at every iteration) allows the directional information to be obtained from neighboring samples (mates), thus lowering the number of design evaluations that must be performed. This paper presents theory on how this information should be used. In describing the theory, insight is gained into the structure of the multi-objective problem by analyzing the geometry of the directional cones at different stages of learning. Reasons for the apparently rapid rate of initial convergence (but poor rate of final convergence) in typical multi-objective EC algorithms are also described.

2 Directional Multi-objective Optimization

In recent years, there have been a number of advances in steepest descent-type algorithms applied to differentiable, multi-objective optimization problems [1][5]. While they suffer from the same disadvantages as their single objective counterparts (slow final convergence, convergence to local minima), they possess both an explicit test for convergence and rapid initial convergence both of which are desirable properties in many practical design problems. This section reviews the basic concepts of these gradient-based, multi-objective algorithms, describing how to calculate a multi-objective gradient, how it can be used to test for optimality and how it can be used to produce a dominating search direction. In addition, insights are given into the structure of the multi-objective EC optimization problem during initial and final convergence, and reasons for the change in the convergence rate are provided. It should be acknowledged that the concepts described in this paper are only directly applicable to differentiable multi-objective design problems. However, a large number of complex shape and formulation optimization problems [7][8] are differentiable. Moreover, the theory presented may aid the reasoning used in EC algorithm design on a broader class of problems.

2.1 Single-Objective Half-Spaces

One way to generalize the conventional, single-objective steepest descent algorithms to a multi-objective setting is by considering which search directions simultaneously minimize each objective. For any single objective (for instance, the j^{th} objective, f_j), a search direction will reduce the objective's value if it lies in the corresponding negative half-space, H , whose normal vector is the negative gradient vector, as illustrated in Fig. 1. This fact is exploited when second-order, single-objective optimization algorithms are derived, because (as long as the Hessian is positive definite) the search direction will lie in the negative half-space. Therefore, the objective function will decrease in value when points are moved into this half space. It is interesting to note that this concept of a half-space is independent of the objective function's form and does not depend on whether the point is close to the local minima or not. It simply states that a small step in any direction will either increase or decrease the objective.

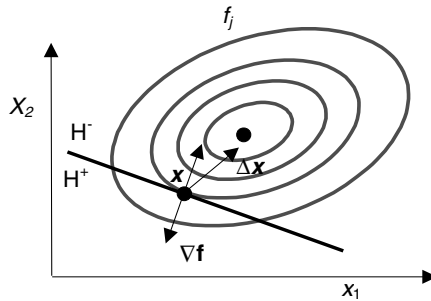


Fig. 1. The half-spaces defined for single-objective f_j . The objective's contours are illustrated as well as the gradient for the current design x .

Also, although one must appeal to probabilistic notions to do so, this idea can be related to modern, real-valued EC, typified by [4]. In such algorithms one can consider those population members selected to survive and recombine to be on one side of an approximate half-space division in the search space, and those deleted from the population without recombination to be on the other side of this division. Note that in the high-performance real-valued EC algorithm introduced in [4], the new individuals generated by GA operators are biased to lie near the selected “parents”, thus enforcing the idea of exploiting the preferred side of this approximate half-space.

2.2 Directional Cones and Multi-objective Search

For the multi-objective space, any search direction that lies in the negative half-space of all the objectives will simultaneously minimize them, and the search direction will be “aligned” with the negative gradients associated with each objective. This is illustrated in Fig. 2.

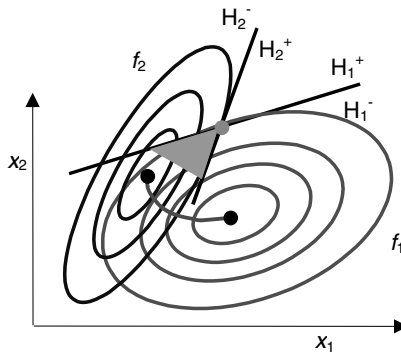


Fig. 2. Directional cones for a 2 variable, 2 objective optimization problem. The Pareto set is the curve between dotted centers, and the directional cone that simultaneously minimizes both objectives is shaded gray.

This region is known as a “directional cone” and in fact, the m half-spaces partition the n -dimensional variable space into 2^m directional cones, within which each objective either increases or decreases in value. This is illustrated in Fig. 3. This interpretation is useful to define search directions that span the Pareto set, rather than converging to it, where some objectives will increase in value and others decrease.

It is also useful to consider the size of the directional cone during initial and final stages of optimization process. When a point is far from the local optima, typically the objective gradients are aligned and the directional cone is almost equal to the half-spaces associated with each objective. Therefore, if the search directions are randomly chosen, there is a 50% chance that a search direction will simultaneously reduce all the objectives. However, when a point is close to the Pareto set/front, the individual objective gradients are contradictory and in almost opposite directions. This follows directly from the definition of Pareto optimality, where, if one objective is decreased, another objective must increase. The size of the directional cone is small. Therefore, if a search direction is selected at random, there is only a small probability that it will lie in this directional cone and thus simultaneously reduce all the objectives. The likelihood is that it will lie in a cone such that some of the objectives will increase and the others decrease, thus spanning the Pareto front, rather than converging to it. This is one of the main differences between single and multi-objective design problems.

Appealing once again to probabilistic notions, this reasoning suggests that the children individuals in a multi-objective EC algorithm should be created to lie within the directional cone. Early in the search process, this is likely to be the same as any given single-objective half space, suggesting that children individuals should be placed near the parents, as in [4]. Later in the search process, this is not the case, and one could expect that locating children near parents will not lead to efficient convergence towards the Pareto front.

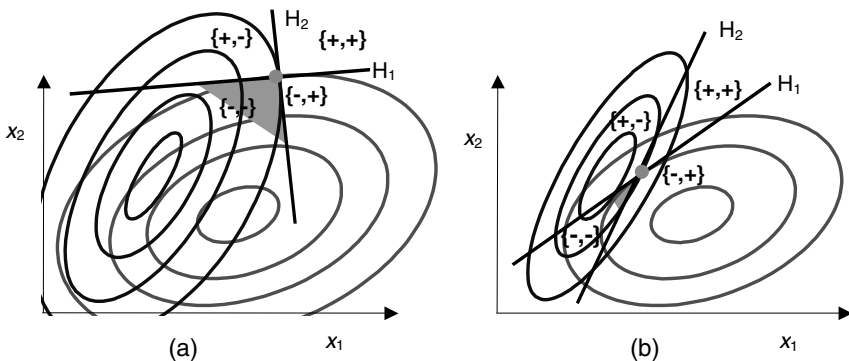


Fig. 3. The directional cones for a 2 parameter, 2 objective design problems during initial (a) and final (b) stages of convergence. The cones are labeled with the sign of the corresponding change in objectives and as can be seen, the descent cone $\{-,-\}$ shrinks to zero during the final stages of convergence.

2.3 Test for Local Pareto Optimality

The interpretation of multi-objective optimization described in the last section is appropriate, as long as the design is not Pareto optimal (i.e., as long as there exists a descent cone that will simultaneously reduce all the objectives). To test whether a design is locally Pareto optimal [5] is an important part of any design process and this can be formulated as:

$$\lambda \in N(\mathbf{J}(\mathbf{x}))$$

for any non-zero vector $\lambda \geq \mathbf{0}$, where $N(\mathbf{J})$ is the null space of the Jacobian matrix, $\mathbf{J}$, and $R(\mathbf{J}^T)$ is the range of $\mathbf{J}^T$. The Jacobian is the matrix of derivative of each variable with respect to each objective. The equation above is equivalent to:

$$\mathbf{J}(\mathbf{x})\lambda = \mathbf{0}$$

The geometric interpretation of this test in objective space (shown in Fig. 4) is that there exists a non-negative combination of the individual gradients that produces an identically zero vector. When this occurs, any changes to the design parameters will affect only $R(\mathbf{J}^T)$, which is orthogonal to λ . Therefore, no changes to the design parameters will produce a descent direction that simultaneously reduces all the objectives. This is the limiting case of the situation described in Section 2.2, during the final stages of convergence, when the gradients become aligned, but in the opposite direction. When the alignment is perfect (local Pareto optimality), any change to the design parameters will increase at least one of the objectives, so the movement will be along the Pareto front, rather than minimizing all the objectives. In fact, for an optimal design, $R(\mathbf{J}^T)$ defines the local tangent to the Pareto front and thus defines the space that must be locally sampled in order to generate the complete local Pareto set/front.

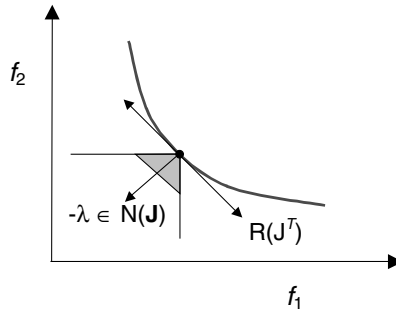


Fig. 4. Geometrical interpretation of the Null space and Range of the Jacobian matrix when a design is Pareto optimal. The vector λ specifies the local normal to the Pareto front.

Once again appealing by analogy to [4], note that concentration of “children” individuals in an EC algorithm near “parents” is not likely to result in individuals within the appropriate directional cone, in a way that is analogous to points being biased to the appropriate half-space in single-objective search. Therefore, to exploit analogous advantages offered by modern, real-valued EC in multi-objective settings, it is appro-

priate to consider further operations as a part of the search process. These are outlined in the following section.

2.4 Multi-objective Steepest Descent

A multi-objective steepest descent search direction must lie in the directional cone that simultaneously reduces all the objectives. This specification can be made unique [1][5] by requiring that the reduction is maximal which can be formulated as:

$$\begin{aligned} (\alpha^*, \mathbf{s}^*) &= \arg \min \alpha + \frac{1}{2} \|\mathbf{s}\|_2^2 \\ \text{st} \quad &\mathbf{J}^T \mathbf{s} \leq \mathbf{1}\alpha \end{aligned} \quad (1)$$

where $\mathbf{J}$ is the local Jacobian matrix, $\mathbf{s}$ is the calculated search direction and α represents the smallest reduction in the objectives' values. This is the primal form of the Quadratic Programming (QP) problem in $(n+1)$ dimensions. It requires as large a reduction in the objectives as possible for a fixed sized variable update. When all the constraints are active, the primal form of the multi-objective optimization problem reduces each objective by the same amount and thus the current point is locally projected towards the Pareto front at 45° in objective space (assuming that the objectives have been scaled to a common range).

It can be shown that when the current point is not Pareto optimal, this problem has a solution such that α^* is negative, and the calculated search direction $\mathbf{s}^*$ therefore lies in the appropriate directional cone. However, it may be easier to solve this problem in the dual form [1][5]:

$$\begin{aligned} \boldsymbol{\lambda}^* &= \arg \min \frac{1}{2} \|\mathbf{J}\boldsymbol{\lambda}\|_2^2 \\ \text{st} \quad &\boldsymbol{\lambda} \geq \mathbf{0} \\ &\sum_j \lambda_j = 1 \end{aligned} \quad (2)$$

This is now a QP problem in m variables. Once it has been solved, the corresponding search direction is given by:

$$\mathbf{s}^* = -\mathbf{J}\boldsymbol{\lambda}^*$$

This search direction will simultaneously reduce all objectives and do so in a maximal fashion, as described by the primal problem. Hence, it is known as the multi-objective steepest descent algorithm. The Multi-Objective Gradient (MOG) that is given by:

$$\mathbf{g} = \mathbf{J}\boldsymbol{\lambda}^* \quad (3)$$

is calculated from a non-negative linear combination of the individual gradients. Therefore, the multi-objective search direction will be "aligned" with the individual gradients, although it should be noted that the degree of alignment, $\boldsymbol{\lambda}^*$, will dynamically change as the point moves closer to the Pareto set. The link with weighted optimization should also be noted, but it should be stressed that this procedure is valid for both convex and concave Pareto fronts.

In order to implement this calculation, it is necessary to obtain the Jacobian $\mathbf{J}$. This can be an expensive operation, especially for practical design problems where it is necessary to perform some form of local experimental design. This is considered further in the next section.

2.5 Dimensionality Analysis

This theory also provides some relevant results about the problem's dimensionality. Firstly, the dimension of both the Pareto set and front has the upper bound $\min(n, m-1)$. This can be derived by simply considering the rank of the Jacobian when a point is Pareto optimal. The dimension of the parameter-objective space mapping is locally $\text{rank}(\mathbf{J}) = \min(n, m)$. When a point is Pareto optimal, this reduces the dimension of the objective space by one. In fact, $\text{rank}(\mathbf{J})$ is the actual local dimension which is bounded above by $\min(n, m-1)$. This is an important result, as it specifies the dimension of the sub-space that a population-based EC algorithm must sample. An EC population must be large enough to adequately sample a space of this size. It is also important as it provides an approximate bound of how the number of objectives and variables should be balanced. The dimension of the actual Pareto set and front is bounded by $\min(n, m-1)$, so it may be unnecessary to have either $n \gg m$ or $n \ll m$. Secondly, it is not always necessary to consider all the design parameters in order to whether a descent direction can be calculated. Suppose that only a single variable is considered for adaptation. As long as this change, either positive or negative, simultaneously reduces all the objectives, it is possible to state that the current point is not Pareto optimal and there exists a descent direction in the current sub-space. This is true for any sub-set of the design parameters considered, and is important as it may allow sparse Jacobian estimates to be used in certain situations.

3 Directional Evolutionary Computation

The work described in this section addresses the question of how to estimate a dominating search direction, based on the information contained in a local neighborhood of designs. In the language of EC, this essentially examines how to perform an "intelligent" crossover in order to produce new designs that dominate the old ones in the population (by placing children in the directional cone). The aim is to minimize the number of actual objective evaluations, by maximally re-using information contained in the population. For each point in the population, a set of local neighbors is used to approximate the local Jacobian and thus calculate the MOG. By searching along this path, a new point will be found that dominates the starting point. In the worst case, each recombination of this sort involves estimating the Jacobian, which is an order $n*m$ computation, and the solution of a quadratic program, which is approximately an $O(\min\{m, n\}^3)$ calculation. However, while this is computationally expensive compared to many conventional crossover operations, this computational expense is likely to be vastly outweighed by the expense of evaluating fitness values for the population (as is usually the case in EC).

3.1 Population-Based Estimation of the Multi-objective Gradient

To begin to understand how the local population can help to determine an “intelligent” crossover operation, consider the trivial case where the local population consists of a point that is dominated by the current point, see Figure 6. By moving along the path $\mathbf{s}$, where $\mathbf{s} = \mathbf{x} - \mathbf{x}_1$, a new design will be found such that it dominates the current design $\mathbf{f}(\mathbf{x})$.

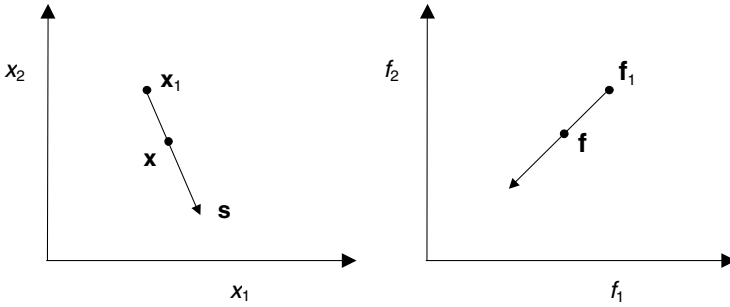


Fig. 5. Calculating a descent direction when the local neighborhood about a design, $\mathbf{x}$, consists of a single dominated design $\mathbf{x}_1$.

In general, the situation is rarely this simple. During the latter stages of convergence, designs rarely dominate each other as they are spread out along the local estimate of the Pareto set/front. In this situation, it is possible to use the primal/dual steepest descent theory described in Section 2.4, as long as the Jacobian information can be generated efficiently. Suppose that a set of r , $r = \min\{m, n\}$, linearly independent designs in the local neighborhood have been identified. Typically, these would be closest set of designs to the current design of interest. As illustrated in Fig. 6, the differences in the design parameters and the objectives along the search directions $\mathbf{s}_i$ can be measured. Two matrices then represent this difference information:

$$\mathbf{X}_s = \frac{\Delta x_i}{\Delta s_j} \quad (4)$$

$$\mathbf{F}_s = \frac{\Delta f_i}{\Delta s_j}$$

and the local Jacobian can be estimated by

$$\mathbf{J} = (\mathbf{X}_x^T \mathbf{X}_s)^{-1} \mathbf{X}_s^T \mathbf{F}_s \quad (5)$$

Therefore, the differences between the point of interest and each member of the local neighborhood is used to estimate gradient information, which is then utilized (2) to

calculate the MOG. This relies on the local neighborhood being sufficiently small so that the gradients can be estimated sufficiently accurately.

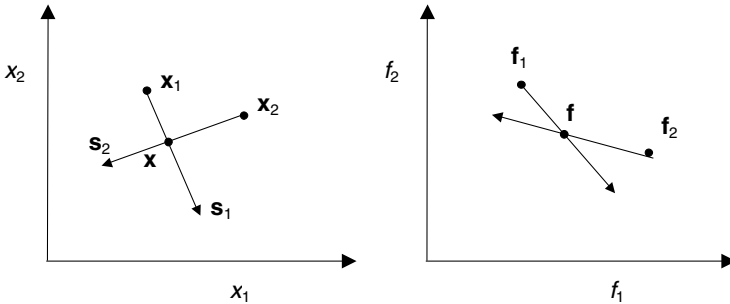


Fig. 6. Estimating the Jacobian matrix by using difference information along the indicated search directions s_i .

3.2 Population-Based Pareto Optimality

When the local Jacobian is calculated using the members of the local population to estimate difference information, it is assumed that the designs are sufficiently well distributed such that they span the necessary r_i dimensions (note that this may be a subset of the complete design space) and that they are sufficiently close to the central point so that any estimation error is small. When this is true, the test for Pareto optimality is described in Section 2.3 and can be (naively) implemented by testing whether the MOG is sufficiently small.

It should be noted that while this paper has concentrated on describing techniques for testing for optimality and calculating descent directions, it is possible to use the geometric insights about the shape of the Pareto set/front in order to specify search directions that span the Pareto set/front as specified by $R(\mathbf{J}^T)$. This will be described in a later paper that analyses the complete algorithm.

4 Example

The use of directional information for multi-objective optimization will now be demonstrated on a simple test problem with 2 variables and 2 objectives. Specifically, we consider two quadratic objective functions, centered on $[0.25, 0.75]$ and $[0.75, 0.25]$ and with Hessian matrices are $\begin{bmatrix} 80 & 40 \\ 40 & 40 \end{bmatrix}$ and $\begin{bmatrix} 40 & 40 \\ 40 & 80 \end{bmatrix}$, respectively. It should be noted that the aim is not to provide a full and rigorous comparison with other approaches, as a complete algorithm has not been described in this paper. Rather the aim is to demonstrate how the theory provides the basis for such an approach, and to give an indication of the power of using directional information.

This concepts described above were implemented as a simple multi-objective optimization procedure where the initial population of 50 points approximately lay along a line with a small amount of random noise added, as illustrated in Fig. 7.

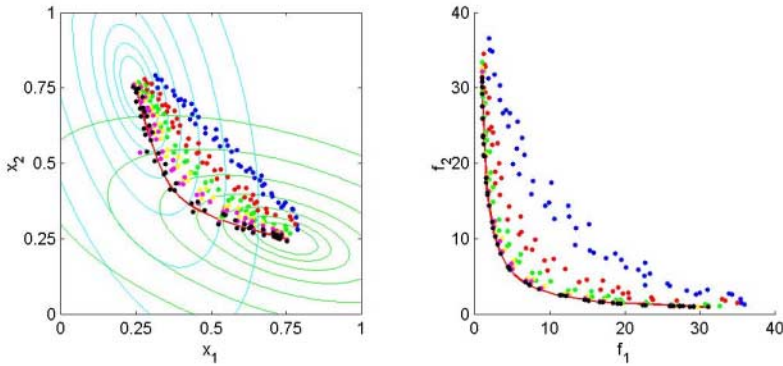


Fig. 7. Using the directional MOG to drive a population towards the Pareto set (left) and front (right). Each progressive population is shown with points in a different color, progressing towards the indicated front.

In Fig. 7, it can be clearly seen that successive iterations produce new “child” values that dominate their parents. Convergence to the Pareto set/front occurred after around 6 iterations. It should be noted that a fixed step size was used here and it would be expected that a more intelligent line search method would produce faster convergence. It should also be noted that the spanning of the Pareto set/front occurs because of the diversity in the initial population. While there is no diversity component in the method described in this paper, it can be clearly seen that the approximate MOG descent calculation projects the points directly towards the Pareto set/front in a rapid and efficient manner. In addition, the development of the diversity component [6] is the subject of current research.

As a final, simple comparison, a naïve genetic multi-objective procedure was developed. A simplex-type scheme was used to adapt the points, where each point was tested against neighboring values, and if the point was dominated or it was too close to a neighboring point (within a threshold distance), it was replaced by a combination of its n neighboring values (thus using averaging of parents as a form of recombination), else it was subject to a small random mutation (addition of Gaussian noise), with a standard deviation = 0.01. The results of this procedure are shown in Fig. 8. The same initial population is used as in Fig. 7. It should be noted that the final population now occurs after 60 iterations and the convergence in variable space is poor. While it is readily acknowledged that more sophisticated combination/selection processes could be used, the aim was to demonstrate that an “intelligent” use of directional information can dramatically improve the rate of convergence for suitably smooth, differentiable design problems. Moreover, we feel that this reasoning can be extended for the formulation of recombination operators in a much broader class of problems.

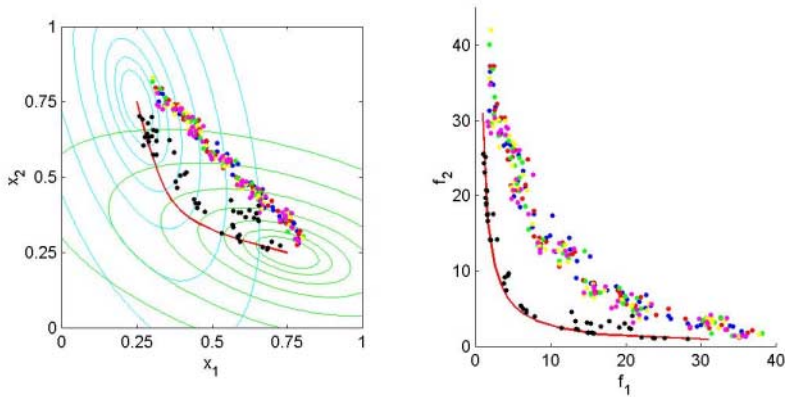


Fig. 8. Using a genetic multi-objective optimization process to solve the simulated problem.

5 Conclusions and Further Work

One must consider how to efficiently estimate a dominating search direction for a multi-objective problem, based on the information contained in a local neighborhood of designs, to properly apply EC to such problems. The theory developed in this paper clarifies how to logically use such information in multi-objective EC. The theory can be both used to explain the performance of current techniques, and gain insights into the problem's dimensionality. Also, the concepts developed (of convergence, directional cones and MOG) are directly applicable to differentiable multi-objective problems, and may be useful in EC algorithm design for a broader class of problems. In particular, a key aspect of EC algorithm design is the specification of recombination and mutation operators. When points are close to the optimal front/set children should be placed in the descent cone, or convergence should be expected to be poor in many cases. This paper shows that appropriate, informed use of direction information in MOEAs will improve performance. Moreover, the authors believe that the theory can both be extended to more logically design operators in non-continuous problems and also to understanding how diversity [6] can be integrated into such schemes, which is a subject of their current research.

References

- [1] Brown, M. 3(2002) Steepest Descent Vector Optimization and Product Design, submitted for publication.in preparation
- [2] Coello, Carlos A., Van Veldhuizen, David A., and Lamont, Gary B (2002) *Evolutionary Algorithms for Solving Multi-Objective Problems*. Kluwer Academic Publishers.
- [3] Deb, K. (2000). *Multi-Objective Optimization Using Evolutionary Algorithms*. Wiley.

- [4] Deb, K , Anand, A., and Joshi, D (2002). A Computationally Efficient Evolutionary Algorithm for Real-Parameter Optimization. KanGAL Report No. 2002003. (to appear in the journal *Evolutionary Computation*).
- [5] Fliege, J. (2000) Steepest Descent Methods for Multicriteria Optimization. *Mathematical Methods of Operations Research*, 51(3), pp. 479–494.
- [6] Laumanns, M., Thiele, L., Deb, K., and Zitzler, E. (2002) Combining Convergence and Diversity in Evolutionary Multi-objective Optimization. *Evolutionary Computation* 10(3) pp. 263–282.
- [7] Parmee, I. C., Watson, A., Cvetkovic, D., Bonham, C. (2000). Multi-objective Satisfaction within an Interactive Evolutionary Design Environment. *Evolutionary Computation*. 8(2). pp 197–222.
- [8] Product Formulation using Intelligent Software,
<http://www.brad.ac.uk/acad/profits/website/>
- [9] Sobieszczanski-Sobieski, J.; and Haftka, R.T. (1997). Multidisciplinary aerospace design optimization: survey of recent developments. *Structural Optimization* 14, pp. 1-23.
- [10] Vicini, A., and Quagliarella, D. (1997) Inverse and direct airfoil design by means of a multi-objective genetic algorithm. *AIAA Journal*. 35(9).

Pruning Neural Networks with Distribution Estimation Algorithms

Erick Cantú-Paz

Center for Applied Scientific Computing
Lawrence Livermore National Laboratory
Livermore, CA 94551
`cantupaz@llnl.gov`

Abstract. This paper describes the application of four evolutionary algorithms to the pruning of neural networks used in classification problems. Besides of a simple genetic algorithm (GA), the paper considers three distribution estimation algorithms (DEAs): a compact GA, an extended compact GA, and the Bayesian Optimization Algorithm. The objective is to determine if the DEAs present advantages over the simple GA in terms of accuracy or speed in this problem. The experiments considered a feedforward neural network trained with standard backpropagation and 15 public-domain and artificial data sets. In most cases, the pruned networks seemed to have better or equal accuracy than the original fully-connected networks. We found few differences in the accuracy of the networks pruned by the four EAs, but found large differences in the execution time. The results suggest that a simple GA with a small population might be the best algorithm for pruning networks on the data sets we tested.

1 Introduction

The success of neural networks (NNs) largely depends on their architecture, which is usually determined by a trial-and-error process. Evolutionary algorithms (EAs) have been used in numerous ways to optimize the network architecture to reach the highest possible classification accuracy [1,2]. In the present paper, we examine neural network pruning by four evolutionary algorithms to improve the generalization accuracy in classification problems.

We experimented with a simple genetic algorithm (sGA) and three distribution estimation algorithms (DEAs): a compact GA (cGA), an extended compact GA (ecGA), and the Bayesian Optimization Algorithm (BOA). Instead of the mutation and crossover operations of conventional GAs, DEAs use a statistical model of the individuals that survive selection to generate new individuals. Numerous experimental and theoretical results show that DEAs can solve hard problems reliably and efficiently [3,4,5].

The objective of this study is to determine if DEAs present advantages over simple GAs in terms of accuracy or speed when applied to neural network pruning. The experiments used conventional feedforward perceptrons with one hidden

layer and were trained with the backpropagation algorithm. The experiments used 13 public-domain and two artificial data sets.

Our target was to maximize the accuracy of classification. The experiments demonstrate that, in most cases, the accuracy of the pruned networks is at least as good as that of fully-connected networks. We found few significant differences in the accuracy of networks pruned by the four EAs, but found large differences in the execution time.

The next section presents background on neural network pruning, including some previous applications of EAs to this task. Section 3 describes the algorithms, data sets, and the method used to compare the algorithms. The experimental results are presented in section 4. Section 5 concludes this paper with a summary, the conclusions of this study, and a discussion of future research directions.

2 Neural Network Pruning

It is well known that a network that is too big for a particular classification task is more likely to overfit the training data and have poor performance on unseen examples (i.e., poor generalization) than a small network. Therefore, a heuristic to obtain good generalization is to use the smallest network that will learn to classify correctly the training data. However, the optimal network size is usually unknown and tedious experimentation becomes necessary to find it. An alternative to improve generalization is to train a network that is believed to be larger than necessary and prune the excess parts.

Numerous algorithms have been used to prune neural networks [6]. Pruning begins by training a fully-connected neural network. Most pruning methods delete a single weight at a time in a greedy fashion, which may result in sub-optimal pruning. Additionally, many pruning methods fail to account for the interactions among multiple weights. This may be problematic if deleting one weight makes it appear as if another weight that should be pruned is important for the operation of the network. An algorithm that considers weight interactions and more than one weight at a time may have a better chance of reducing the size of the network significantly without affecting the classification accuracy. For these reasons, GAs and DEAs seem promising for NN pruning.

Genetic algorithms have been used to prune networks with good results [7,8,9]. Applying GAs to prune networks is straightforward: The chromosomes contain one bit for each weight of the original network, and the value of the bit determines whether the weight will be used in the final network. This simple binary encoding is used in the experiments in the present paper. More sophisticated methods of simultaneously training and pruning the networks were introduced by Schmidt and Stidsen [10].

Whitley [11] suggests to retrain the network for a few epochs after pruning the weights. We performed experiments to test this idea, but our experiments show only limited advantages of retraining.

It is also possible to prune entire (input and hidden) nodes, but in the present paper we experiment only with the more common approach of pruning individual weights. We leave pruning units to future work.

3 Methods

This section describes the algorithms and the data sets used in this paper as well as the statistical method used to compare the algorithms.

3.1 Algorithms

The simple genetic algorithm in this study uses binary strings, pairwise tournament selection without replacement, uniform crossover, and bitwise point mutation. Simple GAs such as this have been used successfully in many applications. However, it has long been recognized that the problem-independent crossover operators used in simple GAs can disrupt groups of related variables and prevent the algorithm from reaching the global optimum, unless exponentially-sized populations are used. (Thierens [12] gives a good description of this problem).

One approach to identify and exploit the relationships among variables is to estimate the joint distribution of the individuals that survive selection and use this model to generate new individuals. The complexity of the models has increased over time as more sophisticated methods of building models from data and more powerful computers become available. Interested readers can consult the reviews by Pelikan et al. [13] and Larrañaga et al. [14].

The simplest model-building EA used in the experiments reported here is the compact GA [15]. This algorithm assumes that the variables (bits) that represent the problem are independent, and therefore the cGA models the population as a product of Bernoulli distributions. The compact GA receives its name from its small memory requirements: Instead of using an explicit population, the cGA uses a vector p of length equal to the problem's length, l . Each element of p contains the probability that a sample will take the value 1. If the Bernoulli trial is not successful the sample will be 0. All positions of p are initialized to 0.5 to simulate the usual uniform random initialization of simple GAs. New individuals are obtained by sampling consecutively from each position of p and concatenating the values obtained. The probabilities vector is updated by comparing the fitness of two individuals obtained from it. For each p_k , $k = 1, \dots, l$, if the fittest individual has a 1 in the k -th position, p_k is increased by $1/n$, where n is the size of the virtual population that the user wants to simulate. Likewise, if the fittest individual has a 0 in the k -th position, p_k is decreased by $1/n$. The cGA iterates until all positions in p_k contain either zero or one. PBIL [16] and the UMDA [17] are other algorithms that use univariate models and operate on binary alphabets. They differ from the cGA in the method to update the probabilities vector.

The extended compact GA [18] uses a product of marginal distributions on a partition of the variables. In this model, subsets of variables are modeled jointly, and the subsets are considered independent of other subsets. Formally, the model

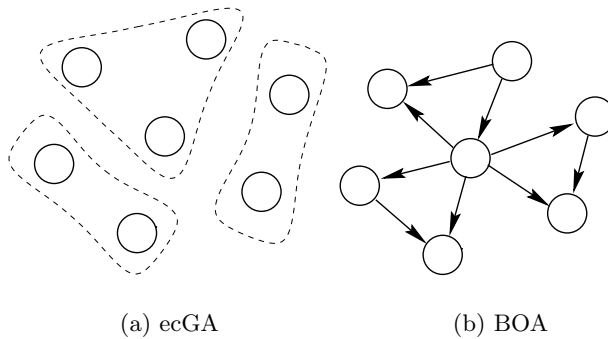


Fig. 1. Representation of the models used in the ecGA and the BOA. Variables are represented as circles. The ecGA groups related variables into subsets, but cannot represent individual relationships among variables in the same subset.

is $P = \prod_{i=0}^m P_i$, where m is the number of subsets in a partition of the variables and P_i represents the distribution of the i -th subset. The distribution of a subset with k members is stored in a table with $2^k - 1$ entries. The challenge is to find a partition that models the population correctly. Harik [18] proposed a greedy search that initially supposes that all variables are independent. The model search tries to merge all pairs of subsets and chooses the merger that minimizes a complexity measure based on information theory. The search continues until no further subsets can be merged. In contrast to the cGA, the ecGA has an explicit population that is evaluated and is subject to selection at each iteration of the algorithm. The algorithm builds the model considering only those solutions that survive selection. The population is initialized randomly, and new individuals are generated by sampling consecutively from the m subset distributions.

The Bayesian Optimization Algorithm [3] models the selected individuals using a Bayesian network, which can represent dependence relations among an arbitrary number of variables. Independently, Etxeberria and Larrañaga [4] and Mühlenbein and Mahnig [5] introduced similar algorithms. The BOA uses a greedy search to optimize the Bayesian Dirichlet metric, a measure of how well the network represents the data (the BOA could use other metrics). The user specifies the maximum number of incoming edges to any node of the network. This number corresponds to the highest degree of interaction assumed among the variables of the problem. As the ecGA, the BOA builds the model considering only the solutions that survived selection. New individuals are generated by sampling from the Bayesian network.

The main difference between the ecGA and the BOA is the model that they use to represent the survivors. Figure 1 illustrates the different models used by the ecGA and the BOA. The ecGA cannot represent individual relationships among the variables in a subset.

The experiments used the C++ implementations of the ecGA [19] and the BOA version 1.0 [20] that are distributed by their authors on the web (at <http://www-illigal.ge.uiuc.edu>). The ecGA code has a non-learning mode that emulates the cGA. The sGA and the neural network were developed in C++. All programs were compiled with g++ version 2.96 using -O2 optimizations. The experiments were executed on a single processor of a Linux (Red Hat 7.2) workstation with dual 2.4 GHz Intel Xeon processors and 512 Mb of memory. The ecGA and the BOA codes were modified to use a Mersenne Twister random number generator, which was also used in the GA and the data partitioning.

The algorithms used populations with 1024 individuals and were initialized uniformly at random. The GA used uniform crossover with probability 1.0, and mutation with probability $1/l$, where l was the length of the chromosomes and corresponds to the total number of weights in the network. Promising solutions were selected with pairwise binary tournaments without replacement. The cGA, ecGA, and the BOA used the default parameters provided in their distributions: The cGA and ecGA used tournaments among 16 individuals, and the BOA used truncation selection with a threshold of 50%. All algorithms were terminated after observing no improvement in the best individual over five consecutive generations, or until a limit of 50 generations was reached.

The network used in the experiments was a fully-connected perceptron with a single hidden layer. The hidden and output units compute their output as $f(\text{net}) = \tanh(\text{net})$, where $\text{net} = \sum_{i=1}^d x_i w_i + w_0$ is the net activation, the x_i are inputs to the unit, w_i are the connection weights and w_0 is a bias term. The weights were initialized uniformly at random in the interval $[-1,1]$. Before each EA run, a fully-connected network was trained with simple backpropagation using a learning rate of 0.15 and a momentum term of 0.9. In each epoch, the examples were presented to the network in a different random order. The sizes of the network and the number of training epochs varied for each data set and are specified in table 1.

For all the algorithms, the classification accuracy of the pruned network on the training data served as the fitness function. In cases where the pruned network was retrained with backpropagation, the algorithms exploited the Baldwin effect: The retrained pruned network was used to evaluate the fitness, but the retrained weights were not inherited. Note that the fitness measure does not bias the search explicitly toward networks with few weights. Adding this bias is a future extension of the work presented here.

3.2 Data Sets

The data sets used in the experiments are described in table 1. The data sets are available in the UCI machine learning repository [21], except for Random21 and Redundant21, which are artificial data sets with 21 features each. The target concept of these two data sets is whether the first nine features are closer to $(0,0,\dots,0)$ or $(9,9,\dots,9)$ in Euclidean distance. The features were generated uniformly at random in the range $[3,6]$. All the features in Random21 are random,

Table 1. Description of the data sets used in the experiments. For each data set, the table shows the number of instances; the number of classes; the number of continuous and discrete features; the number of input, hidden, and output units; and the number of epochs of backpropagation used to train the networks.

Domain	Cases Class		Features		Neural Network			
			Cont.	Disc.	Input	Output	Hidden	Epochs
Breast-Cancer	699	2	9	–	9	1	5	20
Credit-Australian	653	2	6	9	46	1	10	35
Credit-German	1000	2	7	13	62	1	10	30
Heart-Cleveland	303	2	6	7	26	1	5	40
Housing	506	3	12	1	13	3	2	70
Ionosphere	351	2	34	–	34	1	10	40
Iris	150	3	4	–	4	3	5	80
Kr-vs-kp	3196	2	–	36	74	1	15	20
Pima-Diabetes	768	2	8	–	8	1	5	30
Segmentation	2310	7	19	–	19	7	15	20
Sonar	208	2	60	–	60	1	10	60
Vehicle	846	4	18	–	18	4	10	40
Wine	178	3	13	–	13	3	5	15
Random21	2500	2	21	–	21	1	1	100
Redundant21	2500	2	21	–	21	1	1	100

and the first, fifth, and ninth features are repeated four times each in Redundant21. We took the definition of Redundant21 from the paper by Inza et al. [22].

Each numeric feature in the data was linearly normalized to the interval $[-1, 1]$. The discrete features and the class labels were encoded with the usual 1-in- C coding if there are $C > 2$ values (one of the C outputs is set to 1 and the rest to -1). Binary values were encoded as a single -1 or 1 value.

The instances with missing values in Credit-Australian were deleted. Following the usual practice, the missing values in Pima-Diabetes (denoted with zeroes) were not removed and were treated as if their values were meaningful. Following Lim et al. [23], the classes in Housing were obtained by discretizing the attribute “mean value of owner-occupied homes” as follows: class = 1 if $\log(\text{median value}) \leq 9.84$, class = 2 if $9.84 < \log(\text{median value}) \leq 10.075$, and class = 3 otherwise.

3.3 Evaluation Method

To evaluate the generalization accuracy of the pruning methods, we used 5 iterations of 2-fold crossvalidation (5x2cv). In each iteration, the data were randomly divided in halves. One half was input to the EAs. The best pruned network found by the EA was tested on the other half of the data. The accuracy results presented in table 2 are the averages of the ten tests.

To determine if the differences among the algorithms were statistically significant, we used a combined F test proposed by Alpaydin [24]. Let $p_i^{(j)}$ denote

Table 2. Mean accuracies found in the 5x2cv experiments. **Bold** typeface indicates the best result and those not significantly different from the best according to the combined F test at a 0.05 level of significance.

Domain	Unpruned	sGA	cGA	ecGA	BOA
Breast Cancer	96.39	96.54	96.13	95.84	96.42
Cr-Australian	82.53	85.78	85.75	86.18	85.84
Cr-German	70.12	70.68	70.92	70.30	70.14
Heart-Cleveland	58.17	89.70	88.05	88.78	89.37
Housing	64.62	75.36	67.11	64.18	66.24
Ionosphere	84.77	84.61	82.95	82.22	84.22
Iris	94.53	92.93	70.13	67.73	93.60
Kr-vs-kp	74.30	92.56	93.53	93.81	93.85
Pima-Diabetes	73.30	74.84	75.91	76.04	75.88
Segmentation	44.16	64.02	62.45	64.32	63.66
Sonar	73.17	83.46	86.15	84.90	83.55
Vehicle	69.71	78.20	76.73	76.64	78.62
Wine	95.16	94.15	89.88	87.41	93.48
Random21	91.70	94.04	94.08	94.03	94.09
Redundant21	91.75	95.77	95.82	95.82	95.72

the difference in the accuracy rates of two classifiers in fold j of the i -th iteration of 5x2cv, $\bar{p} = (p_i^{(1)} + p_i^{(2)})/2$ denote the mean, and $s_i^2 = (p_i^{(1)} - \bar{p})^2 + (p_i^{(2)} - \bar{p})^2$ the variance, then

$$f = \frac{\sum_{i=1}^5 \sum_{j=1}^2 \left(p_i^{(j)}\right)^2}{2 \sum_{i=1}^5 s_i^2}$$

is approximately F distributed with 10 and 5 degrees of freedom, and we rejected the null hypothesis that the two algorithms have the same error rate with a 0.05 level of significance if $f > 4.74$ [24]. The algorithms used the same data partitions and started from identical initial populations.

4 Experiments

Table 2 has the average accuracies obtained with each method. For each data set, the best observed result and those that according to the combined F test are not significantly different from the best are highlighted in **bold** type.

These results suggest that, in most cases, the accuracy of the pruned networks is at least as good as the original fully-connected networks. In these experiments the networks were not retrained after pruning. Unexpectedly, pruning does not seem to have harmful effects on the accuracy, except in two cases (Iris and Wine) where the networks pruned with cGA and ecGA perform significantly worse than the fully-connected networks. The simple GA and the BOA performed equally well, and their results were not significantly different than the best result for all the data sets we tried.

Pruning results in only minor accuracy gains over the fully-connected networks, except when the fully-connected nets performed poorly. In those cases, pruning resulted in dramatic improvements. For example, the pruned networks on Heart-Cleveland show improvements of $\approx 30\%$ in accuracy, while in Kr-vs-kp and Segmentation the improvements are $\approx 20\%$, and in Vehicle the improvements are $\approx 10\%$.

One reason why pruning might improve the accuracy is because pruning may eliminate the effect of irrelevant or redundant inputs. The experiments with Random21 and Redundant21 were intended to explore this hypothesis. In Random21, the pruning methods always selected weights corresponding to the nine true inputs, but the algorithms always selected two or three additional weights corresponding to random inputs. However, the performance does not seem to degrade much. It is possible that backpropagation had assigned low values to those irrelevant weights or it may be that the hypothesis that pruning improves the accuracy by removing irrelevant weights is wrong. Further work is required to clarify these results. In Redundant21, the pruning methods did not eliminate the redundant features. In fact, the pruned networks retained more than 20 of their 24 weights. Again, it is not clear why the performance did not degrade with the redundant weights and additional work is needed to address this issue.

With respect to the number of weights of the final networks, all algorithms had similar results, successfully pruning between 30 and 50% of the total weights (with the exception of Redundant 21 discussed above).

Table 3 shows that the sGA and the BOA finished in similar number of generations (except for Credit-Australian and Heart-Cleveland), and were the slowest algorithms in most cases. On most data sets, the ecGA finishes faster than the other algorithms.¹ However, the ecGA produced networks with inferior accuracy than the other methods or the fully-connected networks in three cases (Housing, Iris, and Wine). Despite the occasional inferior accuracies, it *seems* that the ecGA is a good pruning method with a good compromise of accuracy and execution time. However, further experiments described below suggest that simple GAs might be the best option.

We performed additional experiments retraining the networks after pruning for one, two, and five epochs of backpropagation (results not shown). In most cases, retraining the networks improves the classification accuracy only slightly over pruning without retraining (1–2%), and there does not appear to be a significant advantage to retrain for more than one epoch. Among the data sets we tested, the largest impact of retraining (using one epoch) was in Housing with an increase of approximately 7% over pruning without retraining.

Retraining, however, had a large impact on the number of generations until the algorithms terminated. In most cases, retraining for one epoch reduced the generations by approximately 40%. Only in one case (sGA on Random21) the

¹ The time needed by the DEAs to build a model of the selected individuals and generate new ones was short compared to the time consumed evaluating the individuals, so one generation took roughly the same time in all algorithms.

Table 3. Mean generations until termination. **Bold** typeface indicates the best result and those not significantly different from the best according to the combined F test at a 0.05 level of significance.

Domain	sGA	cGA	ecGA	BOA
Breast Cancer	9.2	6.7	7	10.9
Credit-Australian	10	14	14.9	14.4
Credit-German	17.1	22.8	21.3	14.3
Heart-Cleveland	9.8	10.4	10.2	15.8
Housing	19.4	7.4	7.1	18.6
Ionosphere	16.8	15.7	15.1	17.8
Iris	10.1	5.9	5.9	10.1
Kr-vs-kp	37.7	28.8	26	35.7
Pima-Diabetes	12.8	14.7	11.5	14.2
Segmentation	26	18.1	17.4	24.9
Sonar	14.5	20.5	19.3	16.9
Vehicle	26.1	16.5	14.8	30.2
Wine	12.5	9.9	9.4	11.7
Random21	13.6	9	9.1	14.8
Redundant21	13.7	8.5	8.5	16.1

number of generations increased (from 13.6 to 20). Retraining for more than one epoch did not have a noticeable effect on the number of generations. Of course, in all cases, retraining increased the total execution time considerably.

The population size of 1024 individuals was chosen because the DEAs require a large population to estimate correctly the parameters of the models of selected individuals. However, for the simple GAs, it is likely that such a large population is unnecessary. In additional experiments, we set the sGA population size to the largest of 20 or $3\sqrt{l}$, where l is the size of the chromosomes (number of weights in the network). The only significant difference in accuracy between the sGA with 1024 individuals and the smaller population was in Iris (87.73% with 20 individuals vs. 92.93% with 1024). There were no other significant differences with the sGA with the large population or the best pruning method for each data set. Naturally, the execution time was much shorter with the smaller populations. Therefore, for pruning neural networks, it seems that the best alternative among the algorithms we examined is a simple GA with small populations.

5 Conclusions

This paper presented experiments with four evolutionary algorithms applied to neural network pruning. The experiments considered public-domain and artificial data sets. With these data sets we found that there are few differences in the accuracy of networks pruned by the four EAs, but that the extended compact GA needs fewer generations to finish. However, we also found that, in a few cases, the ecGA results in networks with lower accuracy than those obtained by the other EAs or a fully-connected network.

We also found that in most cases retraining the pruned networks improves the classification accuracy only very slightly but incurs in a much higher computational cost. Therefore, it appears that retraining is only recommended in applications where time is not critical.

Additional experiments revealed that a simple GA with a small population can reach results that are not significantly different from the best pruning methods. Since the smaller populations result in much shorter execution times, the simple GA seems to have an advantage over the other methods.

The experiments with redundant and irrelevant attributes presented here are not conclusive and additional work is needed to clarify those results. Future work is also necessary to explore methods to improve the computational efficiency of the algorithms to deal with much larger data sets. In particular, subsampling the training sets and parallelizing the fitness evaluations seem like promising alternatives. Another possible extensions of this work are to prune entire units and attempt to reduce the size of the pruned networks by including a bias toward small networks in the fitness function.

Acknowledgments. I thank Martin Pelikan for providing the graphs in figure 1 and the anonymous reviewers for their detailed and constructive comments. UCRL-JC-151521. This work was performed under the auspices of the U.S. Department of Energy by University of California Lawrence Livermore National Laboratory under contract No. W-7405-Eng-48.

References

1. Yao, X.: Evolving artificial neural networks. *Proceedings of the IEEE* **87** (1999) 1423–1447
2. Castillo, P.A., Arenas, M.G., Castillo-Valdivieso, J.J., Merelo, J.J., Prieto, A., Romero, G.: Artificial neural networks design using evolutionary algorithms. In: *Proceedings of the Seventh World Conference on Soft Computing*. (2002)
3. Pelikan, M., Goldberg, D.E., Cantú-Paz, E.: BOA: The Bayesian optimization algorithm. In Banzhaf, W., Daida, J., Eiben, A.E., Garzon, M.H., Honavar, V., Jakiela, M., Smith, R.E., eds.: *Proceedings of the Genetic and Evolutionary Computation Conference 1999: Volume 1*, San Francisco, CA, Morgan Kaufmann Publishers (1999) 525–532
4. Etxeberria, R., Larrañaga, P.: Global optimization with Bayesian networks. In: *II Symposium on Artificial Intelligence (CIMAF99)*. (1999) 332–339
5. Mühlenbein, H., Mahnig, T.: FDA-A scalable evolutionary algorithm for the optimization of additively decomposed functions. *Evolutionary Computation* **7** (1999) 353–376
6. Reed, R.: Pruning algorithms—a survey. *IEEE Transactions on Neural Networks* **4** (1993) 740–747
7. Whitley, D., Starkweather, T., Bogart, C.: Genetic algorithms and neural networks: Optimizing connections and connectivity. *Parallel Computing* **14** (1990) 347–361
8. Hancock, P.J.B.: Pruning neural networks by genetic algorithm. In Aleksander, I., Taylor, J., eds.: *Proceedings of the 1992 International Conference on Artificial Neural Networks*. Volume 2., Amsterdam, Netherlands, Elsevier Science (1992) 991–994

9. LeBaron, B.: An evolutionary bootstrap approach to neural network pruning and generalization. unpublished working paper (1997)
10. Schmidt, M., Stidsen, T.: Using GA to train NN using weight sharing, weight pruning and unit pruning. Technical report, Aarhus University, Computer Science Department, Aarhus, Denmark (1995)
11. Whitley, D., Bogart, C.: The evolution of connectivity: Pruning neural networks using genetic algorithms. Technical Report CS-89-113, Colorado State University, Department of Computer Science, Fort Collins (1989)
12. Thierens, D.: Scalability problems of simple genetic algorithms. *Evolutionary Computation* **7** (1999) 331–352
13. Pelikan, M., Goldberg, D.E., Lobo, F.: A survey of optimization by building and using probabilistic models. IlliGAL Report No. 99018, University of Illinois at Urbana-Champaign, Illinois Genetic Algorithms Laboratory, Urbana, IL (1999)
14. Larrañaga, P., Etxeberria, R., Lozano, J.A., Peña, J.M.: Optimization by learning and simulation of Bayesian and Gaussian networks. Tech Report No. EHU-KZAA-1K-4/99, University of the Basque Country, Conostia-San Sebastian, Spain (1999)
15. Harik, G.R., Lobo, F.G., Goldberg, D.E.: The compact genetic algorithm. In: Proceedings of the 1998 IEEE International Conference on Evolutionary Computation, Piscataway, NJ, IEEE Service Center (1998) 523–528
16. Baluja, S.: Population-based incremental learning: A method for integrating genetic search based function optimization and competitive learning. Tech. Rep. No. CMU-CS-94-163, Carnegie Mellon University, Pittsburgh, PA (1994)
17. Mühlenbein, H.: The equation for the response to selection and its use for prediction. *Evolutionary Computation* **5** (1998) 303–346
18. Harik, G.: Linkage learning via probabilistic modeling in the ECGA. IlliGAL Report No. 99010, University of Illinois at Urbana-Champaign, Illinois Genetic Algorithms Laboratory, Urbana, IL (1999)
19. Lobo, F.G., Harik, G.R.: Extended compact genetic algorithm in C++. IlliGAL Report No. 99016, University of Illinois at Urbana-Champaign, Illinois Genetic Algorithms Laboratory, Urbana, IL (1999)
20. Pelikan, M.: A simple implementation of the Bayesian optimization algorithm (BOA) in C++ (version 1.0). IlliGAL Report No. 99011, University of Illinois at Urbana-Champaign, Illinois Genetic Algorithms Laboratory, Urbana, IL (1999)
21. Blake, C., Merz, C.: UCI repository of machine learning databases (1998)
22. Inza, I., Larrañaga, P., Etxeberria, R., Sierra, B.: Feature subset selection by Bayesian networks based on optimization. *Artificial Intelligence* **123** (1999) 157–184
23. Lim, T.J., Loh, W.Y., Shih, Y.S.: A comparison of prediction accuracy, complexity, and training time of thirty-three old and new classification algorithms. *Machine Learning* **40** (2000) 203–228
24. Alpaydin, E.: Combined $5 \times 2cv$ F test for comparing supervised classification algorithms. *Neural Computation* **11** (1999) 1885–1892

Are Multiple Runs of Genetic Algorithms Better than One?

Erick Cantú-Paz¹ and David E. Goldberg²

¹ Center for Applied Scientific Computing
Lawrence Livermore National Laboratory
7000 East Avenue, Livermore, CA 94550
`cantupaz@llnl.gov`

² Department of General Engineering
University of Illinois at Urbana-Champaign
104 S. Mathews Avenue Urbana, IL 61801
`deg@uiuc.edu`

Abstract. There are conflicting reports over whether multiple independent runs of genetic algorithms (GAs) with small populations can reach solutions of higher quality or can find acceptable solutions faster than a single run with a large population. This paper investigates this question analytically using two approaches. First, the analysis assumes that there is a certain fixed amount of computational resources available, and identifies the conditions under which it is advantageous to use multiple small runs. The second approach does not constrain the total cost and examines whether multiple properly-sized independent runs can reach the optimal solution faster than a single run. Although this paper is limited to additively-separable functions, it may be applicable to the larger class of nearly decomposable functions of interest to many GA users. The results suggest that, in most cases under the constant cost constraint, a single run with the largest population possible reaches a better solution than multiple independent runs. Similarly, a single large run reaches the global faster than multiple small runs. The findings are validated with experiments on functions of varying difficulty.

1 Introduction

Suppose that we are given a fixed number of function evaluations to solve a particular problem with a genetic algorithm (GA). How should we use these evaluations to maximize the expected quality of the solution? One possibility would be to use all the evaluations in a single run of the GA with the largest population possible. This approach seems plausible, because it is well known that, in general, the solution quality improves with larger populations. Alternatively, we could use a smaller population and run the GA multiple times, keeping the best solution found by the different runs. Although the quality per run is expected to decrease, we would have more chances of reaching a good solution.

This paper examines the tradeoff between increasing the likelihood of success of a single run vs. using more trials to reach the goal. The first objective is to

determine what configuration reaches solutions with the highest quality. The paper also examines the question of single vs. multiple runs removing the constant cost constraint. The objective in this case is to determine what configuration reaches the solution faster.

It would be desirable to find that multiple runs are advantageous, because they could be executed concurrently on different processors. Multiple independent runs are a special case of island-model parallel GAs, and have been studied in that context before with conflicting and controversial results [1,2,3,4,5]. Some results suggest that multiple runs can reach solutions of similar or better quality than a single run in a shorter time, which implies that superlinear speedups are possible.

Most of the previous work on this topic has been experimental, which makes it difficult to identify the problem characteristics that give an advantage to multiple runs. Instead of trying to analyze experimental results from a set of arbitrarily-chosen problems, we use simple mathematical models and consider only additively separable functions. The paper clearly shows when one approach can be superior, and reveals that, for the functions considered, multiple runs are preferable only in conditions of limited practical value.

The paper also considers the extreme case when multiple runs with a single individual—which are equivalent to random search—are better in terms of expected solution quality than a single GA. Although it is known that in some problems random search must be better than GAs [6], it is not clear on what problems this occurs. This paper sheds some light on this topic.

The next section summarizes related work on this area. The gambler's ruin (GR) model [7] is summarized in section 3 and extended to multiple independent runs in section 4. Section 5 presents experiments that validate the accuracy of the models. Section 6 lifts the total cost constraint and discusses multiple short runs. Finally, section 7 presents a summary and the conclusions.

2 Related Work

Since multiple runs can be executed in parallel, they have been considered by researchers working with parallel GAs. Tanese [1] found that, in some problems, the best overall solution found in any generation by multiple isolated populations was at least as good as the solution found by a single run. Similarly, multiple populations showed an advantage when she compared the best individual in the final generation. However, when she compared the average population quality at the end of the experiments, the single runs seemed beneficial.

Other studies also suggest that multiple isolated runs can be advantageous. For example, Shonkwiler [2] used a Markov chain model to argue that multiple small independent GAs can reach the global solution using fewer function evaluations than a single GA. He suggested that superlinear parallel speedups are possible if the populations are executed concurrently on a parallel computer.

Nakano, Davidor, and Yamada [8] proved that, under the fixed cost constraint, there is an optimal population size and corresponding run count that

maximizes the chances of reaching a solution of certain quality, if the single-run success probability increases with larger populations until it reaches a saturation point (less than 1). The method used in the current paper can be used to find this optimum, but a numerical optimization would be required, because efforts to characterize the optimal configuration in closed form have been unsuccessful.

Cantú-Paz and Goldberg [3] compared multiple isolated runs against a single run that reaches a solution of the same expected quality. They determined that—even without a fixed time constraint—the savings on execution time seemed marginal when compared against a single GA, and recommended against using isolated runs. The findings in the present paper, however, show that with the cost constraint there are some cases where multiple runs are advantageous.

Recently, Fuchs [4] and Fernández et al. [5] studied empirically multiple isolated runs of genetic programming. They found that in some cases it is advantageous to use multiple small runs. Luke [9] studied the tradeoff between executing a single run for many generations or using multiple shorter runs to find solutions of higher quality given a fixed amount of time. In two out of three problems, his experiments showed that multiple short runs were preferable.

There have been several attempts to characterize the problems in which GAs perform better than other methods [10,11]. However, without relating the performance of the algorithms to properties of the problems it is difficult to make predictions and recommendations for unseen problems, even if they belong to the same class. This paper identifies cases where random search reaches better solutions based on properties that describe the difficulty of the problems.

3 The Gambler's Ruin Model

It is common in GAs to encode the variables of the problem using a finite alphabet Σ . A schema is a string over $\Sigma \cup \{*\}$ that represents the set of individuals that have a fixed symbol $F \in \Sigma$ in exactly the same positions as the schema. The $*$ is a “don't care” symbol that matches anything. For example, in a domain that uses 10-bit binary strings, the individuals that start with 1 and have a 0 in the second position are represented by the schema $10*****$.

The number k of fixed positions in a schema is its order. Low-order highly-fit schemata are sometimes called building blocks (BBs) [12]. Following Harik et al. [7], we refer to the lowest-order schema that consistently leads to the global optimum as the correct BB. In this view, the correct BB must (1) match the global optimum *and* (2) have the highest average fitness of all the schemata in the same partition. All other schemata in the partition are labeled as incorrect.

Harik, Cantú-Paz, Goldberg, and Miller [7] modeled selection in GAs as a biased random walk. The number of copies of the correct BB in a population of size n is represented by the position, x , of a particle on a one-dimensional space. Absorbing barriers at $x = 0$ and $x = n$ bound the space, and represent ultimate convergence to the wrong and to the right solutions, respectively. The initial position of the particle, x_0 , is the number of copies of the correct BB in the initial population.

At each step of the random walk there is a probability, p , of obtaining one additional copy of the correct BB. This probability depends on the problem that the GA is facing, and Goldberg et al. [13] showed how to calculate it for functions composed of m uniformly-scaled subfunctions. The probability that a particle will eventually be captured by the absorbing barrier at $x = n$ is [14]

$$P_{bb}(x_0, n) = \frac{1 - \left(\frac{q}{p}\right)^{x_0}}{1 - \left(\frac{q}{p}\right)^n} \quad (1)$$

where $q = 1 - p$. Therefore, the expected probability of success is

$$P_s(n) = \sum_{x_0=0}^n P_0(x_0) \cdot P_{bb}(x_0, n), \quad (2)$$

where $P_0(x_0) = \binom{n}{x_0} \left(\frac{1}{\chi^k}\right)^{x_0} \left(1 - \frac{1}{\chi^k}\right)^{n-x_0}$ is the probability of having exactly x_0 correct BBs in the initial population, and $\chi = |\Sigma|$ is the cardinality of Σ .

The GR model makes several assumptions, but it has been shown that it accurately predicts the solution quality of artificial and real-world problems [7, 15]. For details, the reader is referred to the paper by Harik et al. [7], but one assumption affects the experiments in this paper: Having absorbing walls bounding the random walk implicitly assumes that mutation and crossover do not create or destroy BBs. The only source of BBs is the random initialization of the population. This is why the experiments described below do not use mutation.

4 Multiple Small Runs

We measure the quality, Q , of the solution as the number of partitions that converge to the correct BBs. The probability that one partition converges correctly is given by the GR model, $P_s(n)$ (Equation 2). For convenience, we use $P_1 = P_s(n_1)$ to denote the probability that a partition converges correctly in one run with population size n_1 and $P_r = P_s(n_r)$ for the probability that a partition converges correctly in one of the multiple runs with a population size n_r .

4.1 Solution Quality

Under the assumption that the m partitions are independent, the quality has a binomial distribution with parameters m and $P_s(n)$. Therefore, the expected solution quality of a single run is $E(Q) = mP_s(n)$. Of course, some runs will reach better solutions than others, and when we use multiple runs we consider that the problem is solved when one of them finds a solution of the desired quality. Let $Q_{r:r}$ denote the quality of the best solution found by r runs of size n_r . We are interested in its expected value, which can be calculated as [16]

$$E(Q_{r:r}) = \sum_{x=0}^{m-1} 1 - F^r(x), \quad (3)$$

where $F(x) = P(Q \leq x) = \sum_{j=0}^x \binom{m}{j} P_r^j (1 - P_r)^{m-j}$ is the cumulative distribution function of the solution quality. Unfortunately, there is no closed-form expression for the means of maximal order statistics of binomial distributions. However, there are approximations for the extreme order statistics of the Gaussian distribution, and we can use them to make some progress in our analysis. We can approximate the binomial distribution of the quality with a Gaussian, and normalize the number of correct partitions by subtracting the mean and dividing by the standard deviation: $Z_{r:r} = \frac{Q_{r:r} - mP_r}{\sqrt{mP_r(1-P_r)}}$. Let $\mu_{r:r} = E(Z_{r:r})$ denote the expected value of $Z_{r:r}$. We can approximate the expected value of the best quality in r runs as

$$E(Q_{r:r}) \approx mP_r + \mu_{r:r} \sqrt{mP_r(1-P_r)}. \quad (4)$$

If there are no restrictions on the total cost, adding more runs to an experiment results in a higher quality. The problem is that $\mu_{r:r}$ increases very slowly as more runs are used: $\mu_{r:r} \approx \sqrt{\sqrt{2} \ln r}$. Therefore, the increase in quality is marginal, and multiple isolated runs seem unappealing [20].

However, the situation may be different if the total cost is constrained. Equation 4 shows an interesting tradeoff: $\mu_{r:r}$ grows as r increases, but P_r decreases because the population size per run must decrease to keep the cost constant. Multiple runs would perform better than a single one if the quality degradation is not too pronounced. In fact, the tradeoff suggests that there is an optimal number of runs and population size that maximize the expected quality. Unfortunately, we cannot obtain a closed-form expression for these optimal parameters.

The quality reached by multiple runs is better than one run if

$$mP_r + \mu_{r:r} \sigma_r > mP_1, \quad (5)$$

where $\sigma_r = \sqrt{mP_r(1-P_r)}$. We can bound the standard deviation as $\sigma_r = 0.5\sqrt{m}$ to obtain an upper bound on the quality of the multiple runs. Substituting this bound into the inequality above, dividing by m , and rearranging we obtain

$$\frac{\mu_{r:r}}{2\sqrt{m}} > P_1 - P_r. \quad (6)$$

This equation shows that multiple runs are more likely to be beneficial on short problems (small m), everything else being equal. This is bad news for the case of multiple runs, because interesting problems in practice may be very long.

The equation above also shows that multiple runs can be advantageous if the difference between the solution qualities is small. This may happen at very small population sizes where the quality is very poor, even for a single run. This case is not very interesting, because normally we want to find high-quality solutions. However, the difference is also small when the quality does not improve much after a critical population size. This is the case that Nakano et al. [8] examined, and represents an interesting possibility where multiple runs can be beneficial. The optimum population size is probably near the point where there is no further improvement: Using a larger population would be a waste of resources, which would be better used in multiple runs to increase the chance of success.

4.2 Models of Convergence Time

We can write the fixed number of function evaluations that are available as

$$T = rgn_r, \quad (7)$$

where g is the domain-dependent number of generations until the population converges to a unique value, r is the number of independent runs, and n_r is the population size of each run. GAs are often stopped after a fixed number of generations, with the assumption that they have converged by then. In the remainder we assume that the generations until convergence are constant. Therefore, to maintain a fixed total cost, the population size of each of the multiple runs must be $n_r = n_1/r$, where n_1 denotes the population size that a single run would use.

Assuming that g is constant may be an oversimplification, since it has been shown that the convergence time depends on factors such as the population size and the selection intensity, I . For example, under some conditions, the generations until convergence are given by $g \approx \frac{\pi}{2} \frac{\sqrt{n}}{I}$ [17]. In general, if the generations until convergence are given by the power-law model $g = \kappa n^\theta$, the population size of each of the multiple runs would have to be $n_r = n_1/r^{1/(\theta+1)}$ to keep the total cost constant (e.g., in the previous equation, $\theta = 1/2$ and n_r would be $n_1/r^{2/3}$). This form of n_r would give an advantage to the multiple runs, because their sizes (and the quality of their solutions) would not decrease as much as with the constant g assumption, so this assumption is a conservative one.

4.3 Random Search

Using all the available computation time in one run with a large population is clearly one extreme. The other extreme are multiple runs with the smallest population, which is one individual. The latter case is equivalent to random search, because there is no evolution possible (we are assuming no mutation). The models above account for the two extreme cases. When the population size is one, $P_r = \frac{1}{\chi^k}$, because only one term in equation 2 is different from zero. The quality of the best solution found by r runs of size one can be calculated with equation 3.¹

To identify when random search can outperform a GA, we calculated the expected solution quality using equation 3 varying the order of the BBs, k , and the number of runs. The next section will define the functions used in these calculations; for now we only need to know that k varied. Figure 1 shows the ratio of the quality obtained by random search over the quality found by a simple GA with a population size of $n_1 = r$. Values over 1 indicate that multiple runs perform better. The figure shows that random search has an advantage as the problems become harder (with longer BBs). However, this peculiar behavior occurs only at extremely low population sizes, where the solution quality is so low

¹ Taking $Q_{r:r} = m[1 - (1 - \frac{1}{\chi^k})^r]$ may seem tempting, but it greatly overestimates the true quality. This calculation implicitly assumes that the final solution is formed by correct BBs that may have been obtained in different runs.

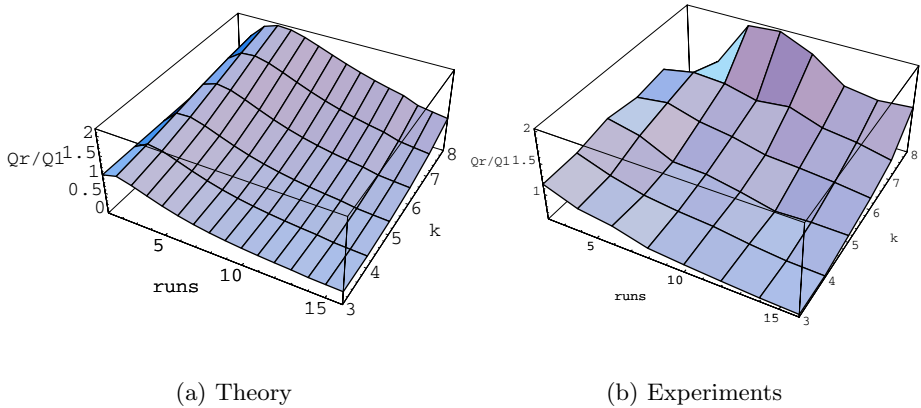


Fig. 1. Ratio of the quality of multiple runs of size 1 (random search) vs. a single run varying the order of the BBs and the number of runs.

that it is of no practical importance. When we increase the population size (and the number of random search trials), the GA moves ahead of random search.

These results suggest that superlinear speedups can be obtained if random trials are executed in parallel and the simple GA is used as the base case. Interestingly, Shonkwiler [2] used very small population sizes (≈ 2 individuals) and at least two of his functions are easily solvable by random search.

5 Experiments

The GA in the experiments used pairwise tournament selection without replacement, one-point crossover with probability 1, and no mutation. All the results presented in this section are the average of 200 trials.

The first function is the one-max function with a length of $m = 25$ bits. We varied the population size n_r from 2 to 50 individuals. For each population size, we varied the number of runs from 1 to 8 and recorded the quality of the best solution found in any of the runs, $Q_{r:r}$. Figure 2 shows the ratio of $Q_{r:r}$ over the quality Q_1 that a GA with a population size $n_1 = rn_r$ reached. The experiments match the predictions well, and in all cases the larger single runs reached solutions of better quality than the multiple smaller runs.

To illustrate that multiple runs are more beneficial when m is small, we conducted experiments varying the length of the problem to $m = 100$ and $m = 400$ bits. The population size per run was fixed at $n_r = 10$, and the number of runs varied from 1 to 8. The results in figure 3 clearly show that as the problems become longer, the single large runs find better solutions than the multiple runs.

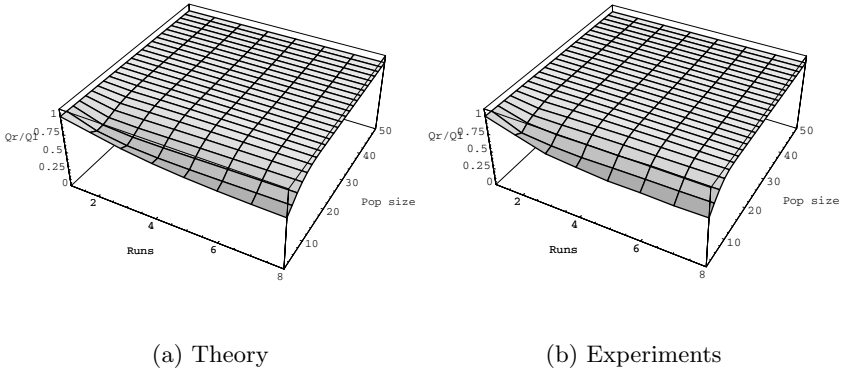


Fig. 2. Ratio of the quality of multiple runs vs. a single run for the one-max with $m = 25$ bits.

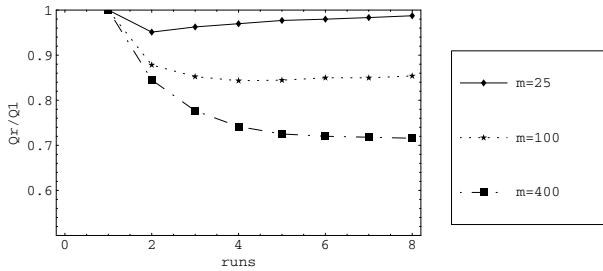


Fig. 3. Ratio of the quality of multiple runs vs. a single run varying the problem size.

The next two test functions are formed by adding fully-deceptive trap functions [18]. The order- k traps are defined as

$$f_{\text{dec}}^{(k)}(u) = \begin{cases} k - u - 1 & \text{if } u < k, \\ k & \text{if } u = k. \end{cases} \quad (8)$$

Two deceptive test function were formed by concatenating $m = 25$ copies of $f_{\text{dec}}^{(3)}$ and $f_{\text{dec}}^{(4)}$. Figures 4 and 5 show the ratio $Q_{r:r}/Q_1$, varying the run size from 2 to 100 individuals and the number of runs from one to eight. The experimental results are very close to the predictions, except with very small population sizes, where the GR model is inaccurate. In most cases, the ratio is less than one, indicating that a single large run reaches a solution with better quality than multiple small runs. The exceptions occur at very small population sizes, where even random search performs better.

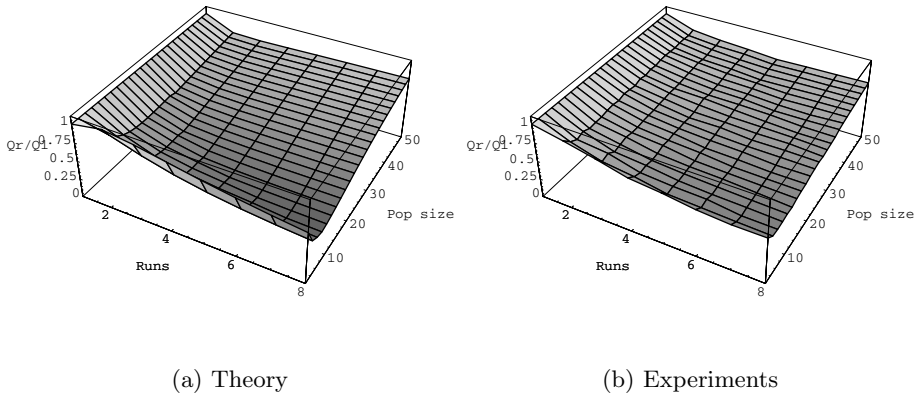


Fig. 4. Ratio of the quality of multiple runs vs. a single run for the order-3 trap.

We performed experiments to validate the results about random search. Figure 1b shows the ratio of the quality of the solutions found by the best of r random trials and the solution obtained by a GA with a population size of r . For each value of k from 3 to 8, the test functions were formed by concatenating $m = 25$ order- k trap functions. The experiments show the same general tendency as the predictions (figure 1a).

6 Multiple Short Runs

Until now we have examined the solution quality under the constant cost constraint and after the population converges to a unique solution. However, in practice it is common to stop a GA run as soon as it finds a solution that meets some quality criterion. The framework introduced in this paper could be applied to this type of experiment, if we had a model that predicted the solution quality as a function of time: $P_s(n, t)$. In any generation (or any other suitable time step), the expected solution quality in one run would be $mP_s(n, t)$, but again we would be interested in the expected value of the best solution in the r runs, which can be found by substituting the appropriate distribution in equation 3.

There are existing models of quality as a function of time, but they assume that the population is sized such that the GA will reach the global solution and that recombination of BBs is perfect [17]. If we adopt these assumptions, we could use the existing models, but we would not be able to reduce the population size to respect the constraint of fixed cost. Mühlenbein and Schlierkamp-Voosen [17] derived the following expression for the one-max function:

$$P_s(n, t) = \frac{1}{2} \left(1 + \sin\left(\frac{I}{\sqrt{n}}t\right) \right), \quad (9)$$

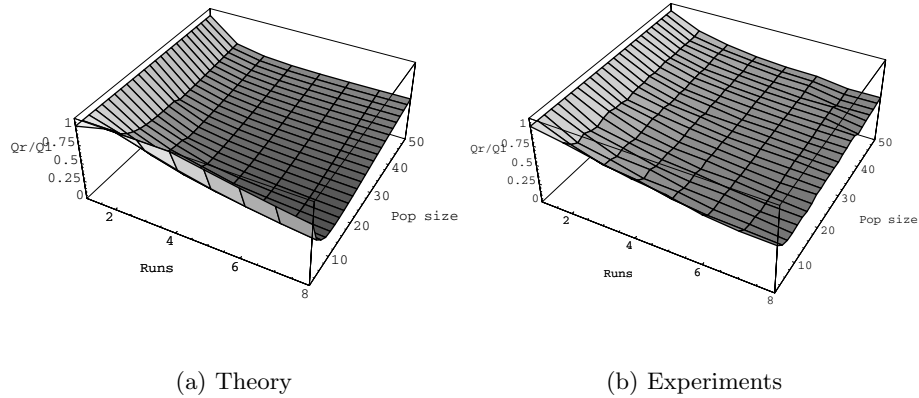


Fig. 5. Ratio of the quality of multiple runs vs. a single run for the order-4 trap.

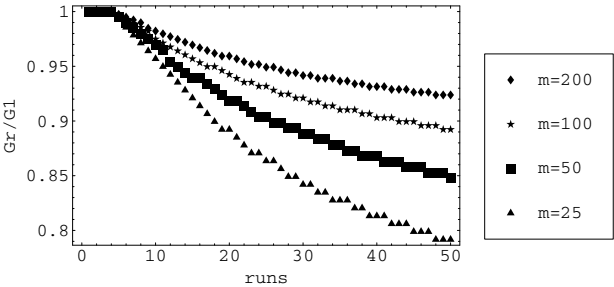


Fig. 6. Ratio of the generations until convergence of multiple over single runs. The total cost is not constant.

and Miller and Goldberg [19] used it successfully to predict the quality of deceptive functions. If we abandon the cost constraint, we can show that the best of multiple runs of the same size (that is at least large enough to reach the global optimum) reaches the solution in fewer generations than a single run of the same size. This argument has been used in the past to support the use of multiple parallel runs [2].

Figure 6 shows the ratio of the number of generations until convergence (to the global) of multiple runs over the number of generations of convergence of a single run. The figure shows that the time decreases as more runs are used, and the advantage is more pronounced for shorter problems. If each run was executed concurrently on a different processor of a parallel machine, the elapsed time to reach the solution would be reduced (assuming that the cost to determine convergence by any run is negligible, which may not be the case). However, this

scheme offers a relatively small advantage, and it is probably not the best use of multiple processors since we can obtain almost linear speedups in other ways [20].

7 Summary and Conclusions

There are conflicting reports of the advantage of using one or multiple independent runs. This problem has consequences on parallel GAs with isolated populations and also to determine when random search can outperform a GA. This paper presented an analytical study that considered additively-separable functions. Under a constraint of fixed cost and assuming no mutation, the analysis showed that the expected quality of the solution reached by multiple independent small runs is higher than the quality reached by a single large run only in very limited conditions. In particular, multiple runs seem advantageous at very small population sizes, which result in solutions of poor quality, and close to a saturation point where the solution quality does not improve with increasingly larger populations. In addition, the greatest advantage of multiple independent runs is on short problems, and the advantage tends to decrease with higher BB order. The results suggest that for difficult problems (long and with high-order BBs), the best alternative is to use a single run with the largest population possible. Small independent runs should be avoided.

Acknowledgments. We would like to thank Hillol Kargupta, Jeffrey Horn, and Georges Harik for many interesting discussions on this topic. UCRL-JC-142172. This work was performed under the auspices of the U.S. Department of Energy by University of California Lawrence Livermore National Laboratory under contract no. W-7405-Eng-48. Portions of this work were sponsored by the Air Force Office of Scientific Research, Air Force Materiel Command, USAF, under grant F49620-00-0163. Research funding for this work was also provided by the National Science Foundation under grant DMI-9908252.

References

1. Tanese, R.: Distributed genetic algorithms. In Schaffer, J.D., ed.: *Proceedings of the Third International Conference on Genetic Algorithms*, Morgan Kaufmann (1989) 434–439
2. Shonkwiler, R.: Parallel genetic algorithms. In Forrest, S., ed.: *Proceedings of the Fifth International Conference on Genetic Algorithms*, Morgan Kaufmann (1993) 199–205
3. Cantú-Paz, E., Goldberg, D.E.: Modeling idealized bounding cases of parallel genetic algorithms. In Koza, J., et al., eds.: *Proceedings of the Second Annual Genetic Programming Conference*, Morgan Kaufmann (1997) 353–361
4. Fuchs, M.: Large populations are not always the best choice in genetic programming. In Banzhaf, W., et al., eds.: *Proceedings of the Genetic and Evolutionary Computation Conference*, Morgan Kaufmann (1999) 1033–1038

5. Fernández, F., Tomassini, M., Punch, W., Sánchez, J.M.: Experimental study of isolated multipopulation genetic programming. In Whitley, D., et al., eds.: *Proceedings of the Genetic and Evolutionary Computation Conference*, Morgan Kaufmann (2000) 536
6. Wolpert, D., Macready, W.: No-free-lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation* **1** (1997) 67–82
7. Harik, G., Cantú-Paz, E., Goldberg, D., Miller, B.L.: The gambler's ruin problem, genetic algorithms, and the sizing of populations. *Evolutionary Computation* **7** (1999) 231–253
8. Nakano, R., Davidor, Y., Yamada, T.: Optimal population size under constant computation cost. In Davidor, Y., Schwefel, H.P., Männer, R., eds.: *Parallel Problem Solving from Nature, PPSN III*, Berlin, Springer-Verlag (1994) 130–138
9. Luke, S.: When short runs beat long runs. In Spector, L. et al., eds.: *Proceedings of the Genetic and Evolutionary Computation Conference*, Morgan Kaufmann (2001) 74–80
10. Mitchell, M., Holland, J.H., Forrest, S.: When will a genetic algorithm outperform hill climbing? In *Advances in Neural Information Processing Systems* 6 (1994) 51–58
11. Baum, E., Boneh, D., Garrett, C.: Where genetic algorithms excel. *Evolutionary Computation* **9** (2001) 93–124
12. Goldberg, D.E.: *Genetic algorithms in search, optimization, and machine learning*. Addison-Wesley, Reading, MA (1989)
13. Goldberg, D.E., Deb, K., Clark, J.H.: Genetic algorithms, noise, and the sizing of populations. *Complex Systems* **6** (1992) 333–362
14. Feller, W.: *An Introduction to probability theory and its applications*. 2nd edn. Volume 1. John Wiley and Sons, New York, NY (1966)
15. van Dijk, S., Thierens, D., de Berg, M.: Scalability and efficiency of genetic algorithms for geometrical applications. In Schoenauer, M., et al., eds.: *Parallel Problem Solving from Nature—PPSN VI*, Berlin, Springer-Verlag (2000) 683–692
16. Arnold, B., Balakrishnan, N., Nagaraja, H.N.: *A first course in order statistics*. John Wiley and Sons, New York, NY (1992)
17. Mühlenbein, H., Schlierkamp-Voosen, D.: Predictive models for the breeder genetic algorithm: I. Continuous parameter optimization. *Evolutionary Computation* **1** (1993) 25–49
18. Deb, K., Goldberg, D.E.: Analyzing deception in trap functions. In Whitley, L.D., ed.: *Foundations of Genetic Algorithms 2*, Morgan Kaufmann (1993) 93–108
19. Miller, B.L., Goldberg, D.E.: Genetic algorithms, selection schemes, and the varying effects of noise. *Evolutionary Computation* **4** (1996) 113–131
20. Cantú-Paz, E.: *Efficient and Accurate Parallel Genetic Algorithms*. Kluwer Academic Publishers, Boston, MA (2000)

Constrained Multi-objective Optimization Using Steady State Genetic Algorithms

Deepti Chafekar, Jiang Xuan, and Khaled Rasheed

Computer Science Department
University of Georgia
Athens, GA 30602 USA
{chafekar, xuan, khaled}@cs.uga.edu

Abstract. In this paper we propose two novel approaches for solving constrained multi-objective optimization problems using steady state GAs. These methods are intended for solving real-world application problems that have many constraints and very small feasible regions. One method called Objective Exchange Genetic Algorithm for Design Optimization (OEGADO) runs several GAs concurrently with each GA optimizing one objective and exchanging information about its objective with the others. The other method called Objective Switching Genetic Algorithm for Design Optimization (OSGADO) runs each objective sequentially with a common population for all objectives. Empirical results in benchmark and engineering design domains are presented. A comparison between our methods and Non-Dominated Sorting Genetic Algorithm-II (NSGA-II) shows that our methods performed better than NSGA-II for difficult problems and found Pareto-optimal solutions in fewer objective evaluations. The results suggest that our methods are better applicable for solving real-world application problems wherein the objective computation time is large.

1 Introduction

This paper concerns the application of steady state Genetic Algorithms (GAs) in realistic engineering design domains which usually involve simultaneous optimization of multiple and conflicting objectives with many constraints. In these problems instead of a single optimum there usually exists a set of trade-off solutions called the non-dominated solutions or Pareto-optimal solutions. For such solutions no improvement in any objective is possible without sacrificing at least one of the other objectives. No other solutions in the search space are superior to these Pareto-optimal solutions when all objectives are considered. The user is then responsible for choosing a particular solution from the Pareto-optimal set later.

Some of the challenges faced in the application of GAs to engineering design domains are:

- The search space can be very complex with many constraints and the feasible (physically realizable) region in the search space can be very small.
- Determining the quality (fitness) of each point may involve the use of a simulator or an analysis code which takes a non-negligible amount of time. This simulation time can range from a fraction of a second to several days in some cases. Therefore it is impossible to be cavalier with the number of objective evaluations in an optimization.

For such problems steady state GAs may perform better than generational GAs because they better retain the feasible points found in their populations and may have higher selection pressure which is desirable when evaluations are very expensive. With good diversity maintenance, steady state GAs have done very well in several realistic domains [1]. Significant research has yet to be done in the area of steady state multi-objective GAs. We therefore decided to focus our research on this area.

The area of multi-objective optimization using Evolutionary Algorithms (EAs) has been explored for a long time. The first multi-objective GA implementation called the Vector Evaluated Genetic Algorithm (VEGA) was proposed by Schaffer in 1985 [9]. Since then, many Evolutionary algorithms for solving multi-objective optimization problems have been developed. The most recent ones are the Non-Dominated Sorting Genetic Algorithm-II (NSGA-II) [3], Strength Pareto Evolutionary Algorithm-II (SPEA-II) [16], Pareto Envelope based selection-II (PESA-II) [17]. Most of these approaches propose the use of a generational GA. Deb proposed an Elitist Steady State Multi-objective Evolutionary Algorithm (MOEA) [18] which attempts to maintain spread [15] while attempting to converge to the true Pareto-optimal front. This algorithm requires sorting of the population for every new solution formed thereby increasing its time complexity. Very high time complexity makes the Elitist steady state MOEA impractical for some problems. To the best of our knowledge, apart from Elitist Steady State MOEA, the area of steady state multi-objective GAs has not been widely explored. Also constrained multi-objective optimization which is very important for real-world application problems has not received the deserved exposure. In this paper we propose two methods for solving constrained multi-objective optimization using steady state GAs. These methods are relatively fast and practical. It is also easy to transform a single-objective GA to a multi-objective GA by using these methods.

In the first method called the Objective Exchange Genetic Algorithm for Design Optimization (OEGADO) several single objective GAs run concurrently. Each GA optimizes one of the objectives. At certain intervals these GAs exchange information about their respective objectives with each other. In the second method called the Objective Switching Genetic Algorithm for Design Optimization (OSGADO) a single GA runs multiple objectives in a sequence switching at certain intervals between objectives.

Our methods can be viewed as multi-objective transformations of GADO (Genetic Algorithm for Design Optimization) [1, 2]. GADO is a GA that was designed with the goal of being suitable for the use in engineering design. It uses new operators and

search control strategies that target engineering domains. GADO has been applied in a variety of optimization tasks which span many fields. It has demonstrated a great deal of robustness and efficiency relative to competing methods.

In GADO, each individual in the GA population represents a parametric description of an artifact. All parameters have continuous intervals. The fitness of each individual is based on the sum of a proper measure of merit computed by a simulator or some analysis code, and a penalty function if relevant. A steady state model is used, in which several crossover and mutation operators including specific and innovative operators like guided crossover are applied to two parents selected by linear rank based selection. The replacement strategy used is a crowding technique, which takes into consideration both the fitness and the proximity of the points in the GA population. GADO monitors the degree of diversity of the GA population. If at any stage it is discovered that the individuals in the population became very similar to one another, the diversity maintenance module rebuilds the population using previously evaluated points in a way that restores diversity. The diversity maintenance module in GADO also rejects proposed points that are extremely similar to previously evaluated points. The GA stops when either the maximum number of evaluations has been exhausted or the population loses diversity and practically converges to a single point in the search space. Floating point representation is used. GADO also uses some search control strategies [2] such as a screening module which saves time by avoiding the full evaluation of points that are unlikely to correspond to good designs.

We compared the results of our two methods with the state-of-the-art Elitist Non-Dominated Sorting Algorithm-II (NSGA-II) [3]. NSGA-II is a non-dominated sorting based multi-objective evolutionary algorithm with a computational complexity of $O(MN^2)$ (where M is the number of objectives and N is the population size). NSGA-II incorporates an elitist approach, a parameter-less niching approach and a simple constraint handling strategy. Due to NSGA-II's low computational requirements, elitist features and constraint handling capacity, it has been successfully used in many applications. It proved to be better than many other multi-objective optimization GAs [3, 18].

In the remainder of the paper, we provide a brief description of our two proposed methods. We then present results of the comparison of our methods with NSGA-II. Finally, we conclude the paper with a discussion of the results and future work.

2 Methods for Multi-objective Optimization Using Steady State GAs

We propose two methods for solving constrained multi-objective optimization problems using steady state GAs. One is the Objective Exchange Genetic Algorithm for Design Optimization (OEGADO), and other is the Objective Switching Genetic Algorithm for Design Optimization (OSGADO). It should be noted that for multi-objective GAs, maintaining diversity is a key issue. However we did not need to take

any extra measures for diversity maintenance as the diversity maintenance module already present in GADO [1, 2] seemed to handle this issue effectively. We focused on the case of two objectives in our experiments for simplicity of implementation and readability of the results, but the methods are applicable for multi-objective optimization problems with more than two objectives.

2.1 Objective Exchange Genetic Algorithm for Design Optimization (OEGADO)

The main idea of OEGADO is to run several single objective GAs concurrently. Each of the GAs optimizes one of the objectives. All the GAs share the same representation and constraints, but have independent populations. They exchange information about their respective objectives every certain number of iterations.

In our implementation, we have used the idea of informed operators (IOs) [4]. The main idea of the IOs is to replace pure randomness in traditional GA operators with decisions that are guided by reduced models formed using the methods presented in [5, 6, 7]. The reduced models are approximations of the fitness function, formed using some approximation techniques, such as least squares approximation [5, 7, 8]. These functional approximations are then used to make the GA operators such as crossover and mutation more informed. These IOs generate multiple children [4], rank them using the approximate fitness obtained from the reduced model and select the best.

Every single objective GA in OEGADO uses least squares to form a reduced model of its own objective. Every GA exchanges its own reduced model with those of the other GAs. In effect, every GA, instead of using its own reduced model, uses other GAs' reduced models to compute the approximate fitness of potential individuals. Therefore each GA is informed about other GAs' objectives. As a result each GA not only focuses on its own objective, but also gets biased towards the objectives which the other GAs are optimizing.

The OEGADO algorithm for two objectives looks as follows:

1. Both the GAs are run concurrently for the same number of iterations, each GA optimizes one of the two objectives while also forming a reduced model of it.
2. At intervals equal to twice the population size, each GA exchanges its reduced model with the other GA.
3. The conventional GA operators such as initialization (only applied in the beginning), mutation and crossover are replaced by informed operators. The IOs generate multiple children and use the reduced model to compute the approximate fitness of these children. The best individual based on this approximate fitness is selected to be the newborn. It should be noted that the approximate fitness function used is of the other objective.
4. The true fitness function is then called to evaluate the actual fitness of the newborn corresponding to the current objective.
5. The individual is then added to the population using the replacement strategy.

6. Steps 2 through 5 are repeated till the maximum number of evaluations is exhausted.

If all objectives have similar computational complexity, the concurrent GAs can be synchronized, so that they exchange the current approximations at the right time. On the other hand, when objectives vary considerably in their time complexity, the GAs can be run asynchronously.

It should be noted that OEGADO is not really a multi-objective GA, but several single objective GAs working concurrently to get the Pareto-optimal region. Each GA finds its own feasible region, by evaluating its own objective. For the feasible points found by a single GA, we need to run the simulator to evaluate the remaining objectives. Thus for OEGADO with two objectives:

Total number of objective evaluations = Sum of objective evaluations of each GA + Sum of the number of feasible points found by each GA

A potential advantage of this method is speed, as the concurrent GAs can run in parallel. Therefore multiple objectives can be evaluated at the same time on different CPUs. Also the asynchronous OEGADO works better for objectives having different time complexities. If some objectives are fast, they are not slowed down by the slower objectives. It should be noted that because of the exchange of reduced models, each GA optimizes its own objective and also gives credit to the other objectives.

2.2 Objective Switching Genetic Algorithm for Design Optimization (OSGADO)

The main idea of OSGADO is to use a single GA that optimizes multiple objectives in a sequential order. Every objective is optimized for a certain number of evaluations, then a switch occurs and the next objective is optimized. The population is not changed when objectives are switched. This continues till the maximum number of evaluations is complete.

We modified GADO [1, 2] to create multi-objective OSGADO. OSGADO is inspired from the Vector Evaluated GA (VEGA) [9]. Schaffer (1985) proposed VEGA for generational GAs. In VEGA the population is divided into m different parts for m diff objectives; part i is filled with individuals that are chosen at random from current population according to objective i . Afterwards the mating pool is shuffled and crossover and mutation are performed as usual. Though VEGA gave encouraging results, it suffered from bias towards the extreme regions of the Pareto-optimal curve.

The OSGADO algorithm looks as follows:

1. The GA is run initially with the first objective as the measure of merit for a certain number of evaluations. The fitness of an individual is calculated based on its measure of merit and the constraint violations. Selection, crossover and mutation take place in the regular manner.

2. After a certain numbers of evaluations, the GA is run for the next objective. When the evaluations for the last objective are complete, the GA switches back to the first objective.
3. Step 2 is repeated till the maximum number of evaluations is reached.

In order to fairly compare the methods, in the experiments we first ran OEGADO and obtained the number of feasible points found by each of the two GAs. We then ran OSGADO for the number of evaluations calculated as follows,

Total number of objective evaluations = Sum of evaluations of each objective in OEGADO + Sum of the number of feasible points found by each objective in OEGADO

OSGADO has certain advantages over VEGA. In VEGA every solution is evaluated for only one of the objectives each time and therefore it can converge to individual objective optima (the extremes of the Pareto-optimal curve) without adequately sampling the middle section of the Pareto-optimal curve. However OSGADO evaluates every solution using each of the objectives at different times. So OSGADO is at less risk of converging at individual objective optima.

3 Experimental Results

In this section, we first describe the test problems used to compare the performance of OEGADO, OSGADO and NSGA-II. We then briefly discuss the parameter settings used. Finally, we discuss the results obtained for various test cases by these three methods.

3.1 Test Problems

The test problems for evaluating the performance of our methods were chosen based on significant past studies. We chose four problems from the benchmark domains commonly used in past multi-objective GA research, and two problems from the engineering domains. The degree of difficulty of these problems varies from fairly simple to difficult.

The problems chosen from the benchmark domains are BNH used by Binh and Korn [10], SRN used by Srinivas, Deb [11], TNK suggested by Tanaka [12] and OSY used by Osyczka, Kundu [13]. The problems chosen from the engineering domains are Two-Bar Truss Design used by Deb [14] and Welded Beam design used by Deb [14]. All these problems are constrained multi-objective problems. Table 1 shows the variable bounds, objective functions and constraints for all these problems.

Table 1. Test problems used in this study, all objective functions are to be minimized

Problem	Variable bounds	Objectives functions $f(x)$ and Constraints $C(x)$
BNH	$x_1 \in [0,5]$ $x_2 \in [0,3]$	$f_1(x) = 4x_1^2 + 4x_2^2$ $f_2(x) = (x_1 - 5)^2 + (x_2 - 5)^2$ $C_1(x) \equiv (x_1 - 5)^2 + x_2^2 \leq 25$ $C_2(x) \equiv (x_1 - 8)^2 + (x_2 + 3)^2 \geq 7.7$
SRN	$x_1 \in [-20,20]$ $x_2 \in [-20,20]$	$f_1(x) = 2 + (x_1 - 2)^2 + (x_2 - 2)^2$ $f_2(x) = 9x_1 - (x_2 - 1)^2$ $C_1(x) \equiv x_1^2 + x_2^2 \leq 225$ $C_2(x) \equiv x_1 - 3x_2 + 10 \leq 0$
TNK	$x_1 \in [0,\pi]$ $x_2 \in [0,\pi]$	$f_1(x) = x_1$ $f_2(x) = x_2$ $C_1(x) \equiv x_1^2 + x_2^2 - 1 - 0.1\cos(16\arctan \frac{x_1}{x_2}) \geq 0$ $C_2(x) \equiv (x_1 - 0.5)^2 + (x_2 - 0.5)^2 \leq 0.5$
OSY	$x_1 \in [0,10]$ $x_2 \in [0,10]$ $x_3 \in [1,5]$ $x_4 \in [0,6]$ $x_5 \in [1,5]$ $x_6 \in [0,10]$	$f_1(x) = -[25(x_1 - 2)^2 + (x_2 - 2)^2 + (x_3 - 1)^2 + (x_4 - 4)^2 + (x_5 - 1)^2]$ $f_2(x) = x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_5^2 + x_6^2$ $C_1(x) \equiv x_1 + x_2 - 2 \geq 0$ $C_2(x) \equiv 6 - x_1 - x_2 \geq 0$ $C_3(x) \equiv 2 - x_2 + x_1 \geq 0$ $C_4(x) \equiv 2 - x_1 + 3x_2 \geq 0$ $C_5(x) \equiv 4 - (x_3 - 3)^2 - x_4 \geq 0$ $C_6(x) \equiv (x_5 - 3)^2 + x_6 - 4 \geq 0$
Two-bar Truss Design	$x_1 \in [0,0.01]$ $x_2 \in [0,0.01]$ $x_3 \in [1,3]$	$f_1(x) = x_1\sqrt{16 + x_3^2} + x_2\sqrt{1 + x_3^2}$ $f_2(x) = \max(\sigma_1, \sigma_2)$ $C_1(x) \equiv \max(\sigma_1, \sigma_2) \leq 10^5$ $\sigma_1 = 20\sqrt{16 + x_3^2} / x_1x_3$ $\sigma_2 = 80\sqrt{1 + x_3^2} / x_2x_3$

Welded Beam Design	$h \in [0.125, 5]$ $b \in [0.125, 5]$ $l \in [0.1, 10]$ $t \in [0.1, 10]$	$f_1(x) = 1.10471h^2l + 0.04811tb(14+l)$ $f_2(x) = 2.1952/t^3b$ $C_1(x) \equiv 13600 - \tau(x) \geq 0$ $C_2(x) \equiv 30000 - \sigma(x) \geq 0$ $C_3(x) \equiv b - h \geq 0$ $C_4(x) \equiv P_c(x) - 6000 \geq 0$ $\tau = \sqrt{(\tau')^2 + (\tau'')^2} + l\tau'\tau'' / \sqrt{0.25(l^2 + (h+t)^2)}$ $\tau' = 6000 / \sqrt{2hl}$ $\tau'' = \frac{6000(14 + 0.5l)\sqrt{0.25(l^2 + (h+t)^2)}}{2\sqrt{2hl}(l^2/12 + 0.25(h+t)^2)}$ $\sigma = 504000/t^2b$ $P_c = 64746.022(1 - 0.0282346t)tb^3$

3.2 Parameter Settings

Each optimization run was carried out with similar parameter settings for all the methods. The following are the parameters for the three GAs.

Let $ndim$ be equal to the number of dimensions of the problems.

1. Population size: For OEGADO and OSGADO the population size was set to $10*ndim$. For NSGA-II the population size was fixed to 100 as recommended in [19].
2. Number of objective evaluations: Since the three methods work differently the number of objective evaluations is computed differently. The number of objective evaluations for OEGADO and OSGADO according to Section 2.1 and 2.2 is given as *Objective evaluations for OEGADO and OSGADO* = $2*500*ndim + \text{sum of feasible points found by each GA in OEGADO model}$

NSGA-II is a generational GA, therefore for a two-objective NSGA-II:

Total number of objective evaluations = $2*population\ size * number\ of\ generations$

Since we did not know exactly how many evaluations would be required by OEGADO before hand, to give fair treatment to NSGA-II, we set the number of generations of NSGA-II to be $10*ndim$. In effect NSGA-II ended up doing significantly more evaluations than OEGADO and OSGADO for some problems. We however did not decrease the number of generations for NSGA-II and repeat the experiments as our methods outperformed it in most domains anyway.

3.3 Results

In the following section, Figures 1-4 present the graphical results of all three methods in the order of OEGADO, OSGADO and NSGA-II for all problems. The outcomes of five runs using different seeds were unified and then the non-dominated solutions were selected and plotted from the union set for each method. We are using graphical

representations of the Pareto-optimal curve found by the three methods to compare their performance.

It is worth mentioning that the number of Pareto-optimal solutions obtained by NSGA-II is limited by its population size. Our methods keep track of all the feasible solutions found during the optimization and therefore do not have any restrictions on the number of Pareto-optimal solutions found.

The BNH and the SRN (figures not shown) problems are fairly simple in that the constraints may not introduce additional difficulty in finding the Pareto-optimal solutions. It was observed that all three methods performed equally well within comparable number of objective evaluations (mentioned in Section 3.2), and gave a dense sampling of solutions along the true Pareto-optimal curve.

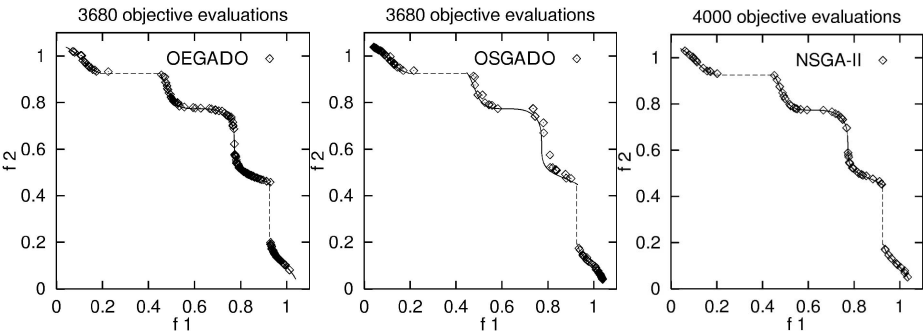


Fig. 1. Results for the benchmark problem TNK

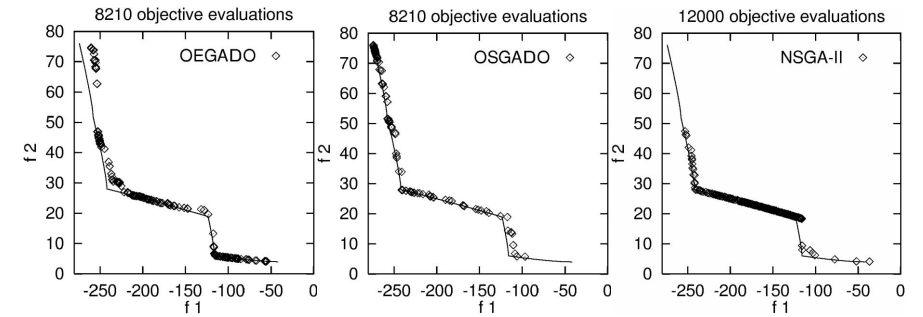


Fig. 2. Results for the benchmark problem OSY

The TNK problem (Fig. 1) and the OSY problem (Fig. 2) are relatively difficult. The constraints in the TNK problem make the Pareto-optimal set discontinuous. The constraints in the OSY problem divide the Pareto-optimal set into five regions that can demand a GA to maintain its population at different intersections of the constraint boundaries. As it can be seen from the above graphs for the TNK problem, within comparable number of fitness evaluations, the OEGADO model and the NSGA-II model performed equally well. They both displayed a better distribution of the Pareto-

optimal points than the OSGADO model. OSGADO performed well at the extreme ends, but found very few Pareto points at the mid-section of the curve. For the OSY problem, it can be seen that OEGADO gave a good sampling of points at the mid-section of the curve and also found points at the extreme ends of the curve. OSGADO also performed well, giving better sampling at one of the extreme ends of the curve. NSGA-II however did not give a good sampling of points at the extreme ends of the Pareto-optimal curve and gave a poor distribution of the Pareto-optimal solutions. In this problem OEGADO and OSGADO outperformed NSGA-II while running for fewer objective evaluations.

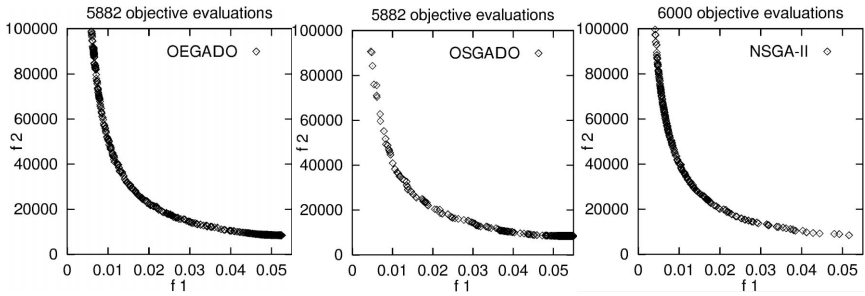


Fig. 3. Results for the Two-bar Truss design problem

For the Two-bar Truss design problem (Fig. 3), within comparable fitness evaluations, NSGA-II performed slightly better than our methods in the first objective. OEGADO showed a uniform distribution of the Pareto-optimal curve. OSGADO however gave a poor distribution at one end of the curve, but it achieved very good solutions at the other end and converged to points that the other two methods failed to reach.

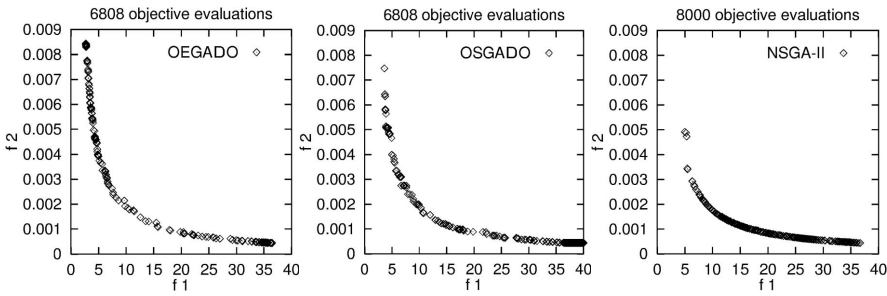


Fig. 4. Results for the Welded Beam design problem

In the Welded Beam design problem (Fig. 4), the non-linear constraints can cause difficulties in finding the Pareto solutions. As shown in Fig. 4, within comparable fitness evaluations, OEGADO outperformed OSGADO and NSGA-II in both distribution and spread [15]. OEGADO found the best minimum solution for f_1 with a value of 2.727 units. OSGADO was able to find points at the other end that the other

two methods failed to reach. NSGA-II did not achieve a good distribution of the Pareto solutions at the extreme regions of the curve.

4 Conclusion and Future Work

In this paper we presented two methods for multi-objective optimization using steady state GAs, and compared our methods with a reliable and efficient generational multi-objective GA called NSGA-II. The results show that a steady state GA can be used efficiently for constrained multi-objective optimization. For the simpler problems our methods performed equally well as NSGA-II. For the difficult problems, our methods outperformed NSGA-II in most respects. In general, our methods demonstrated robustness and efficiency in their performance. OEGADO in particular performed consistently well and outperformed the other two methods in most of the domains. Moreover, our methods were able to find the Pareto-optimal solutions for all the problems in fewer objective evaluations than NSGA-II. For real-world problems, the number of objective evaluations performed can be critical as each objective evaluation takes a long time. Based on this study we believe that our methods can outperform multi-objective generational GAs for such problems. However, we need to experiment more and find out whether there are other factors that contribute to the success of our methods other than their steady state nature.

In the future, we would like to experiment with several steady state GAs as the base method. We would also like to improve both of our methods. Currently they do not have any explicit bias towards non-dominated solutions. We therefore intend to enhance them by giving credit to non-dominated solutions. OEGADO has shown promising results and we would like to further improve it, extend its implementation to handle more than two objectives and further explore its capabilities. The current OSGADO implementation can already handle more than two objectives. We would also like to use our methods for more complex real-world applications.

Acknowledgement. This research is sponsored by the US National Science Foundation under grant CTS-0121058. The program managers are Drs. Frederica Darema, C. F. Chen and Michael Plesniak.

References

1. Khaled Rasheed. GADO: A genetic algorithm for continuous design optimization. Technical Report DCS-TR-352, Department of Computer Science, Rutgers, The State University of New Jersey, New Brunswick, NJ, January 1998. Ph.D. Thesis, <http://webster.cs.uga.edu/~khaled/thesis.ps>.
2. Khaled Rasheed and Haym Hirsh. Learning to be selective in genetic-algorithm-based design optimization. *Artificial Intelligence in Engineering, Design, Analysis and Manufacturing*, 13:157–169, 1999.
3. Deb, K., S. Agrawal, A. Pratap, and T. Meyarivan (2000). A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. In *Proceedings of the Parallel Problem Solving from Nature VI*, pp.849–858.

4. Khaled Rasheed and Haym Hirsh. Informed operators: Speeding up genetic-algorithm-based design optimization using reduced models. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2000)*, pp. 628–635, 2000.
5. K. Rasheed., S. Vattam, X. Ni. Comparison of Methods for Using Reduced Models to Speed up Design Optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2002)*, pp. 1180–1187, 2002.
6. Khaled Rasheed. An incremental-approximate-clustering approach for developing dynamic reduced models for design optimization. In *Proceedings of the Congress on Evolutionary Computation (CEC'2002)*, pp. 986–993, 2002.
7. K. Rasheed., S. Vattam, X. Ni. Comparison of methods for developing dynamic reduced models for design optimization. In *Proceedings of the Congress on Evolutionary Computation (CEC'2002)*, pp. 390–395, 2002.
8. William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes in C: the Art of Scientific Computing*. Cambridge University Press, Cambridge [England]; New York, 2nd edition, 1992.
9. J.D.Schaffer .Multi-objective optimization with vector evaluated genetic algorithms. In *Proceedings of an International Conference on Genetic Algorithms and Their Applications*, J.J. Grefenstette, Ed., Pittsburg, PA, July 24–26 1985, pp. 93–100, sponsored by Texas Instruments and U.S. Navy Center for Applied Research in Artificial Intelligence (NCARAI).
10. Binh and Korn. MOBES: A multi-objective Evolution Strategy for constrained optimization Problems. In *Proceedings of the 3rd International Conference on Genetic Algorithm MENDEL 1997*, Brno, Czech Republic, pp.176–182.
11. Srinivas, N. and Deb, K. (1995). Multi-Objective function optimization using non-dominated sorting genetic algorithms. *Evolutionary Computation* (2), 221–248.
12. Tanaka, M. (1995). GA-based decision support system for multi-criteria, optimization. In *Proceedings of the International Conference on Systems, Man and Cybernetics-2*, pp. 1556–1561.
13. Osyczka, A. and Kundu, S. (1995). A new method to solve generalized multicriteria optimization problems using the simple genetic algorithm. *Structural Optimization* (10). 94–99.
14. Deb, K. Pratap, A. and Moitra, S. (2000). Mechanical Component Design for Multiple Objectives Using Elitist Non-Dominated Sorting GA. *KanGAL Report No. 200002*.
15. Ranjithan, S.R., S.K. Chetan, and H.K. Dakshina (2001). Constraint method-based evolutionary algorithm (CMEA) for multi-objective optimization. In E.Z. et al. (Ed.), *Evolutionary Multi-Criteria Optimization 2001, Lecture Notes in Computer Science* 1993, pp. 299–313. Springer-Verlag.
16. Zitzler, E., Laumanns, M., and Thiele, L. (2001). SPEA2: Improving the Strength Pareto Evolutionary Algorithm. Technical Report 103, *Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH) Zurich*, Gloriastrasse 35, CH-8092 Zurich, Switzerland.
17. Crone, D. W., Knowles, J.D., and Oates, M.J. (2000). The Pareto Envelope-based Selection Algorithm for Multi-objective Optimization. In Schoenauer, M., Deb, K., Rudolph, G., Yao, X., Luton, E., Merelo, J.J., and Schewfel, H.-P., editors, *Proceedings of the Parallel Problem Solving from Nature VI Conference*, pp.839–848, Paris, France. Springer. Lecture Notes in Computer Science No. 1917.
18. K. Deb. Multi-objective optimization using evolutionary algorithms. Chichester, UK: John Wiley, 2001.
19. K. Deb. S. Gulati (2001). Design of truss-structures for minimum weight using genetic algorithms, In *Journal of Finite Elements in Analysis and Design*, pp.447–465, 2001.

An Analysis of a Reordering Operator with Tournament Selection on a GA-Hard Problem

Ying-ping Chen¹ and David E. Goldberg²

¹ Department of Computer Science and Department of General Engineering
University of Illinois, Urbana, IL 61801, USA
`ypchen@illigal.ge.uiuc.edu`

² Department of General Engineering
University of Illinois, Urbana, IL 61801, USA
`deg@uiuc.edu`

Abstract. This paper analyzes the performance of a genetic algorithm that utilizes tournament selection, one-point crossover, and a reordering operator. A model is proposed to describe the combined effect of the reordering operator and tournament selection, and the numerical solutions are presented as well. Pairwise, s -ary, and probabilistic tournament selection are all included in the proposed model. It is also demonstrated that the upper bound of the probability to apply the reordering operator, previously derived with proportionate selection, does not affect the performance. Therefore, tournament selection is a necessity when using a reordering operator in a genetic algorithm to handle the conditions studied in the present work.

1 Introduction

In order to ensure a genetic algorithm (GA) works well, the building blocks represented in the chromosome of the underlying problem have to be tightly linked. Otherwise, studies [1,2] have shown that a GA may fail to solve problems without such prior knowledge. Because it is difficult to guarantee that the chosen chromosome representation can provide tightly linked building blocks for processing, linkage learning operators should be adopted to overcome the difficulty, which is called the *coding trap* [3]. Currently, one way to conduct linkage learning is to use the (*gene number*, *allele*)-style coding scheme and reordering operators in a genetic algorithm. Reordering operators, including inversion [4, 5,6,7,8], order-based crossover operators [9,10,11,12,13,14,15], and so on, have already been studied for quite some time. The effectiveness of using an *idealized reordering operator* (IRO) has been demonstrated [3], but an upper bound on the probability to apply the IRO was also pointed out in the same work.

Since the introduction of the minimal deceptive problem (MDP) as a tool for genetic algorithm modeling and performance analysis [16], the MDP has been widely used and discussed. Some studies [3,17,18] tested the GA performance with their theoretical frameworks on the MDP, while others [19,20,21]

were interested in the nature and property of the MDP and tried to understand the relationship among the epistasis, deception, and difficulty for genetic algorithms. In the present work, we use the MDP with different initial conditions as our test problems in the theoretical model because of its simplicity for analysis.

Previous analysis on reordering [3] was based on a genetic algorithm including proportionate selection, one-point crossover, and idealized reordering operator. Because genetic algorithms nowadays usually do not use proportionate selection, this paper seeks the answer to whether the effectiveness of using a reordering operator with selection other than proportionate selection changes or not. In particular, we first modularize the previous model so that different selection operators can be easily plugged into the framework. Then tournament selection, including its variants, is put into the model with the idealized reordering operator on the minimal deceptive problem, and the performance of the model is displayed and analyzed.

The organization of this paper is in the following. The next section gives a brief review of the framework, which includes the test problems, our assumptions, and the previous results. Section 3 describes the modularization and extension of the theoretical model in detail and presents the numerical solutions. Finally, the conclusions and future work of this paper are presented in Sect. 4.

2 The Framework

In this section, we introduce the problem we use in this paper for research and analysis, the assumptions we make to build the theoretical model, and the previous results based on the model.

2.1 Minimal Deceptive Problem

In order to understand how a reordering operator can help a GA to solve problems, we have to use a test problem which is hard enough so that a GA cannot solve it by itself. On the other hand, the test problem should be not so complicated that we can easily have it theoretically analyzed. In this study, we employ a problem of known and controllable difficulty as our study subject.

In particular, the *minimal deceptive problem* (MDP) [16] is adopted as the test problem. The MDP is a two-bit problem and designed to mislead a GA away from the optimal solution and toward sub-optimal ones. There are two types of MDP [16] depending on whether $f_{0,1}$ is greater or less than $f_{0,0}$, where $f_{0,1}$ and $f_{0,0}$ are the fitness for point $(0, 1)$ and $(0, 0)$, respectively. Further analysis shows that the MDP Type II is more difficult than Type I because the GA cannot converge to the optimal solution if the initial population is biased toward the sub-optimal solution.

By utilizing the MDP Type II and setting the initial condition which makes a GA diverge, we conduct our analysis on the combined effect of a reordering operator and tournament selection. Figure 1 shows the MDP Type II, and in this paper, we have the following fitness values for each point:

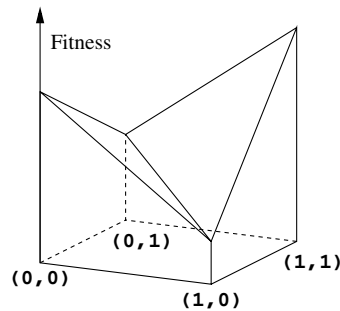


Fig. 1. The Minimal Deceptive Problem (MDP) Type II. $f_{0,0} > f_{0,1}$.

$$f_{1,1} = 1.1; \quad f_{0,0} = 1.0; \quad f_{0,1} = 0.9; \quad f_{1,0} = 0.5.$$

2.2 Assumptions

In the present paper, we study a generational genetic algorithm that combines tournament selection, one-point crossover, and a reordering operator on the MDP Type II. The following assumptions are made for simplifying the theoretical study and analysis.

First, instead of analyzing any particular reordering operator, an *idealized reordering operator* (IRO) [3] is analyzed. The IRO transfers a building block from *short* to *long* or from *long* to *short* with a *reordering probability* p_r . Here we consider the net effect produced by the IRO. The difference of a building block being short or long reflects on the effective crossover probability p_c . The longer the building block is, the more likely it will be disrupted, and vice versa.

Second, crossover events can only occur between individuals containing the building block of the identical defining length. This assumption might be untrue for actual implementations and finite populations. However, it further simplifies our analysis, makes the model more capable of displaying the transition between shorts and longs, and gives us more insights about linkage learning process.

Finally, because population portions of different schemata are considered, an infinite population is assumed implicitly as well.

2.3 Reordering and Linkage Learning

Conducting linkage learning in a GA can overcome the difficulty of the chromosome representation design when no prior knowledge about the problem structure exists. One of the straightforward methods for linkage learning is to employ the (*gene number*, *allele*)-style coding scheme and reordering operators. For an example of a five-bit problem, an individual 01101 might be represented as

$$((2, 1) (4, 0) (1, 0) (5, 1) (3, 1)) \quad \text{or} \quad ((5, 1) (4, 0) (3, 1) (2, 1) (1, 0)).$$

If we consider an order-two schema composed of gene 2 and gene 3, for the first case, the schema is 1***1, while it is **11* for the second case. The ordering of the (gene number, allele)'s does not affect the fitness value of the individual but affects the defining length of the schema and therefore the probability to disrupt the schema when processing. Thus, reordering operators can effectively change the linkage among genes during the evolutionary process in this manner, and it is the reason to study reordering operators as linkage learning operators in our present work.

2.4 Previous Results

A genetic algorithm with IRO on the MDP Type II was analyzed and compared to one without IRO [3]. The results showed that a GA without IRO might diverge under certain initial conditions, and IRO can help a GA to overcome such a difficulty. However, they also derived an upper bound on the probability p_r to apply the reordering operator

$$0 < p_r \leq \frac{(r-1)(1-P_f)}{r}, \quad (1)$$

where proportionate selection is used, r is the ratio of the fitness value of the optimal schema to that of the sub-optimal schema, and the converged population contains proportion of at least P_f optimal individuals.

Calculating the upper bound of p_r on the MDP Type II used in the paper is straightforward:

$$r = \frac{f_{1,1}}{f_{0,0}} = \frac{1.1}{1.0} = 1.1.$$

If at least 50% optimal solutions are desired in the converged population, the upper bound of p_r will be

$$p_r \leq \frac{(r-1)(1-P_f)}{r} = \frac{0.1}{1.1}(1-0.5) = 0.0455.$$

It was showed that if p_r is greater than the upper bound, the GA still diverges even with the help of IRO. Therefore, although IRO was demonstrated to be useful for helping a GA to overcome the *coding trap*, the upper bound of the reordering probability quite limits its applicability.

3 IRO with Tournament Selection

Now, we propose our theoretical model and analyze the combined effect of IRO and tournament selection. We start from the model developed based on using proportionate selection [3]. By separating the parts of selection and crossover and making the model modularized, we then develop the corresponding selection part of pairwise tournament selection. After adding IRO into the model, we generalize tournament selection of our model to s -ary tournament selection and probabilistic tournament selection.

3.1 Separating Selection and Crossover

Start from the model for proportionate selection [16]:

$$\begin{aligned} P_{0,0}^{t+1} &= P_{0,0}^t \frac{f_{0,0}}{\bar{f}} \left[1 - p_c \frac{f_{1,1}}{\bar{f}} P_{1,1}^t \right] + p_c \frac{f_{0,1} f_{1,0}}{\bar{f}^2} P_{0,1}^t P_{1,0}^t; \\ P_{0,1}^{t+1} &= P_{0,1}^t \frac{f_{0,1}}{\bar{f}} \left[1 - p_c \frac{f_{1,0}}{\bar{f}} P_{1,0}^t \right] + p_c \frac{f_{1,1} f_{0,0}}{\bar{f}^2} P_{1,1}^t P_{0,0}^t; \\ P_{1,0}^{t+1} &= P_{1,0}^t \frac{f_{1,0}}{\bar{f}} \left[1 - p_c \frac{f_{0,1}}{\bar{f}} P_{0,1}^t \right] + p_c \frac{f_{0,0} f_{1,1}}{\bar{f}^2} P_{0,0}^t P_{1,1}^t; \\ P_{1,1}^{t+1} &= P_{1,1}^t \frac{f_{1,1}}{\bar{f}} \left[1 - p_c \frac{f_{0,0}}{\bar{f}} P_{0,0}^t \right] + p_c \frac{f_{1,0} f_{0,1}}{\bar{f}^2} P_{1,0}^t P_{0,1}^t, \end{aligned}$$

where $P_{i,j}^t$, $i, j \in \{0, 1\}$ is the portion of population of schema (i, j) at generation t , p_c is the *effective* crossover probability which combines the actual crossover probability with the disrupting probability introduced by the linkage of the schema, and $\bar{f}$ is the average fitness value. We can separate the selection and crossover parts of the model by defining the population portion after proportionate selection as

$$Q_{i,j}^t = \frac{f_{i,j}}{\bar{f}} P_{i,j}^t \quad i, j \in \{0, 1\}.$$

By writing the model, we obtain

$$\begin{aligned} P_{i,j}^{t+1} &= P_{i,j}^t \frac{f_{i,j}}{\bar{f}} \left[1 - p_c \frac{f_{(1-i),(1-j)}}{\bar{f}} P_{(1-i),(1-j)}^t \right] \\ &\quad + p_c \frac{f_{i,(1-j)} f_{(1-i),j}}{\bar{f}^2} P_{i,(1-j)}^t P_{(1-i),j}^t \\ &= P_{i,j}^t \frac{f_{i,j}}{\bar{f}} - p_c \frac{f_{i,j} f_{(1-i),(1-j)}}{\bar{f}^2} P_{i,j}^t P_{(1-i),(1-j)}^t \\ &\quad + p_c \frac{f_{i,(1-j)} f_{(1-i),j}}{\bar{f}^2} P_{i,(1-j)}^t P_{(1-i),j}^t \\ &= Q_{i,j}^t - p_c Q_{i,j}^t Q_{(1-i),(1-j)}^t + p_c Q_{i,(1-j)}^t Q_{(1-i),j}^t \end{aligned}$$

where $i, j \in \{0, 1\}$.

Hence, the model can be described as two separate modules:

1. Proportionate selection:

$$Q_{i,j}^t = \frac{f_{i,j}}{\bar{f}} P_{i,j}^t \quad i, j \in \{0, 1\}. \quad (2)$$

2. One-point crossover:

$$P_{i,j}^{t+1} = Q_{i,j}^t - p_c Q_{i,j}^t Q_{(1-i),(1-j)}^t + p_c Q_{i,(1-j)}^t Q_{(1-i),j}^t \quad i, j \in \{0, 1\}.$$

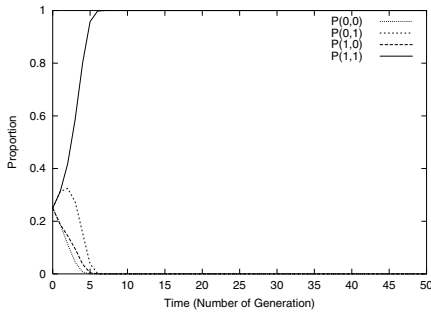


Fig. 2. Numerical solution of the MDP Type II showing convergence to the optimal solution when the initial condition is $P_{i,j}^0 = 0.25$ $i, j \in \{0, 1\}$.

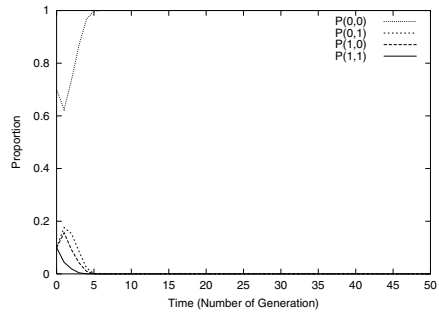


Fig. 3. Numerical solution of the MDP Type II showing divergence away from the optimal solution when the initial condition is $P_{0,0}^0 = 0.7; P_{0,1}^0 = P_{1,0}^0 = P_{1,1}^0 = 0.1$.

3.2 Pairwise Tournament Selection

After getting separate parts of the model, replacing the selection part with pairwise tournament selection is straightforward. Because the fitness values of the test function follow

$$f_{1,1} > f_{0,0} > f_{0,1} > f_{1,0},$$

we can easily write down the equations representing the portion of population after pairwise tournament selection:

$$\begin{aligned} Q_{1,1}^t &= 1 - (1 - P_{1,1}^t)^2; \\ Q_{0,0}^t &= (1 - P_{1,1}^t)^2 - (1 - (P_{1,1}^t + P_{0,0}^t))^2; \\ Q_{0,1}^t &= (1 - (P_{1,1}^t + P_{0,0}^t))^2 - (P_{1,0}^t)^2; \\ Q_{1,0}^t &= (P_{1,0}^t)^2. \end{aligned} \tag{3}$$

Substituting the proportionate selection module with the pairwise tournament selection module, we get the model combining IRO and tournament selection. Figures 2 and 3 show the numerical results of the pairwise tournament selection model for two different initial conditions. In the first initial condition, portions of all schemata are equal, i.e., $P_{i,j}^0 = 0.25$ $i, j \in \{0, 1\}$. In the second initial condition, the initial population is biased toward the sub-optimal solution that $P_{0,0}^0 = 0.7; P_{0,1}^0 = P_{1,0}^0 = P_{1,1}^0 = 0.1$.

The two initial conditions used here are identical to that used elsewhere [3] for comparison purpose. The results show that replacing proportionate selection with pairwise tournament selection alone does not make the GA capable of overcoming the difficulty. It still diverges under the second initial condition. The difference of using tournament selection is that the convergence or divergence comes much faster. Since it is well-known that the takeover time of tournament

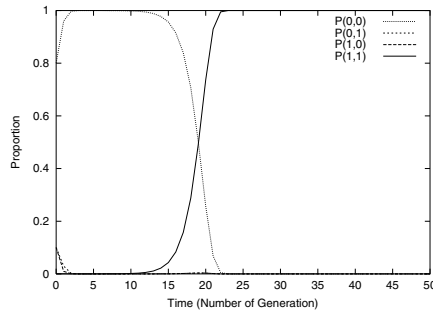


Fig. 4. Numerical solution of the MDP Type II showing convergence to the optimal solution when $p_r = 0.01$. Combined results for both short building blocks and long building blocks.

selection is much shorter than that of proportionate selection [22], the time difference is expected.

3.3 Using IRO

Apparently, replacing proportionate selection does not change the basic behavior of a GA. We now insert the *idealized reordering operator* (IRO) into our model to verify its performance. IRO is assumed to transfer a build block between its *long* version (loose linkage) and *short* version (tight linkage). For simplicity, we add another index k to the model equation terms for distinguishing short ($k = 0$) and long ($k = 1$). The difference of being long or short reflects on the effective crossover probability. If a building block is tightly linked (short), we assume that the effective crossover probability $p_{c,0} = 0$, which means the building block will not be disrupted. Otherwise, we assume $p_{c,1} = 1$, meaning the schema is very likely to be destroyed.

Because crossover events only occur between individuals of the same defining length of building blocks, we can write the crossover parts with the extra index by introducing a new intermediate portion $R_{i,j,k}^t$ as

$$R_{i,j,k}^{t+1} = Q_{i,j,k}^t - p_{c,k} Q_{i,j,k}^t Q_{(1-i),(1-j),k}^t + p_{c,k} Q_{i,(1-j),k}^t Q_{(1-i),j,k}^t \quad i, j, k \in \{0, 1\}, \quad (4)$$

where $R_{i,j,k}^t$ is the population portion of schema (i, j, k) at generation t after crossover.

After crossover, IRO is responsible for transferring a building block between its long and short version with reordering probability p_r as

$$P_{i,j,k}^{t+1} = (1 - p_r) R_{i,j,k}^t + p_r R_{i,j,(1-k)}^t \quad i, j, k \in \{0, 1\}, \quad (5)$$

where on the right hand side, the first term indicates the building blocks remaining to be the same version, and the second term specifies the building blocks transferred from the other version.

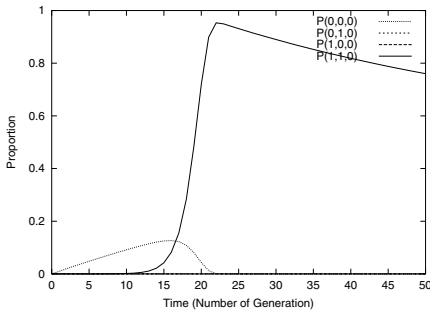


Fig. 5. Numerical solution of the MDP Type II showing convergence to the optimal solution when $p_r = 0.01$. Short building blocks.

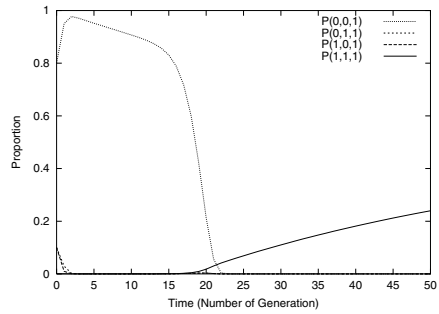


Fig. 6. Numerical solution of the MDP Type II showing convergence to the optimal solution when $p_r = 0.01$. Long building blocks.

Thus, the model with IRO consists of the following three modules:

1. Pairwise tournament selection (Equation (3));
2. One-point crossover (Equation (4));
3. Idealized reordering operator (Equation (5)).

To make the problem harder, we adopt the third initial condition that $P_{0,0}^0 = 0.8$; $P_{0,1}^0 = P_{1,0}^0 = 0.1$; $P_{1,1}^0 = 0$ [3]. This initial condition specifies that the way to have schema (1, 1) is to create it via crossover and make it stay in the population without being disrupted. We first try a low reordering probability $p_r = 0.01$ to see if the reordering operator also helps a GA to converge with tournament selection.

Figures 4, 5, and 6 show the numerical results after inserting IRO into the model. Apparently, IRO works as we expected to help the GA to converge to the optimal solutions. The process can be roughly divided into three stages. First, the short version of (1, 1) is created by the crossover. Only the short version can survive at this stage because it cannot be disrupted even both short and long versions are equally favored by the selection. Then, the optimal schema starts to takeover the population. The period of this stage is determined by the takeover time. After the optimal schema takeover the population, there is no need to maintain linkage. Therefore, the portion of long starts to grow, and the portion of short starts to decrease until reaching the balance.

Until now, there seems no fundamental difference between using proportionate selection and using tournament selection. Except for the time scale, the behavior does not seem to be different. However, if we use a higher reordering probability $p_r = 0.10$. We can get the numerical results in Figure 7.

Unexpectedly, the GA also converged to the optimal solution. Using the same reordering probability, the GA diverges instead of converges. Because the upper bound for the reordering probability was developed based on using proportionate selection, it might be different if tournament selection is used. Therefore, we

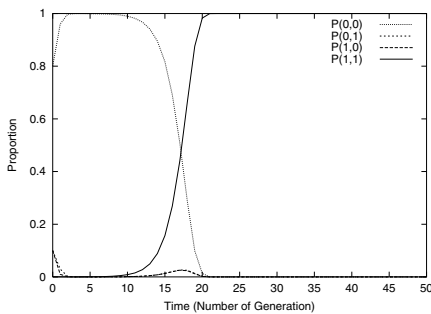


Fig. 7. Numerical solution of the MDP Type II showing convergence to the optimal solution even when $p_r = 0.10$. Combined results for both short building blocks and long building blocks.

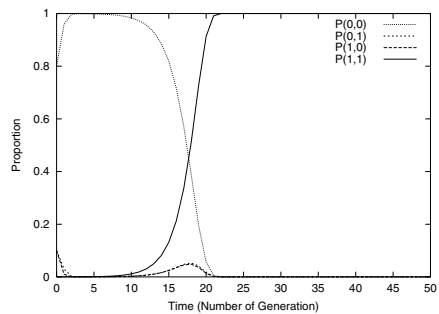


Fig. 8. Numerical solution of the MDP Type II showing convergence to the optimal solution even when $p_r = 0.25$. Combined results for both short building blocks and long building blocks.

conduct simulations with even higher reordering probabilities $p_r = 0.25, 0.75$, and 0.99 . The results are shown in Figures 8, 9, and 10.

Surprisingly, the GA still converged to the optimal solution even with a very high reordering probability. It indicates that there might not be an upper bound for reordering probability except that $0 < p_r < 1$.

3.4 S -ary Tournament Selection

In addition to pairwise tournament selection, we also generalize the model to include the commonly used s -ary tournament selection as follows. First, we define an order function $o(\cdot)$ for each schema based on their fitness values:

$$o(0) = (-1, -1); \quad o(1) = (1, 1); \quad o(2) = (0, 0); \quad o(3) = (0, 1); \quad o(4) = (1, 0),$$

where $(-1, -1)$ is a boundary condition for convenience, and $P_{-1,-1}^t = 0 \forall t \geq 0$. Second, we define the accumulated population portion with the order given by $o(\cdot)$ as

$$A_{o(n)}^t = \sum_{m=0}^n P_{o(m)}^t \quad 0 \leq n \leq 4.$$

With the help of the ordering function and accumulated portion, we can rewrite (3) as follows:

$$Q_{o(n)}^t = \begin{cases} 0 & n = 0 \\ \left(1 - A_{o(n-1)}^t\right)^2 - \left(1 - A_{o(n)}^t\right)^2 & 0 < n \leq 4 \end{cases} \quad (6)$$

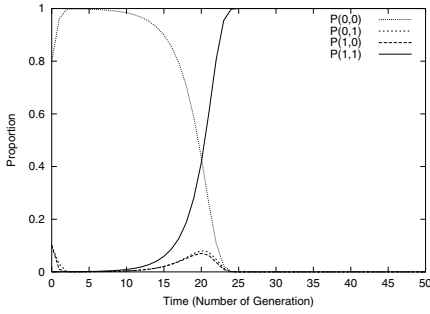


Fig. 9. Numerical solution of the MDP Type II showing convergence to the optimal solution even when $p_r = 0.75$. Combined results for both short building blocks and long building blocks.

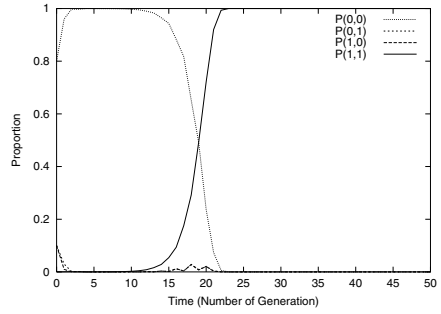


Fig. 10. Numerical solution of the MDP Type II showing convergence to the optimal solution even when $p_r = 0.99$. Combined results for both short building blocks and long building blocks.

Then we can generalize the equation to s -wise tournament selection by replacing the square with the sth power:

$$Q_{o(n)}^t = \begin{cases} 0 & n = 0 \\ \left(1 - A_{o(n-1)}^t\right)^s - \left(1 - A_{o(n)}^t\right)^s & 0 < n \leq 4 \end{cases} \quad (7)$$

3.5 Probabilistic Tournament Selection

Probabilistic tournament selection can also be modeled with our framework. Considering pairwise tournament selection, after tournament, the winner gets into the next generation with a fixed probability p , $0.5 < p \leq 1$. Pairwise tournament selection can be considered as a special case with $p = 1$.

To include probabilistic tournament selection, we start from (6). Since when some schema wins, it actually gets into the next generation with p , and it also gets selected with $1 - p$ when losing a tournament, we can modified (6) to model probabilistic tournament selection as

$$Q_{o(n)}^t = \begin{cases} 0 & n = 0 \\ p \left(\left(1 - A_{o(n-1)}^t\right)^2 - \left(1 - A_{o(n)}^t\right)^2 \right) \\ \quad + (1 - p) \left(\left(A_{o(n)}^t\right)^2 - \left(A_{o(n-1)}^t\right)^2 \right) & 0 < n \leq 4 \end{cases} \quad (8)$$

Numerical results for both s -ary tournament selection and probabilistic tournament selection are omitted because of the length limitation. Those results are also consistent with our previous observation on pairwise tournament selection. The fundamental behavior of the algorithm remains the same when different kinds of tournament selection are employed.

4 Conclusions

It has been demonstrated that an idealized reordering operator can help a genetic algorithm to overcome certain difficulty on the MDP Type II [3]. However, an upper bound of the reordering probability was also derived to explain why the reordering operator did not help the genetic algorithm when the probability was set to be a little bit high. In this paper, we extend the model to include the commonly used tournament selection. The proposed model can describe pairwise tournament selection, s -ary tournament selection, and probabilistic tournament selection. By analyzing the performance of a GA with the proposed model, we can find that there seems no upper bound on reordering probability. The genetic algorithm still converges even if the probability approaches to 1. Therefore, using the reordering operator with tournament selection can give us much better results than with proportionate selection.

Tournament selection has been widely utilized because of its excellence, including independence of fitness scaling, ease for implementation, and so on. However, based on the study, when conducting linkage learning, tournament selection becomes a necessity rather than a choice. If proportionate selection is used, the reordering probability has to be limited to ensure success. But in practice, the reordering probability might not be known or even controllable. Hence, we have to use tournament selection under this condition so that the algorithm is able to achieve its goal.

Future work along this line includes extending the model so that more operators and parameters can be described, using the model to observe and explain the linkage learning process, and deriving characteristic properties, such as tightness time, of linkage learning.

Acknowledgments. The work was sponsored by the Air Force Office of Scientific Research, Air Force Materiel Command, USAF, under grant F49620-00-0163. Research funding for this work was also provided by a grant from the National Science Foundation under grant DMI-9908252. The US Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation thereon.

The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Air Force Office of Scientific Research, the National Science Foundation, or the U.S. Government.

References

1. Goldberg, D.E., Deb, K., Clark, J.H.: Genetic algorithms, noise, and the sizing of populations. *Complex Systems* **6** (1992) 333–362
2. Goldberg, D.E., Deb, K., Thierens, D.: Toward a better understanding of mixing in genetic algorithms. *Journal of the Society of Instrument and Control Engineers* **32** (1993) 10–16

3. Goldberg, D.E., Bridges, C.L.: An analysis of a reordering operator on a GA-hard problem. *Biological Cybernetics* **62** (1990) 397–405
4. Bagley, J.D.: *The Behavior of Adaptive Systems Which Employ Genetic and Correlation Algorithms*. PhD thesis, University of Michigan (1967) (University Microfilms No. 68-7556).
5. Cavicchio, Jr., D.J.: *Adaptive Search Using Simulated Evolution*. Unpublished doctoral dissertation, University of Michigan, Ann Arbor, MI (1970) (University Microfilms No. 25-0199).
6. Frantz, D.R.: *Non-linearities in genetic adaptive search*. PhD thesis, University of Michigan (1972) (University Microfilms No. 73-11,116).
7. Holland, J.H.: *Adaptation in natural and artificial systems*. University of Michigan Press, Ann Arbor, MI (1975)
8. Smith, S.F.: A learning system based on genetic adaptive algorithms. *Dissertation Abstracts International* **41** (1980) 4582B (University Microfilms No. 81-12638).
9. Davis, L., Smith, D.: Adaptive design for layout synthesis. Internal report, Texas Instruments, Dallas (1985)
10. Davis, L.: Applying adaptive algorithms to epistatic domains. *Proceedings of the 9th International Joint Conference on Artificial Intelligence* **1** (1985) 162–164
11. Goldberg, D.E., Lingle, Jr., R.: Alleles, loci, and the traveling salesman problem. *Proceedings of an International Conference on Conference on Genetic Algorithms* (1985) 154–159
12. Oliver, I.M., Smith, D.J., Holland, J.R.C.: A study of permutation crossover operators on the traveling salesman problem. *Proceedings of the Second International Conference on Genetic Algorithms* (1987) 224–230
13. Whitley, D., Starkweather, T., Fuquay, D.: Scheduling problems and traveling salesmen: The genetic edge recombination operator. *Proceedings of the Third International Conference on Genetic Algorithms* (1989) 133–140
14. Banzhaf, W.: The 'molecular' traveling salesman. *Biological Cybernetics* **64** (1990) 7–14
15. Starkweather, T., McDaniel, S., Mathias, K., Whitley, D., Whitley, C.: A comparison of genetic sequencing operators. *Proceedings of the Fourth International Conference on Genetic Algorithms* (1991) 69–76
16. Goldberg, D.E.: Simple genetic algorithms and the minimal, deceptive problem. In Davis, L., ed.: *Genetic algorithms and simulated annealing*. Morgan Kaufmann Publishers, Inc., Los Altos, CA (1987) 74–88
17. Yamamura, M., Satoh, H., Kobayashi, S.: An Markov analysis of generation alternation models on minimal deceptive problems. In Wang, P.P., ed.: *International Conference of Information Sciences: Fuzzy Logic, Intelligent Control & Genetic Algorithm*. Volume 1., Durham, NC, Duke University (1997) 47–50
18. Novkovic, S., Šverko, D.: The minimal deceptive problem revisited: the role of 'genetic waste'. *Computers and Operations Research* **25** (1998) 895–911
19. Grefenstette, J.J.: Deception considered harmful. *Foundations of Genetic Algorithms 2* (1993) 75–91
20. Forrest, S., Mitchell, M.: What makes a problem hard for a genetic algorithm? Some anomalous results and their explanation. *Machine Learning* **13** (1993) 285–319
21. Naudts, B., Verschoren, A.: Epistasis and deceptivity. *Simon Stevin - Bulletin of the Belgian Mathematical Society* **6** (1999) 147–154
22. Goldberg, D.E., Deb, K.: A comparative analysis of selection schemes used in genetic algorithms. *Foundations of Genetic Algorithms 1* (1991) 69–93 (Also TCGA Report 90007).

Tightness Time for the Linkage Learning Genetic Algorithm

Ying-ping Chen¹ and David E. Goldberg²

¹ Department of Computer Science and Department of General Engineering
University of Illinois, Urbana, IL 61801, USA
`ypchen@illigal.ge.uiuc.edu`

² Department of General Engineering
University of Illinois, Urbana, IL 61801, USA
`deg@uiuc.edu`

Abstract. This paper develops a model for *tightness time*, linkage learning time for a single building block, in the linkage learning genetic algorithm (LLGA). First, the existing models for both linkage learning mechanisms, *linkage skew* and *linkage shift*, are extended and investigated. Then, the tightness time model is derived and proposed based on the extended linkage learning mechanism models. Experimental results are also presented in this study to verify the extended models for linkage learning mechanisms and the proposed model for tightness time.

1 Introduction

Linkage learning, one of the fundamental challenges of the research and development of genetic algorithms (GAs), has often been either ignored or overlooked in the field of evolutionary computation. In spite of Holland's call for the evolution of tight linkage in his historical publication of *Adaptation in Natural and Artificial Systems* [1], there have been relatively few efforts made on the subject of linkage learning and evolution until the past decade. Thierens and Goldberg [2,3] showed that simple genetic algorithms fail to solve hard problems without tight linkage and analyzed several possible but unsuccessful techniques to overcome the linkage learning difficulty. Recognizing the importance of linkage evolution in powerful and general evolutionary processes, linkage-related operators and mechanisms were therefore developed.

Among the ways to achieve tight linkage or its equivalent such as perturbation techniques, model builders, and linkage learners is the linkage learning genetic algorithm (LLGA), which uses (*gene number*, *allele*) style coding scheme with non-coding segments [4,5,6] to create an evolvable genotypic structure that makes GAs capable of learning tight linkage of building blocks through the special expression mechanism, *probabilistic expression* (PE) [7,8]. Compared to the perturbation-based algorithms, the LLGA does not employ extra linkage detection procedures to gain knowledge of linkage. Compared to the model builders,

the LLGA operates in a local, distributed manner without building a global model across all individuals. Therefore, the linkage learning process is a very special and essential part of the LLGA, and this paper seeks to better understand the LLGA's linkage learning mechanisms.

In particular, this paper examines the current theoretical analysis of the linkage learning mechanisms [9,7]. Models for both linkage learning mechanisms, *linkage skew* and *linkage shift*, are refined and extended. The theoretical results are confirmed with experiments. Moreover, a model for *tightness time*, linkage learning time for a single building block, is therefore proposed and empirically verified. Artificial evolutionary systems usually create fitness associated with a greedy choice of the best alleles before linkage has evolved. Understanding the race between allelic convergence and linkage convergence is critical to designing LLGAs that work well. In particular, understanding tightness time is critical to getting time scale right, especially in systems like the LLGA.

This paper is organized as follows. The following section gives a short survey of competent GAs and a brief review of the LLGA. Section 3 extends the existing theoretical analysis of both linkage learning mechanisms, and Sect. 4 verifies the models with experimental results. Section 5 proposes the tightness time model based on the extended mechanism models. Finally, the conclusions of this paper are drawn in Sect. 6.

2 Brief Review of Competent GAs and the LLGA

Current linkage detection, adaptation, or learning techniques can be roughly classified into three categories. The first category is based on perturbation techniques. Algorithms in this category, such as messy GA (mGA) [10], fast messy GA (fmGA) [11], gene expression messy GA (GEMGA) [12], and linkage identification by nonlinearity check/non-monotonicity detection (LINC/LIMD) [13, 14], perturb chromosomes in some particular way and detect linkage among genes by observing the difference of fitness. After determining the relationship among genes, building-block (BB) preserving recombination operators are then employed to create promising solutions.

The second category uses model building techniques. Population-based incremental learning (PBIL) [15], univariate marginal distribution algorithm (UMDA) [16], compact GA (cGA) [17], extended compact GA (ECGA) [18], iterated distribution estimation algorithm (IDEA) [19], probabilistic incremental program evolution (PIPE) [20], and BOA [21] belong to this category. Model builders usually grasp the linkage or relationship among genes by building a global probabilistic model with the current population. New individuals are then derived according to the model, and the model building-utilizing process is repeated until the termination criterion is satisfied.

The final category adapts linkage through the chromosome representation and genetic operators. The linkage learning genetic algorithm (LLGA) [9,7,8] falls into this category. By making the chromosome itself capable of representing

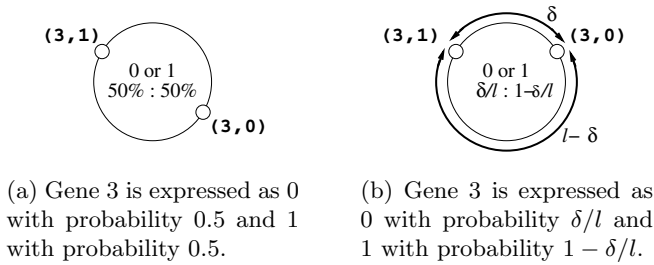


Fig. 1. Probability distributions of gene 3's alleles represented by PE chromosomes.

linkage, the LLGA integrates the linkage learning and utilizing into a unified operation and creates offspring with conceptually well-known genetic operators.

The LLGA is briefly reviewed in the remainder of this section. Readers who are interested in more detail should refer to other materials [9,7,8].

2.1 Chromosome Representation

The LLGA's chromosome representation is mainly composed of moveable genes, non-coding segments, and probabilistic expression. Moveable genes are encoded as (*gene number*, *allele*) pairs in the LLGA chromosome, and a LLGA chromosome is considered as a circle. These genes are allowed to move around and reside anywhere in any order in the chromosome. Non-coding segments are inserted into the chromosome to create an evolvable genotype capable of expressing linkage. Non-coding segments act as non-functional genes, which have no effect on fitness, residing between adjacent genes to generate gaps for expressing linkage precisely.

Probability expression (PE) was proposed to preserve building-block level diversity. For each gene, all possible alleles coexist in a PE chromosome at the same time. For the purpose of evaluation, a chromosome is *interpreted* with a *point of interpretation* (POI). The allele for each gene is determined by the order in which the chromosome is traversed clock-wisely from the point of interpretation. A complete string is then expressed and evaluated.

Consequently, each PE chromosome represents not just a single solution but a probability distribution over the range of possible solutions. Figure 1 shows the probability distribution over alleles of gene 3 of the chromosome. Therefore, if different points of interpretation are selected, a PE chromosome might be interpreted as different solutions. Furthermore, the probability of a PE chromosome to be expressed as a particular solution depends on the length of the non-coding segment between genes critical to that solution. It is the essential technique of the LLGA to capture the knowledge and to prompt the evolution of linkage.

2.2 Exchange Crossover

The exchange crossover operator is another key to make the LLGA able to learn linkage. It is defined on a pair of chromosomes. One of the two chromosomes is

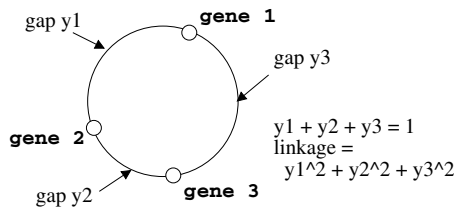


Fig. 2. Calculation for the linkage of a three-gene building block.

the *donor*, and the other is the *recipient*. The exchange crossover cuts a *random* segment of the donor, selects a grafting point on the recipient, and grafts the segment onto the recipient. The grafting point is the point of interpretation of the offspring. Starting from the point of interpretation, redundant genetic materials caused by injection are removed right after crossover to ensure the validity.

2.3 Mechanisms Making the LLGA Work

With the integration of PE and the exchange crossover operator, the LLGA is capable of solving difficult problems without prior knowledge of good linkage. Traditional GAs have been shown to perform poorly on difficult problems [2] without such knowledge. To understand the working of the LLGA, two mechanisms of linkage learning: *linkage skew* and *linkage shift* has been identified and analyzed [9]. Linkage skew occurs when an optimal building block is transferred from the donor to the recipient. Linkage shift occurs when an optimal building block resides in the recipient and survives an injection. Both linkage skew and linkage shift make the building block’s linkage tighter. Thus, the linkage of building blocks can evolve during the linkage learning process, and tightly linked building blocks are formed.

3 Linkage Learning Mechanisms

In this section, we extend the existing theoretical models for both linkage learning mechanisms. First, the definition of linkage of a building block is introduced. Then, the existing models for linkage skew and linkage shift [9] are extended.

3.1 Quantifying Linkage

In order to show the linkage learning and evolutionary process of the LLGA, linkage of building blocks has to be quantified. In this paper, we employ a proposed definition [9], which is the sum of the square of its inter-gene distances, considering the chromosome to be a circle of circumference 1. Figure 2 shows an example for calculating the linkage of a three-gene building block. The definition is appropriate in that linkage in such definition specifies a measure directly

proportional to the probability for a building block to be preserved under the exchange crossover operator. Additionally, it was also theoretically justified that for any linkage learning operator working on the same form of representation, the expected linkage of a randomly spaced order- k building block is $\frac{2}{k+1}$ [9].

3.2 Linkage Skew

Linkage skew, the first linkage learning mechanism, occurs when an optimal building block is successfully transferred from the donor onto the recipient. The conditions for an optimal building block to be transferred are (1) the optimal building block resides in the cut segment, and (2) the optimal building block gets expressed before the deceptive one does. The effect of linkage skew was found to make linkage distributions move toward higher linkages by eliminating less fit individuals. Linkage skew does not make the linkage of a building block of any particular individual tighter. Instead, it drives the whole linkage distribution to a higher state.

Let $A_t(\lambda)$ be the probability density function of the random variable λ representing the linkage of the optimal building block at generation t . The following model to describe the evolution of linkage under linkage skew only has been proposed [9]:

$$A_{t+1}(\lambda) = \frac{\lambda A_t(\lambda)}{\overline{A_t}}. \quad (1)$$

Based on (1), the linkage average at generation $t + 1$ was calculated as [7]:

$$\overline{A_{t+1}} = \int_0^1 \lambda A_{t+1}(\lambda) d\lambda = \frac{\overline{A_t^2}}{\overline{A_t}}, \quad (2)$$

and thus

$$\overline{A_{t+1}} = \overline{A_t} + \frac{\sigma^2(A_t)}{\overline{A_t}}, \quad (3)$$

where $\sigma^2(A_t)$ is the variance in the linkage distribution at generation t .

In addition to the average (i.e. the first moment) of $A_{t+1}(\lambda)$, we can actually calculate all other moments of $A_{t+1}(\lambda)$ in the same way:

$$\overline{A_{t+1}^n} = \int_0^1 \lambda^n A_{t+1}(\lambda) d\lambda = \frac{\overline{A_t^{n+1}}}{\overline{A_t}}. \quad (4)$$

According to the property of a probability distribution, the moments about the origin completely characterize a probability distribution [22]. From (4), we can construct all the moments about the origin of $A_{t+1}(\lambda)$ as follows:

$$\overline{A_{t+1}^n} = \frac{\overline{A_t^{n+1}}}{\overline{A_t}} \quad n = 1, 2, 3, \dots \quad (5)$$

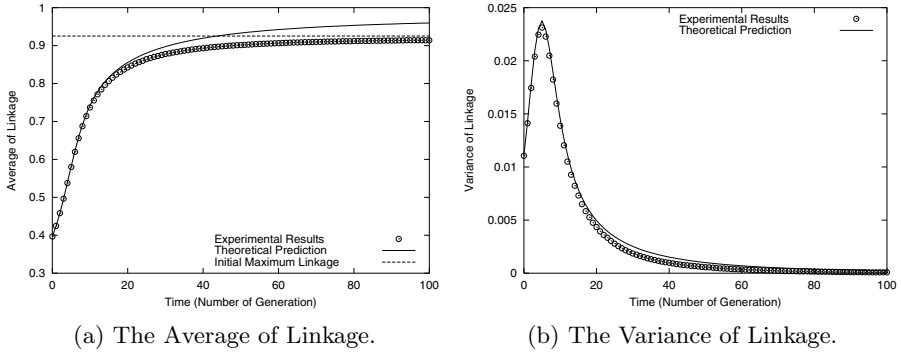


Fig. 3. Linkage Skew on an Order-4 Trap Building Block.

Hence, the relation between $\Lambda_t(\lambda)$ and $\Lambda_{t+1}(\lambda)$ is established.

After knowing the relation between $\Lambda_t(\lambda)$ and $\Lambda_{t+1}(\lambda)$, the moment generation function (mgf) defined as follows can help us a lot:

$$m_{\Lambda_t}(s) = E[e^{s\Lambda_t}] = \int_{-\infty}^{\infty} e^{s\lambda} \Lambda_t(\lambda) d\lambda. \quad (6)$$

Assume that the mgf of $\Lambda_t(\lambda)$ exists, it can be written as

$$\begin{aligned} m_{\Lambda_t}(s) &= E[e^{s\Lambda_t}] \\ &= E\left[1 + \Lambda_t s + \frac{(\Lambda_t s)^2}{2!} + \frac{(\Lambda_t s)^3}{3!} + \dots\right] \\ &= 1 + \overline{\Lambda_t} s + \overline{\Lambda_t^2} \frac{s^2}{2!} + \overline{\Lambda_t^3} \frac{s^3}{3!} + \dots \end{aligned}$$

The r^{th} moment of $\Lambda_t(\lambda)$ can be obtained with

$$\overline{\Lambda_t^r} = m_{\Lambda_t}^{(r)}(0) = \left. \frac{d^r m_{\Lambda_t}(s)}{ds^r} \right|_{s=0}.$$

Given the relation between $\Lambda_t(\lambda)$ and $\Lambda_{t+1}(\lambda)$ and the property of the moment generating function, we can now get the mgf of $\Lambda_{t+1}(\lambda)$:

$$m_{\Lambda_{t+1}}(s) = m'_{\Lambda_t}\left(\frac{s}{m'_{\Lambda_t}(0)}\right). \quad (7)$$

Therefore, we have a model to calculate $\Lambda_{t+1}(\lambda)$ from $\Lambda_t(\lambda)$ when the mgf is available.

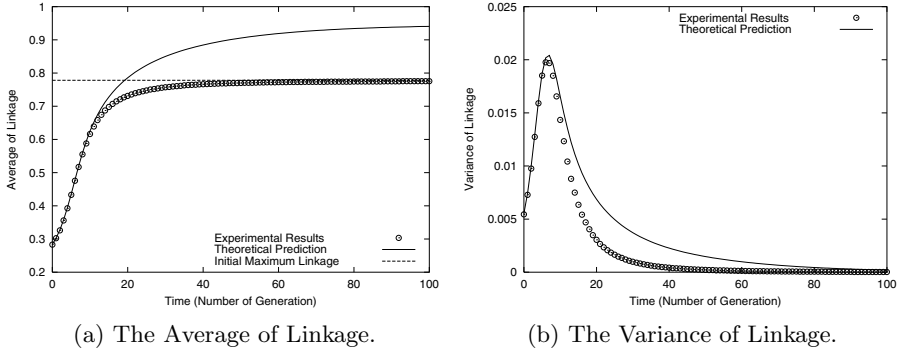


Fig. 4. Linkage Skew on an Order-6 Trap Building Block.

Moreover, also based on (5), we can obtain the following result:

$$\begin{aligned}
 \overline{\Lambda_t^n} &= \frac{\overline{\Lambda_{t-1}^{n+1}}}{\overline{\Lambda_{t-1}}} = \frac{\overline{\Lambda_{t-2}^{n+2}} / \overline{\Lambda_{t-2}}}{\overline{\Lambda_{t-2}^2} / \overline{\Lambda_{t-2}}} \\
 &= \frac{\overline{\Lambda_{t-2}^{n+2}}}{\overline{\Lambda_{t-2}}} = \dots \\
 &= \frac{\overline{\Lambda_0^{t+n}}}{\overline{\Lambda_0}},
 \end{aligned} \tag{8}$$

which clearly indicates that under linkage skew, any moment of the linkage distribution at any given generation can be predicted with the information of the initial linkage distribution. The linkage learning process is solely determined by the initial distribution if there is only linkage skew working.

Finally, based on its property, linkage skew does not really tighten building blocks in any individual. It drives the linkage distribution to a higher place by propagating tight building blocks among individuals. Apparently, the linkage cannot exceed the maximum linkage in the initial population. The evolution of linkage described by (7) is therefore bounded by the initial maximum linkage.

3.3 Linkage Shift

Linkage shift is the second linkage learning mechanism [9]. It occurs when an optimal building block resides in the recipient and survives a crossover event. For the optimal building block to survive, there cannot be any gene contributing to a deceptive building block transferred. Linkage shift gets the linkage of a building block in an individual higher with deletion of duplicate genetic material caused by injection of the exchange crossover. Compared to linkage skew, linkage shift gets linkage of building blocks in each individual higher.

For linkage shift, the following recurrence equation was used [9] to depict the effect of the second mechanism on building blocks that survive crossover:

$$\overline{\lambda_0(t+1)} = \lambda_0(t) + (1 - \lambda_0(t)) \frac{2}{(k+2)(k+3)}, \quad (9)$$

for an order- k building block. Tracking only the average of linkage, we can approximately rewrite (9) as

$$\overline{\lambda_{t+1}} = \overline{\lambda_t} + (1 - \overline{\lambda_t}) \frac{2}{(k+2)(k+3)}. \quad (10)$$

Given a fixed k , let $c = \frac{2}{(k+2)(k+3)}$, we can get the following recurrence relation:

$$\begin{aligned} \overline{\lambda_{t+1}} &= \overline{\lambda_t} + c(1 - \overline{\lambda_t}) \\ &= \overline{\lambda_t} (1 - c) + c. \end{aligned} \quad (11)$$

By solving the recurrence relation, the new linkage shift model is obtained as

$$\overline{\lambda_t} = 1 - (1 - \overline{\lambda_0})(1 - c)^t. \quad (12)$$

Therefore, the rate of linkage learning is mainly determined by the linkage average of the initial linkage distribution. Besides, the higher order the building block is, the longer it takes to evolve to some specific level of linkage.

4 Experimental Results

The experimental results to verify the extended linkage learning mechanism models are presented in this section. First, the parameter settings of the experiments are described. Then, experimental results for both models are shown.

4.1 Parameter Settings

In this paper, we use trap functions [23] to verify the theoretical model. We basically use similar experiments presented elsewhere [9], but the experiments in this study were done for both order-4 and order-6 traps. An order- k trap function used in this study can be described by

$$\text{trap}_k(u) = \begin{cases} u & u = k \\ k - 1 - u & \text{otherwise} \end{cases},$$

where u is the number of ones in the bitstring. In order to simulate the infinite-length chromosome, we let the functional genes occupy only one percent of the chromosome. That is, for the order-4 building block, the 4 genes are embedded in a 400-gene chromosome with 396 nonfunctional genes; for the order-6 building block, the 6 genes are embedded in a 600-gene chromosome with 594 nonfunctional genes. The population size are 5000 in both cases. All results in this section are averaged over 50 independent runs.

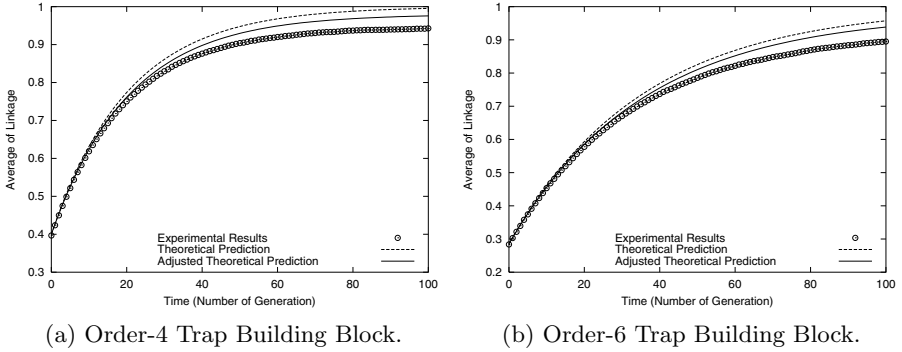


Fig. 5. The Average of Linkage under Linkage Shift.

4.2 Linkage Skew

The extended linkage skew model can predict all moments at any given generation. For illustration purpose, we show only the prediction of the average (the first moment) and the variance (the second moment minus the square of the average) in figures.

Figures 3 and 4 show the experimental results compared to the theoretical prediction. The theoretical prediction was made based on (8). As shown in the figures, the experimental results agree with the prediction, and the extended linkage skew model is confirmed experimentally.

4.3 Linkage Shift

For linkage shift, we predict the average of linkage on both order-4 and order-6 traps. Figure 5 shows the experimental results. The theoretical prediction was made according to (12). We also employ the adjustment scheme [9] to reflect the difference between the infinite-length chromosome model and the real chromosomes in experiments. In this study, the maximum possible linkage in both cases is $(0.99)^2 = 0.9801$, and (12) is adjusted accordingly. As indicated in Fig. 5, the experimental results agree with the adjusted theoretical prediction, and we are now given good reason to believe that the extended model is accurate.

5 Tightness Time

5.1 The Model

Equipped with the extended models for linkage learning, we are now ready to develop the tightness time model. Based on the observation and intuition, the working relationship between linkage skew and linkage shift is as follows. Linkage shift is responsible for making the linkage of a building block in each individual

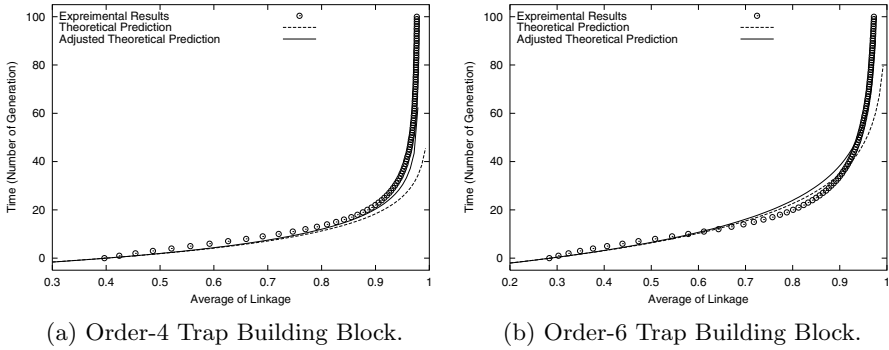


Fig. 6. Tightness Time for a Single Building Block.

tighter; linkage skew is responsible for driving the whole distribution toward a higher place. Considering linkage skew as a refiner, linkage skew as a propagator, and that the effect of linkage skew comes pretty fast based on the experimental result, the linkage learning bottleneck is in fact linkage shift. Hence, we now start to develop the model for tightness time based on the most critical component in the framework first.

Start from (12), we can obtain

$$t = \frac{\log(1 - \overline{\lambda}_t) - \log(1 - \overline{\lambda}_0)}{\log(1 - c)}.$$

Then, it can be rewritten as a function of linkage λ :

$$t(\lambda) = \frac{\log(1 - \lambda) - \log(1 - \overline{\lambda}_0)}{\log(1 - c)}.$$

By taking the propagation effect of linkage skew into account, a constant c_s standing for the linkage learning speed-up caused by linkage skew is added into the model. Thus, we obtain the tightness time model as follows:

$$t_\ell(\lambda) = \frac{\log(1 - \lambda) - \log(1 - \overline{\lambda}_0)}{c_s \log(1 - c)}, \quad (13)$$

where $t_\ell(\lambda)$ is the tightness time for a given linkage λ , and $c_s \approx 2$ is determined empirically.

Furthermore, given the initial linkage distribution, $\overline{\lambda}_0$ remains constant during the whole process. For simplicity, we can define

$$\begin{aligned} \epsilon &= 1 - \lambda, \\ \overline{\epsilon}_0 &= 1 - \overline{\lambda}_0. \end{aligned}$$

Also, $c = \frac{2}{(k+2)(k+3)} \approx \frac{2}{k^2}$ when $k \rightarrow \infty$. Therefore, (13) can be rewritten as a function of ϵ as

$$t'_\ell(\epsilon) = \frac{k^2}{2c_s} \log \frac{\epsilon}{\epsilon_0}. \quad (14)$$

Equation (14) shows that tightness time is proportional to the square of the order of building blocks. The longer the building block, the much longer the tightness time. Besides, tightness time is proportional to the logarithm of the desired linkage.

5.2 Verification

Experiments were also performed to verify the model for tightness time. Using the same parameter settings described in section 4.1, both linkage learning mechanisms work together in the experiments.

The experimental results are shown in Figure 6. The theoretical prediction made based on (13) is also adjusted with the maximum possible linkage 0.9801. The obtained numerical data agree with our tightness time model quite well. Our hypothesis and model are therefore experimentally verified.

6 Conclusions

One of the most important issues of the design of genetic algorithms is linkage learning. Harik took Holland's call for the evolution of tight linkage seriously and developed the linkage learning genetic algorithm, which learns linkage among genes with specially designed chromosome representation and conceptually well-known genetic operators. While the LLGA performs remarkably well on badly scaled building blocks, it does not do so on uniformly scaled building blocks. This paper seeks to gain better understanding of linkage learning mechanisms of the LLGA in order to improve the capability of the LLGA.

In this paper, the current theoretical analysis of the linkage learning mechanisms are extended in several aspects and verified with experiments. Based on the extended models, a model for tightness time is proposed. Under the two linkage learning mechanisms, the evolution of linkage is basically determined by the initial linkage distribution. If the linkage learning process needs to be modified or improved, we have to seek help from other mechanisms outside the framework. Additionally, the cooperation and interaction between linkage skew and linkage shift is also theorized. It helps us to better understand the overall effect of the LLGA's linkage learning mechanisms. Finally, some important insights into the whole linkage learning process are obtained from the tightness time model.

Among existing techniques, perturbation-based algorithms and model builders get linkage information and evolve the population in two separate steps, while the LLGA gets linkage tight and alleles right at the same time. Hence, tightness time for the LLGA provides an important key to correctly handling the competition of linkage and allele on time scale. Understanding tightness time

should enable us to improve the LLGA and design better genetic algorithms as well.

More work along this line still needs to be done to understand the capabilities and limits of linkage adaptation techniques. The results shown in this paper give us more theoretical insight of the LLGA's linkage learning mechanisms and take us one step further toward scalable linkage learning.

Acknowledgments. The author would like to thank Kumara Sastry for many useful discussions and valuable comments.

The work was sponsored by the Air Force Office of Scientific Research, Air Force Materiel Command, USAF, under grant F49620-00-0163. Research funding for this work was also provided by a grant from the National Science Foundation under grant DMI-9908252. The US Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation thereon.

The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Air Force Office of Scientific Research, the National Science Foundation, or the U.S. Government.

References

1. Holland, J.H.: *Adaptation in natural and artificial systems*. University of Michigan Press, Ann Arbor, MI (1975)
2. Thierens, D., Goldberg, D.E.: Mixing in genetic algorithms. *Proceedings of the Fifth International Conference on Genetic Algorithms* (1993) 38–45
3. Thierens, D.: *Analysis and Design of Genetic Algorithms*. doctoral dissertation, Katholieke Universiteit Leuven, Leuven, Belgium (1995)
4. Lindsay, R.K., Wu, A.S.: Testing the robustness of the genetic algorithm on the floating building block representation. *Proceedings of the twelfth National Conference on Artificial Intelligence/eighth Innovative Applications of Artificial Intelligence Conference* (1996) 793–798
5. Wineberg, M., Oppacher, F.: The benefits of computing with introns. *Proceedings of the First Annual Conference on Genetic Programming* (1996) 410–415
6. Levenick, J.R.: Swappers: Introns promote flexibility, diversity and invention. *Proceedings of the Genetic and Evolutionary Computation Conference 1999: Volume 1* (1999) 361–368
7. Harik, G.R.: Learning gene linkage to efficiently solve problems of bounded difficulty using genetic algorithms. Unpublished doctoral dissertation, University of Michigan, Ann Arbor (1997) (Also IlliGAL Report No. 97005).
8. Harik, G.R., Goldberg, D.E.: Learning linkage through probabilistic expression. *Computer Methods in Applied Mechanics and Engineering* **186** (2000) 295–310
9. Harik, G.R., Goldberg, D.E.: Learning linkage. *Foundations of Genetic Algorithms 4* (1996) 247–262
10. Goldberg, D.E., Korb, B., Deb, K.: Messy genetic algorithms: Motivation, analysis, and first results. *Complex Systems* **3** (1989) 493–530

11. Goldberg, D.E., Deb, K., Kargupta, H., Harik, G.: Rapid, accurate optimization of difficult problems using fast messy genetic algorithms. *Proceedings of the Fifth International Conference on Genetic Algorithms* (1993) 56–64
12. Kargupta, H.: The gene expression messy genetic algorithm. *Proceedings of 1996 IEEE International Conference on Evolutionary Computation* (1996) 814–819
13. Munetomo, M., Goldberg, D.E.: Identifying linkage groups by nonlinearity/non-monotonicity detection. *Proceedings of the Genetic and Evolutionary Computation Conference 1999: Volume 1* (1999) 433–440
14. Munetomo, M., Goldberg, D.E.: Linkage identification by non-monotonicity detection for overlapping functions. *Evolutionary Computation* **7** (1999) 377–398
15. Baluja, S.: Population-based incremental learning: A method for integrating genetic search based function optimization and competitive learning. Tech. Rep. No. CMU-CS-94-163, Carnegie Mellon University, Pittsburgh, PA (1994)
16. Mühlenbein, H.: The equation for response to selection and its use for prediction. *Evolutionary Computation* **5** (1997) 303–346
17. Harik, G.R., Lobo, F.G., Goldberg, D.E.: The compact genetic algorithm. *Proceedings of 1998 IEEE International Conference on Evolutionary Computation* (1998) 523–528
18. Harik, G.: Linkage learning via probabilistic modeling in the ECGA. IlliGAL Report No. 99010, University of Illinois at Urbana-Champaign, Illinois Genetic Algorithms Laboratory, Urbana, IL (1999)
19. Bosman, P.A.N., Thierens, D.: Linkage information processing in distribution estimation algorithms. *Proceedings of the Genetic and Evolutionary Computation Conference 1999: Volume 1* (1999) 60–67
20. Salustowicz, R.P., Schmidhuber, J.: Probabilistic incremental program evolution. *Evolutionary Computation* **5** (1997) 123–141
21. Pelikan, M., Goldberg, D.E., Cantú-Paz, E.: BOA: The bayesian optimization algorithm. *Proceedings of the Genetic and Evolutionary Computation Conference 1999: Volume 1* (1999) 525–532
22. Zwillinger, D., Kokoska, S.: *CRC standard probability and statistics tables and formulae*. CRC Press, Boca Raton, Florida (2000)
23. Deb, K., Goldberg, D.E.: Analyzing deception in trap functions. *Foundations of Genetic Algorithms 2* (1993) 93–108

A Hybrid Genetic Algorithm for the Hexagonal Tortoise Problem

Heemahn Choe, Sung-Soon Choi, and Byung-Ro Moon

School of Computer Science and Engineering,
Seoul National University,
Seoul, 141-742 Korea
{hchoe,irranum,moon}@soar.snu.ac.kr

Abstract. We propose a hybrid genetic algorithm for the hexagonal tortoise problem. We combined the genetic algorithm with an efficient local heuristic and aging mechanism. Another search heuristic which focuses on the space around existing solutions is also incorporated into the genetic algorithm. With the proposed algorithm, we could find the optimal solutions of up to a fairly large problem.

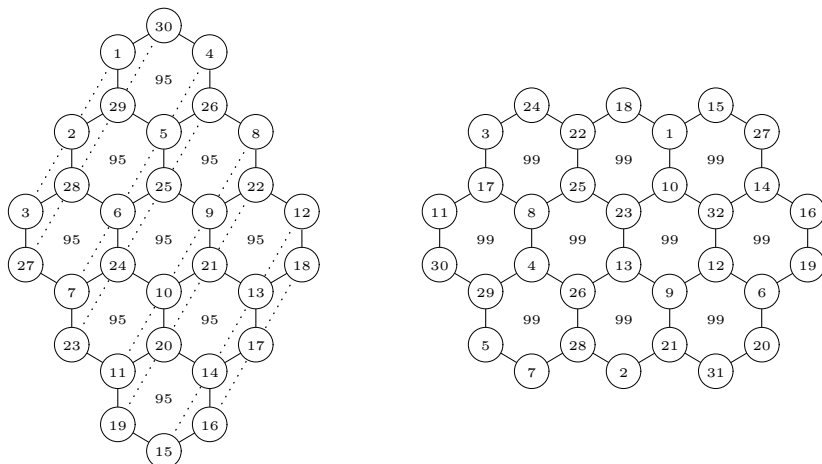
1 Introduction

Hexagonal tortoise problem (HTP), also known as Jisuguimundo, is a numerical puzzle which was invented by medieval Korean scholar and minister Suk-Jung Choi (1646-1715) [1]. The problem is to assign the consecutive numbers 1 through n to the vertices in a graph, which is composed of hexagons like beehives, to make the sum of the numbers of each hexagon the same. Figure 1 shows two example hexagonal tortoises. Choi showed a 30-node hexagonal tortoise in his book along with other various kinds of magic squares.

The HTP is somewhat similar to the magic squares. As for the magic squares, there is no known method that generates solutions for an arbitrary HTP, either. A special kind of HTPs, diamond HTPs, have a particular solution using their symmetry [2]. Figure 1(a) shows the particular solution of a 30-node HTP.

We consider a generalized problem to minimize the variance of hexagonal sums. In this paper, we propose an efficient local optimization heuristic for the HTP using problem-specific knowledge. We also combined a hybrid genetic algorithm with aging mechanism and another search heuristic which focuses on the space around existing solutions.

The rest of this paper is organized as follows. In Section 2, we describe some properties of the hexagonal tortoise problem. In Section 3, we describe our local optimization heuristic. In Section 4, we describe the proposed hybrid genetic algorithm in detail. In Section 5, we show the experimental results. Finally, the conclusion is given in Section 6.



(a) A 30-node 3×3 diamond hexagonal tortoise using the known particular solution. Dotted lines show numbering sequence.

(b) A 32-node hexagonal tortoise

Fig. 1. Example hexagonal tortoises

2 Hexagonal Tortoise Problem

Hexagonal tortoise problem is to assign a sequence of numbers to vertices in a graph with a fixed topology. The graph is composed of a number of overlapping hexagons. The numbers are typically consecutive integers from 1 to n , where n is the number of vertices in the graph. The term “tortoise” comes from the fact that the overall shape of the graph resembles the theca (or shell) of a turtle. Each number has to be assigned to a vertex exactly once. A hexagonal sum is defined to be the sum of the six numbers of a hexagon. The objective is to find an assignment of the numbers to the vertices so that the standard deviation (or variation) of the hexagonal sums is zero. In this paper, we consider a special type of hexagonal tortoise, a $k \times k$ diamond hexagonal tortoise problem, which arrangement of hexagons is like diamond shape [3]. Figure 1(a) shows a solution of 3×3 diamond hexagonal tortoise.

A typical HTP has many solutions with different hexagonal sums, which means that the fitness landscape of the problem is multimodal. For example, in the 16-node HTP, we found 140,915 different optimum solutions among 2^{32} random permutations; from this, we presume there are more than 6×10^8 optimum solutions among total $16!$ possible assignments for the 16-node HTP. The difficulty of multimodal space search with respect to genetic algorithms was pointed out in the literature [4] [5] [6].

To get more insight into the fitness landscape of the problem, we examined the fitness-distance correlation (FDC) [7], which is a popular measure for predicting the difficulty of a problem. We compared it with that of the traveling

Table 1. FDC values of the HTP and the TSP instances

(a) HTP instances		(b) TSP instances	
Instance	FDC value	Instance	FDC value
HTP30	0.1197	lin318	0.6853
HTP48	0.1748	pcb442	0.4999
HTP70	0.0877	att532	0.5617
HTP96	0.0816	rat783	0.6692
HTP126	0.0777	dsj1000	0.2187

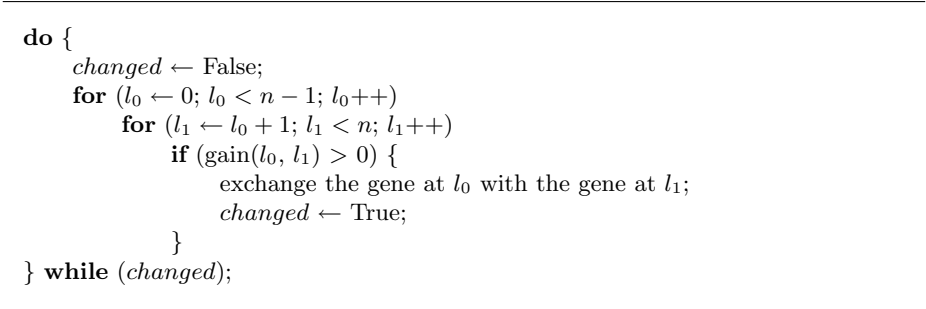


Fig. 2. A simple 2-Opt exchange

salesman problem (TSP). Table 1 compares the FDC values of the HTP and TSP for a number of instances. In the table, “HTP n ” represents the n -node HTP. The table shows that the HTPs with just dozens of nodes have far more rugged landscapes than the TSPs with hundreds of nodes. For example, the HTP with 30 nodes shows a lower FDC value than that of the TSP with 1,000 nodes. Although FDC is not a definite measure for the problem difficulty, this observation says at least the extreme ruggedness of HTP.

3 Local Optimization Heuristic

3.1 Consecutive Exchange

A simple 2-Opt heuristic can be used as a local optimization method for most combinatorial optimization problems [8]. Figure 2 shows the outline of the 2-Opt heuristic for the HTP. It examines all possible $\binom{n}{2}$ pairs of genes. When it finds a pair with positive *gain*, it exchanges them greedily. The gain is defined as $\text{gain}(i, j) = (\text{fitness after exchanging the genes at } i \text{ and } j) - (\text{fitness before the exchange})$.

We used a variant of 2-Opt in the following reasons. When we investigated the gene pairs whose exchanges led to fitness improvements, we found that the gene pairs with small gene-value differences were more probable. Figure 3 shows the frequencies of the gene-value differences for the pairs that led to fitness

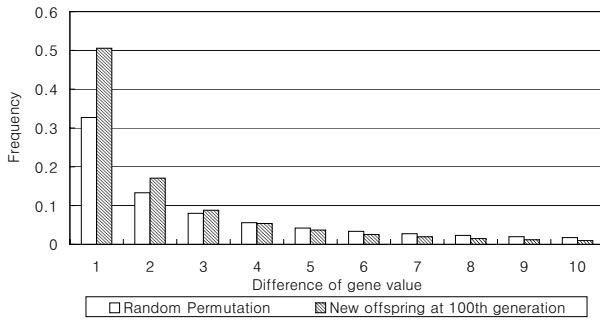


Fig. 3. Frequencies of the gene value of the pairs differences that led to fitness improvements (HTP160)

```

do {
    changed ← False;
    for (v ← 1; v < n; v++) {
        l0 ← locationOf(v);
        l1 ← locationOf(v + 1);
        if (gain(l0, l1) > 0) {
            exchange the gene at l0 with the gene at l1;
            changed ← True;
        }
    }
} while (changed);

```

Fig. 4. The outline of consecutive exchange

improvement in a typical run of our genetic algorithm for the 160-node HTP. For initial chromosomes, the number of the pairs whose difference in value is only one was more than 30% of the total number of pairs. And the number grew further to more than 50% at the 100th generation. This means that the 2-Opt heuristic would waste most time in vainly examining changes of big differences. This is more serious when high-quality solutions are locally optimized.

We designed a local optimization heuristic which examines only the gene pairs whose values are differed by one. We call it *consecutive exchange*. Figure 4 shows the outline of the consecutive exchange. For the problem of size n , while the time complexity of the inner loop of the original 2-Opt exchange in Figure 2 is $\Theta(n^2)$, that of the consecutive exchange is $\Theta(n)$. Comparisons of performances of the 2-Opt and consecutive exchanges are presented in Table 2 which are described in the next section.

```

 $T = \{ \text{original solution} \};$ 
do {
     $changed \leftarrow \text{False};$ 
    for each pair  $l_0, l_1$  by the exchanging policy
        if ( $\text{gain}(l_0, l_1) > 0$ ) {
            exchange the gene at  $l_0$  with the gene at  $l_1$ ;
             $T \leftarrow \{ \text{current solution} \};$ 
             $changed \leftarrow \text{True};$ 
        } else if ( $\text{gain}(l_0, l_1) = 0 \wedge \text{current solution} \notin T$ ) {
            exchange the gene at  $l_0$  with the gene at  $l_1$ ;
             $T \leftarrow T \cup \{ \text{current solution} \};$ 
             $changed \leftarrow \text{True};$ 
        }
    } while ( $changed$ );

```

Fig. 5. Local optimization with tabu search

3.2 Further Optimization Using Tabu Search

In the process of local optimization, there are gene pairs whose exchanges do not affect fitness. Exchanging them may help improvement in other gene positions since it changes the distribution of the hexagonal sums. Usually, there are a great number of equi-fitness solutions near a solution. We adopted tabu search to search these solutions efficiently; we kept a list of the visited equi-fitness solutions and prevented revisiting them.

Tabu search is a meta-heuristic for optimization problems invented by Glover [9]. The tabu search keeps one or more lists of recently visited solutions to prevent the search from revisiting the visited solutions before.

Figure 5 shows the outline of our local optimization incorporated with tabu search. The performances of the 2-Opt heuristic and the consecutive exchange with and without tabu search are summarized at Table 2. We tested them with a traditional genetic algorithm of population size 1,000. We used the number of fitness calculation as the index of processor time. Table 2 shows the numbers of fitness calculations of the heuristics for the initial chromosomes and the chromosomes in 100th generation, respectively. In both cases, the 2-Opt heuristic had slightly lower standard deviation of hexagonal sums (higher fitness) than that of the consecutive exchange. But, the 2-Opt was much slower than the consecutive exchange, especially with tabu search. Table 2 indicates that the consecutive exchanges is more effective in later generations.

4 The Hybrid Genetic Algorithm for the Hexagonal Tortoise Problem

A notable feature of our hybrid genetic algorithm is that it is combined with another search heuristic to search the space around the solutions in the current

Table 2. Performances of the local heuristics. Problem size is 160. SD means standard deviation

(a) The initial chromosomes in a typical GA run

Version	Avg. SD of hex. sums	Avg. fitness gain	Avg. # of fitness calculation	Avg. gain per average calculation
2-Opt without tabu search	1.1648	12364.90	143127.4	0.086391
2-Opt with tabu search	0.7343	12365.75	831483.0	0.014872
Consecutive without tabu search	1.1748	12364.87	8018.8	1.541989
Consecutive with tabu search	0.7516	12365.72	8939.1	1.383337

(b) The chromosomes in the 100th generation in a typical GA run

Version	Avg. SD of hex. sums	Avg. fitness gain	Avg. # of fitness calculation	Avg. gain per average calculation
2-Opt without tabu search	0.9425	401.68	104929.3	0.003828
2-Opt with tabu search	0.6452	402.19	844419.2	0.000476
Consecutive without tabu search	0.9866	401.59	3073.6	0.130660
Consecutive with tabu search	0.6840	402.13	3871.9	0.103858

population. We call it *nearby search*. This scheme can be regarded as a three level optimization; the GA is used for large-scale search, the nearby search is used for medium-scale search, and the local optimization is used for small-scale search.

The overall structure of the proposed hybrid genetic algorithm is described in Figure 6. In the main loop, offsprings are generated and local-optimized as in a typical hybrid GA. Then, the nearby search is applied to all the chromosomes in the population.

The details of the algorithm are as follows.

- Problem Encoding: Each locus is numbered top to bottom in zigzag order. Figure 7 shows indices of vertices in the 30-node HTP. The zigzag order was chosen to reflect more geographic localities of genes in the chromosome.
- Fitness and Selection: The fitness of a chromosome is defined as

$$\text{Fitness} = -\sigma^2$$

where σ^2 is the variance of hexagonal sums. Note that the fitness values are negative. Selection is done based on the rank of the *effective fitness* that is slightly modified from the above fitness. The effective fitness is described in Section 4.2.

- Population: We set the population size to be 512.
- Local optimization: Local optimization used in our algorithm was explained in Section 3.

```

create initial population  $\{c_0, c_1, \dots, c_{N-1}\}$ ;
for ( $i \leftarrow 0$ ;  $i < N$ ;  $i++$ )
     $c_i \leftarrow \text{localOptimize}(c_i)$ ;

while (stop condition is not satisfied) {
    for ( $i \leftarrow 0$ ;  $i < \frac{N}{2}$ ;  $i++$ ) {
        choose two parents  $p_1, p_2$ ;
         $o_i \leftarrow \text{crossover}(p_1, p_2)$ ;
         $o_i \leftarrow \text{mutation}(o_i)$ ;
         $o_i \leftarrow \text{localOptimize}(o_i)$ ;
    }
    replace  $\frac{N}{2}$  chromosomes in population with  $o_0, o_1, \dots, o_{\frac{N}{2}-1}$ ;

    for ( $i \leftarrow 0$ ;  $i < N$ ;  $i++$ ) {
         $n_i \leftarrow \text{modification}(c_i)$ ;
         $n_i \leftarrow \text{localOptimize}(n_i)$ ;
        if ( $\text{fitness}(n_i) \geq \text{fitness}(c_i)$ )  $c_i \leftarrow n_i$ ;
    }
}
return the best solution;

```

Fig. 6. Overall structure of proposed hybrid GA

- Replacement: At each generation, the worst half of the population is replaced.
- Crossover: Two-point crossover is used.
- Mutation: Each gene value is increased by one with probability $\frac{1}{3}$ or decreased by one with probability $\frac{1}{3}$.
- Modification: This is used in the nearby search and described in Section 4.1.
- Repair: Since crossover or mutation can generate infeasible solutions (i.e., one number is assigned to more than one node), repair is needed. In repair process, gene values are replaced with their sorted orders. If there are genes with the same value, they are randomly reordered. Figure 8 shows an example of the process.
- Stop Condition: The algorithm stops when an optimum is found or the number of generations reaches a threshold.

4.1 Nearby Search

Since the HTPs have significantly rugged landscapes as mentioned before, traditional GAs do not solve well the HTPs with dozens or more nodes, even if they are combined with some local optimization heuristics. We adopted another search heuristic which focuses on the space around the solutions in the current population. This heuristic is similar to the idea of iterated local search [10]. But our approach is based on population and the search is applied to each chromosome once at each generation. From the viewpoint of high-quality solutions that

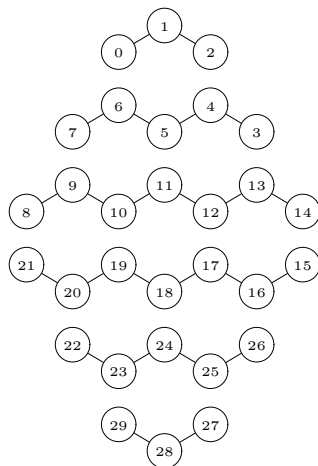


Fig. 7. Problem encoding

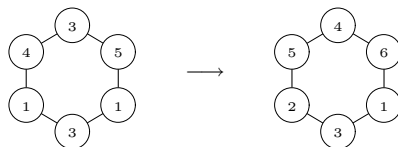


Fig. 8. An example repair process

survive over a number of generations, the heuristic is similar to iterated local search or large-step Markov chain [11].

Nearby search is applied to every chromosome in the population at the end of each generation. Each chromosome is modified and the local optimization heuristic is applied to it. If the fitness of the newly generated chromosome is not worse than the original one, the original chromosome is replaced by the new one.

We devised a guided modification which uses problem-specific knowledge. We define the *error* for a gene k as follows:

$$\varepsilon(k) = \sum_{h \in H_k} [s(h) - \mu],$$

where H_k is the set of hexagons that include k , $s(h)$ is the hexagonal sum of hexagon h , and μ is the mean of hexagonal sums.

The modification decreases the value of the gene whose error is the largest to positive and increases the value of the gene whose error is the largest to negative. It compulsorily alleviates the worst defect caused by those genes and lets the local optimization heuristic to cope with the change in hexagonal sums afterward.

Since the perturbation of the applied modification is weak, many of the modified solutions return to the original ones by the local optimization heuristic. In

our algorithm, we look up the tabu list of the original solutions as well as that of the new ones so that the new chromosomes are different from the original ones.

Experimental results for the nearby search are presented in Section 4.3.

4.2 Aging

Since the HTPs have very rugged landscapes, most of the population is often occupied by local optima that contribute little to find the global optima. We adopted a kind of aging mechanism [12] [13] in which the fitness of a chromosome decreases over the generations of the genetic algorithm. The age of a newly generated solution is set to zero. At each generation, the age increases by one. The effective fitness considering the aging is defined as follows:

$$\text{Effective Fitness} = \text{Fitness} - \frac{1}{10H} \cdot \text{Age}$$

where H is the number of hexagons in a graph. The coefficient $\frac{1}{10H}$ is chosen by the following reasons. Since low-quality solutions are easily replaced by better solutions even without aging, we focused on high-quality solutions. High-quality solutions that can be found in a mature population usually have the property that most of its hexagonal sums have the same value and only a few of the sums are larger or smaller by one. For a chromosome having i hexagons with hexagonal sum $S + 1$, j hexagons with hexagonal sum $S - 1$ and $H - i - j$ hexagons with hexagonal sum S , the variance of its hexagonal sums is

$$\sigma^2 = \frac{i+j}{H} - \frac{(i-j)^2}{H^2}.$$

For small i and j , $\sigma^2 \approx \frac{i+j}{H}$. So the variances of hexagonal sums of high-quality solutions are discrete and the interval between them is about $\frac{1}{H}$. (The property holds even for the hexagons with hexagonal sum $S \pm 2$.) Since the fitness is defined to be $-\sigma^2$, we chose the coefficient $\frac{1}{10H}$ so that the fitness of a solution decreases at intervals of $\frac{1}{H}$ every ten generations. Thus, if the fitness of a chromosome does not improve by the nearby search, it is eventually replaced by a new chromosome.

4.3 Performance Improvement by Nearby Search and Aging

Table 3 shows the effects of nearby search and aging. The 96-node HTP was used for the test. We set four versions of GA depending on the uses of nearby search and aging. All the versions except version I, which is a standard hybrid GA, found optimal solutions on every run.

When we compare version I and Version II, we can see the usefulness of nearby search. Version II was significantly better and faster than Version I. Version III was also similarly better and faster than Version I. Version IV, which combines nearby search and aging, showed further improvement over version II and III.

Table 3. The effect of the nearby search and aging for the 96-node HTP. The maximal number of generation was set to be 10,000. SD and CV means standard deviation and coefficient of variation, respectively.

Version	Nearby Search	Aging	SD of hex. sums		CPU time (seconds)	
			Best	Avg.	Avg.	CV(%)
I	N	N	0.0000	0.1524	2143.69	18.76
II	Y	N	0.0000	0.0000	170.73	138.53
III	N	Y	0.0000	0.0000	257.96	55.16
IV	Y	Y	0.0000	0.0000	73.67	50.48

Table 4. Experimental results for various instances. Generation 0 means that the optimum is found in initial population. SD and CV mean standard deviation and coefficient of variation, respectively.

Size	SD of hex. sums			Generations		CPU time (seconds)		Time per generation
	Best	Avg.	SD	Avg.	CV(%)	Avg.	CV(%)	
30	0.0000	0.0000	0.0000	0.00 (SD=0.00)		0.12	17.70	‡
48	0.0000	0.0000	0.0000	0.59	107.99	0.35	24.98	‡
70	0.0000	0.0000	0.0000	14.90	80.82	3.61	64.82	0.28990
96	0.0000	0.0000	0.0000	253.41	57.31	79.40	55.01	0.32395
126	0.0000	0.0000	0.0000	1424.19	49.19	679.13	47.71	0.48207
160	0.0000	0.0047	0.0239	17480.10	68.89	13782.66	67.35	0.75851

‡ We could not compute reliable values for these problem sizes since the numbers of generations are too small for these problems.

5 Experimental Results

Our official version is the GA with the nearby search and aging mechanism. The algorithm was implemented in C and run on Pentium4 1.5 GHz. Table 4 shows the experimental results of our algorithm for various instances. The result of the 160-node HTP is from 50 runs and those of the rest are from 100 runs. We found the optimal solutions for up to the 160-node HTP in reasonable time. Also, we could find some optimal solutions of the 198-node HTP in a few days. The average time to find the optimal solutions increases exponentially with respect to the problem size. For the problem of sizes 30, 48, 70, 96, and 126, the algorithm found optimal solutions on every run. Figure 9 shows one of the solutions that we found for the 198-node HTP.

6 Conclusion

We proposed an efficient local optimization heuristic for the hexagonal tortoise problem using some problem-specific knowledge. The hybrid GA was further improved by the addition of nearby search. The proposed GA can be regarded as a three level optimization. Another notable aspect of our algorithm is the

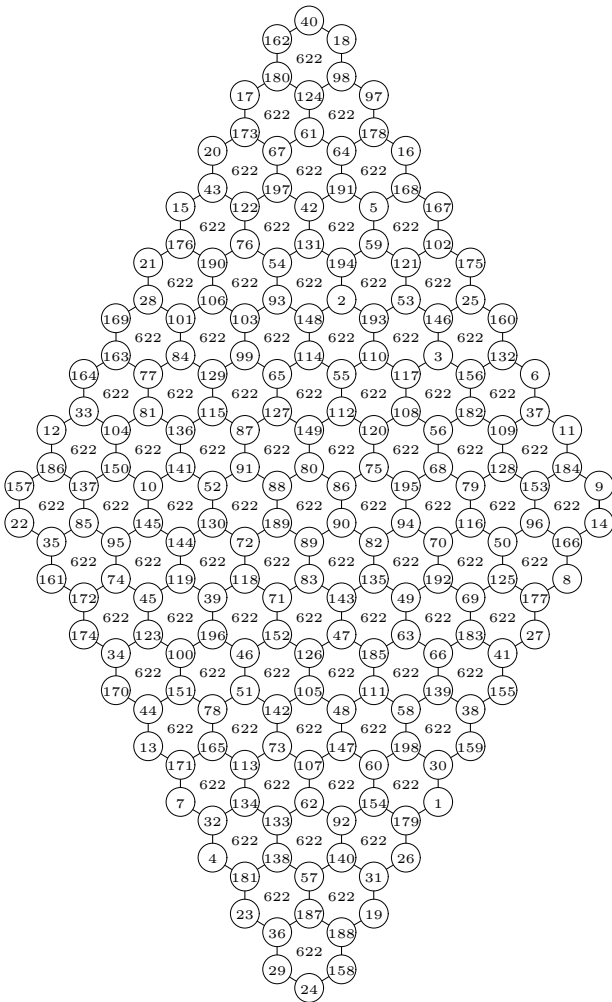


Fig. 9. An optimal solution of the 198-node hexagonal tortoise

aging mechanism. It helps the GA to weed out local optima that contribute little to find global optima.

The proposed GA found optimal solutions for up to the 198-node hexagonal tortoise problem. This seems to be a fairly good achievement considering the extreme ruggedness of the problem. Future studies may include investigating larger size of HTPs, probably with parallel computing, applying the developed ideas to other combinatorial optimization problems, and further refining the GA.

Acknowledgements. This work was partly supported by Optus Inc. and Brain Korea 21 Project. The RIACT at Seoul National University provided research facilities for this study.

References

1. S. J. Choi. *Gusuryak (a reprint)*. Shungshin Womens University Press, Seoul, Korea, 1983.
2. Y. H. Jun. Mysteries of mathematics: The order of the universe hidden in numbers. *Dong-A Science*, 14(7):68–77, 1999.
3. S. K. Lee, D. I. Seo, and B. R. Moon. A hybrid genetic algorithm for optimal hexagonal tortoise problem. In *Genetic and Evolutionary Computation Conference*, page 689, 2002.
4. D. E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison Wesley, 1989.
5. S. Forrest and M. Mitchell. Relative building-block fitness and the building-block hypothesis. In *Foundations of Genetic Algorithms*, volume 2, pages 109–126. Morgan Kaufmann, 1993.
6. J. Horn and D. E. Goldberg. Genetic algorithm difficulty and the modality of fitness landscapes. In *Foundations of Genetic Algorithms*, volume 3, pages 243–270. Morgan Kaufmann, 1995.
7. T. Jones and S. Forrest. Fitness distance correlation as a measure of problem difficulty for genetic algorithms. In *Sixth International Conference on Genetic Algorithms*, pages 184–192. Morgan Kaufmann, 1995.
8. E. Aarts and J. K. Lenstra, editors. *Local Search in Combinatorial Optimization*. John Wiley & Sons, 1997.
9. F. Glover. Tabu search: Part I,. *ORSA Journal of Computing*, 3(1):190–206, 1977.
10. H. R. Lourenço, O. C. Martin, and T. Stützle. Iterated local search. In F. W. Glover and G. Kochenberger, editors, *Handbook of Metaheuristic*, chapter 11. Kluwer Academic Publisher, 2002.
11. O. Martin, S. W. Otto, and E. W. Felten. Large-step markov chains for the traveling salesman problem. *Complex Systems*, 5:299–326, 1991.
12. Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolutionary Programs*. Springer, 1992.
13. A. Ghosh, S. Tsutsui, and T. Tanaka. Function optimization in nonstationary environment using steady state genetic algorithms with aging of individuals. In *IEEE International Conference on Evolutionary Computation*, pages 666–671, 1998.

Normalization in Genetic Algorithms

Sung-Soon Choi and Byung-Ro Moon

School of Computer Science and Engineering
Seoul National University
Seoul, 151-742 Korea
{sschoi,moon}@soar.snu.ac.kr

Abstract. Normalization is an approach that transforms the genotype of one parent to be consistent with that of the other parent. It is a method for alleviating difficulties caused by redundant encodings in genetic algorithms. We show that normalization plays a role of reducing the search space to another one of less size. We provide insight into normalization through theoretical arguments, performance tests, and examination of fitness-distance correlations.

1 Introduction

In a genetic algorithm, solutions are encoded into chromosomes by an encoding scheme, and the chromosomes are handled by the genetic operators. A fitness function plays a role as a bridge that connects chromosomes and solutions in that it evaluates the qualities of solutions by decoding chromosomes. By the analogy with biology, chromosomes and solutions are called genotypes and phenotypes, respectively.

An encoding scheme determines the relation between genotype space and phenotype space. According to the relations between the two spaces, encoding schemes are classified into three classes. First, one phenotype is represented by one genotype. In this case, the genotype space is one of the minimal spaces that represent the phenotype space without any loss of information. This is in general considered as the most desirable [1] [2]. Representative examples are real-valued encodings in function optimization, function approximation, and so on [3] [4].

Next, several phenotypes are represented by one genotype. Such an encoding scheme can be used to reduce the size of the search space by considering only the important parameters related to a problem. It seems to be suitable for hybrid genetic algorithms in which more important parameters are treated by the genetic process and less important ones are fine-tuned by local optimization methods. In most practical problems, however, various parameters are so intertwined that it is another difficult problem to select more important parameters [5].

Finally, one phenotype is represented by several genotypes. Such encoding schemes are called *redundant encodings*. There are quite a few problems in which it is difficult to represent one phenotype by one genotype using traditional encoding schemes due to the characteristics of the problems. Redundant encodings are still being used in most of those problems.

There are two representative groups of problems belonging to the third class. One is the group of *grouping problems* [2]. These are commonly concerned with partitioning a given item set into mutually disjoint subsets. Examples belonging to this group include k -way graph partitioning [6] [7] [8], graph coloring, bin packing, and workshop layouting [2]. The other is the group of the structural optimization problems. These are concerned with finding the optimal structures satisfying given constraints. In the problems, there are often several other solutions structurally equivalent to a solution. Example problems include sorting network optimization [9], neural network optimization [10], and RNA structure prediction [11].

Recently, there were discussions about the benefits of redundancy in encoding in evolutionary algorithms [12] [13] [14]. In particular, Shipman, Shackleton, and Ebner [12] [13] showed that some redundant encodings are useful to help mutation-based hill-climbing search by constructing many different neutral paths in the genotype space that enable moves at the same fitness level. However, the tackled problems were of small sizes and only mutation operators were considered. As mentioned in the next section, in the cases that the problems are of large sizes and highly epistatic, redundant encodings may lead to severe losses in the search powers of traditional crossovers (such as 1-point, multi-point, and uniform) and accordingly of genetic algorithms with them as well.

In order to alleviate the problems caused by redundant encodings, a number of approaches were proposed. In some problems, adaptive crossovers were developed that recombine parents in terms of phenotypes [6] [15] [16]. Van Hoyweghen *et al.* [17] pointed out problems in a certain class of redundant encodings and presented some solutions to them. *Normalization* transforms the genotype of a parent to another genotype to be consistent with the other parent. There were a few successful studies that used normalization [7] [8] [9].

Among those approaches, we focus on normalization. We review the normalization methods used in a few problems and state their merits associated with search space reduction. Finally, we support its validity with experimental results.

The rest of this paper is organized as follows. In Section 2, we argue that redundant encodings lead to severe losses of search power in genetic algorithms, in particular, with respect to traditional crossovers. In Section 3, we describe normalization as a method of alleviating the problems by redundant encodings and show that the normalization plays a role of reducing the search space to another space of less size. In Section 4, we review the redundant encodings and normalization methods used in some problems, and examine the effects of normalization on the search spaces by experiments. Finally, we make our conclusions in Section 5.

2 Drawbacks of Redundant Encoding

In general, *respectfulness* and *combination* are considered to be two major features that crossovers have to possess. Respectfulness means the property that the alleles common to the parents are transmitted to offsprings. The concept

of respectfulness stands on the basis of the commonality hypothesis: “schemata common to above-average solutions are above average” [18]. Many papers referred to the importance of respectfulness in the crossover design [1] [18] [19] [20]. Combination means the property of producing good schemata of higher orders by recombining good schemata of low orders in parents. The concept of combination is based on the building block hypothesis [21] and has received heavy and repeated emphasis in the literature [1] [19] [22] [23].

A redundant encoding represents an optimal phenotype by multiple genotypes and thus the genotype space becomes multimodal [21]. The difficulty of multimodal problems with respect to genetic algorithms has been pointed out in the literature [21] [24] [25]. Similarly to the optimal phenotypes, high-quality schemata are redundantly represented by the encoding: a phenotype schema in an above-average phenotype is represented by multiple genotype schemata that are equivalent in terms of phenotype.¹ On the other hand, traditional crossovers such as 1-point crossover and uniform crossover operate on the genotype space regardless of phenotypes. The redundant encoding is harmful to those crossovers in both aspects of respectfulness and combination, as described below.

- **Respectfulness:** Most traditional crossovers are basically respectful in the genotype space (e.g., 1-point, multi-point, and uniform crossovers). However, they are not respectful in terms of phenotypes. Those crossovers do not preserve “equivalent” genotype schemata of high-quality in parents if they have different representations. The probability that a common phenotype schema in parents is transmitted to the offspring decreases as the schema has more different representations in the genotype space.
- **Combination:** In highly epistatic problems,² a low-order schema in a genotype may have a positive or negative effect on the quality of the genotype, depending on the alleles on the other loci of the genotype, i.e., the genotype context [26]. Traditional crossovers recombine schemata in parents regardless of the genotype context. The change in the context may diminish the longevity of high-quality schemata with respect to crossover. For this reason, it is difficult to build desirable schemata of higher orders by juxtaposition. As a phenotype schema has more different genotypes, the probability that the schema emerges decreases.

Thus, in the problems of large sizes and high epistases, it is hard for genetic algorithms with traditional crossovers to cope with the genotype spaces expanded by redundant encodings. Traditional crossovers cannot handle phenotype schemata effectively. Normalization methods transform a given parent to another genotype so that the genotype contexts of the parents are as similar as possible in crossover. As mentioned in the next section, these methods help

¹ A phenotype schema in a phenotype is implicitly defined to be a partial property contained in the phenotype.

² Most of the problems mentioned in Section 1, in which redundant encodings are used, belong to this class.

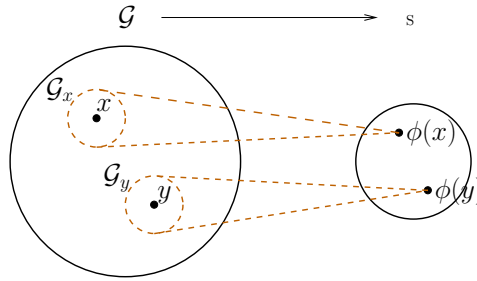


Fig. 1. Genotype-phenotype space relationship

traditional crossovers overcome the problems related to respectfulness and combination. They play a role of reducing the search space by considering multiple genotypes corresponding to a phenotype as a conceptual group.

3 Normalization and Space Reduction

Given the phenotype space $\mathcal{S}$ for a problem, a redundant encoding determines the genotype space $\mathcal{G}$ and a function $\phi : \mathcal{G} \longrightarrow \mathcal{S}$. For a genotype $x \in \mathcal{G}$, consider the subset $\mathcal{G}_x$ of $\mathcal{G}$ as follows:

$$\mathcal{G}_x = \{u \in \mathcal{G} | \phi(u) = \phi(x)\}. \quad (1)$$

$\mathcal{G}_x$ corresponds to the set of genotypes that share the phenotype of x . We call $\mathcal{G}_x$ the coset (neutral set) of x . The above definition of cosets gives an equivalence relation among the genotypes and $\mathcal{G}$ is partitioned by these cosets. We denote the set of cosets by $\bar{\mathcal{G}}$, and then $\bar{\mathcal{G}}$ and $\mathcal{S}$ are in one-to-one correspondence (Figure 1).

As mentioned before, it is necessary to recombine parents in a phenotype level in order to alleviate the problems of redundant encodings. Then, given parents $x, y \in \mathcal{G}$, how do we recombine the corresponding phenotypes, $\phi(x)$ and $\phi(y)$? The essence of normalization is to view the genotypes in a coset as a phenotype. Such a view allows that x is substituted by an element in $\mathcal{G}_x$ that is the most consistent to y . That is, given $\phi(x)$ and $\phi(y)$, normalization operators play a role of selecting the most appropriate $x' \in \mathcal{G}_x$ and returning it to the crossover operator so that high-quality schemata are well preserved and combined.

More formally, suppose a distance function $d : \mathcal{G} \times \mathcal{G} \longrightarrow \mathbb{R}$ and a consistency measure $s : \mathcal{G} \times \mathcal{G} \longrightarrow \mathbb{R}$ defined on genotype pairs ($\mathbb{R}$ denotes the set of real numbers). Given parents $x, y \in \mathcal{G}$, the normalization operator transforms x to $x' \in \mathcal{G}_x$ so that $s(\cdot, y)$ is maximized. Then the crossover recombine the two genotypes x' and y . In other words, the distances between genotypes are redefined by normalization and consequently the landscape of search space alters. Since $\bar{\mathcal{G}}$ and $\mathcal{S}$ are in one-to-one correspondence as mentioned, the genotype space is reduced to the phenotype space by normalization from the viewpoint of crossover (Figure 2).

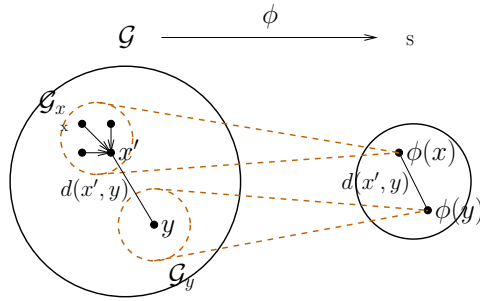


Fig. 2. Space reduction

Space reduction by normalization modifies the structure of the landscape of search space. Thus, it changes the difficulty of problem with respect to genetic algorithms and so affects the performances of the genetic algorithms. The change in the difficulty has much to do with the consistency measure s , and selecting a suitable measure s is crucial in the normalization operator design. It suggests that normalization is more than simply decreasing the genotype distance (e.g., Hamming distance) between parents.

To get insight into the effect of normalization on search space, we consider fitness distance correlation (FDC) [27], which is a popular measure for predicting problem difficulty [28] [29], in the next section. As seen in the section, normalization leads to considerable increases of FDC values.

Normalization gives rise to the moves at the same level of fitness in the genotype space, so called, the neutral walks [30] by replacing parents with the genotypes in their cosets. It is related to the previous studies in the aspect of using the neutral paths created by redundancy [12] [13]. However, normalization attempts to alleviate the negative effects of neutral paths.

4 Normalization Examples

In this section, we review the normalization methods used in two problems. In addition, we consider the effects of normalization on space reduction and analyze the landscapes of the search spaces before and after normalization. Experimental results with respect to normalization are also provided. The problems that we consider in this section are the graph partitioning problem and the sorting network problem, which belong to the groups of grouping problems and structural optimization problems, respectively.

4.1 Graph Partitioning Problem

Given a graph $G = (V, E)$, where V represents the set of vertices and E represents the set of edges, k -way partitioning is grouping the vertex set V into k

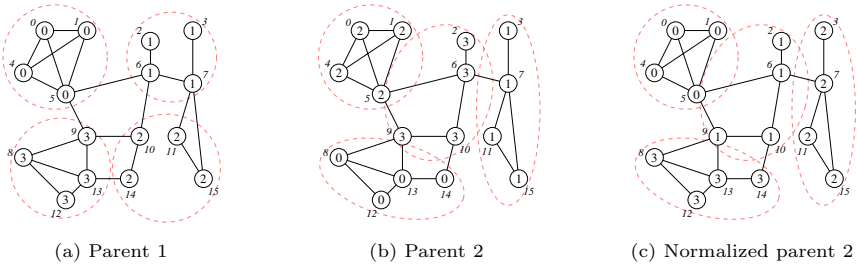


Fig. 3. A normalization example in 4-way graph partitioning

disjoint subsets. In particular, two-way partition is called bisection or bipartition. In a k -way partition, the total number of edges whose endpoints belong to different subsets is called cut size. A k -way partition is said to be balanced if the difference of cardinalities between the largest and the smallest subsets is at most one. The k -way partitioning problem in this section is to find a balanced k -way partition with minimal cut size.

In the k -way partitioning problem, the k -ary encoding scheme, in which k subsets are represented by the integers from 0 to $k - 1$, has been generally used [6] [7] [8] [15] [31]. In this case, a phenotype (k -way partition) is represented by $k!$ different genotypes according to the numbering of the subsets.

In the k -way partitioning problem, a normalization method was used in [8], which is based on the adaptive crossovers proposed in [6] and [15]. The method can be described as follows. First, it selects a partition from each parents so that they contain the most vertices in common, and then it assigns the partitions a common number. It repeats this process with the other partitions and remaining partition numbers. Figure 3 shows an example of normalization. The parent 2 was transformed to be as consistent as possible to the parent 1. Figure 4 shows an example crossover operator with and without normalization. The same parents as in Figure 3 were used. We can see that the offspring was too much perturbed without normalization; good characteristics of the parents were inherited little to the offspring.

Tables 1 and 2 show the experimental results of the cases without normalization (Ordinary) and with normalization (Normalization) for the bipartitioning problem and the 32-way partitioning problem, respectively. The experiments were performed for the instances from Johnson's benchmark [7]. In the tables, Ed and ρ indicate the average hamming distance and the fitness distance correlation coefficient (FDC) value [27], respectively, of 10,000 local optima. To get the local optima, the Kernighan-Lin heuristic [7] and the heuristic in [31] were used for bipartitioning and 32-way partitioning, respectively. Best, Avg., and CPU represent the best cut size, average cut size, and average running time over 100 runs of a hybrid genetic algorithm.

In both problems, the average distances among local optima considerably decreased by normalization. In the 32-way partitioning problem, the distances before normalization were close to the chromosome lengths, which shows that

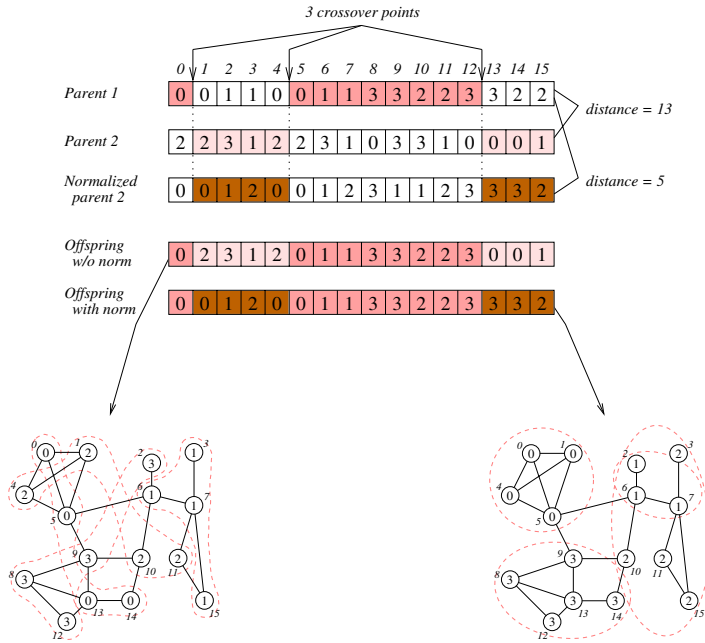


Fig. 4. A normalization example in 4-way graph partitioning

common schemata are rare in genotypes. The distances were dramatically reduced after normalization. In another experiment, we observed that the average distances among local optima were almost the same as those among random solutions in the case without normalization. However, in the case with normalization, we observed that the average distances among local optima were much less than those among random solutions.³ This implies that the local optima got closer to one another in the space after normalization.

The considerable increase of FDC values by normalization are observed from the tables. This indicates that the landscapes of the search spaces altered by normalization, so that the difficulty of search with respect to genetic algorithms diminished. As expected, normalization led to considerable improvement in performance. Figure 5 shows the change of landscape of an instance G500.005 in the 32-way partitioning problem. The x -axis represents the distance from the best out of 10,000 local optima. Before normalization, the solution quality had little to do with the distance from the best local optimum. After normalization, they showed a strong correlation.

³ Experiments showed that, for bipartitioning problem, the average distances among 100,000 random solutions were 250.00 and 241.02 without and with normalization for 500-node instances and 500.00 and 487.32 for 1000-node instances. For 32-way partitioning problem, they were 484.37 and 432.58 for 500-node instances and 968.75 and 899.54 for 1000-node instances.

Table 1. Comparison of experiments in bipartitioning

Graph	Ordinary					Normalization				
	Ed	ρ	Best	Avg.	CPU	Ed	ρ	Best	Avg.	CPU
G500.005	249.99	0.003	51	52.58	0.82	213.22	0.343	50	52.03	0.89
G500.04	250.05	0.012	1744	1745.65	4.51	215.52	0.292	1744	1745.77	4.35
G1000.0025	499.96	-0.018	97	100.57	2.39	445.01	0.398	95	98.18	2.78
G1000.02	500.00	0.005	3382	3385.65	23.54	447.83	0.487	3382	3385.18	22.60
U500.05	249.94	0.009	2	4.80	1.09	215.82	0.312	2	3.48	1.24
U500.40	250.23	-0.004	412	412.00	1.68	33.38	0.867	412	412.00	1.24
U1000.05	499.90	-0.005	1	5.23	2.77	447.85	0.290	1	2.49	3.24
U1000.40	498.24	0.030	737	737.00	9.88	249.13	0.754	737	737.00	7.85

Table 2. Comparison of experiments in 32-way partitioning

Graph	Ordinary					Normalization				
	Ed	ρ	Best	Avg.	CPU	Ed	ρ	Best	Avg.	CPU
G500.005	483.59	0.019	181	184.92	163.05	300.61	0.621	179	181.63	166.97
G500.04	484.36	0.013	4039	4049.56	1410.56	358.82	0.412	4037	4045.03	1197.93
G1000.0025	968.15	0.009	316	328.55	666.85	723.75	0.472	312	321.32	630.88
G1000.02	968.78	0.030	7824	7839.46	4500.29	813.99	0.303	7818	7832.49	4350.69
U500.05	484.00	0.005	113	120.86	278.33	170.76	0.602	113	116.59	290.75
U500.40	484.04	0.016	5363	5391.38	1197.35	173.56	0.338	5359	5380.15	1161.56
U1000.05	968.47	0.021	127	143.74	702.39	525.48	0.610	117	126.04	1199.85
U1000.40	969.04	0.006	7406	7425.16	3405.49	225.09	0.672	7398	7416.85	3218.18

4.2 Sorting Network Problem

A sorting network is a hardware sorting logic in which the comparisons and exchanges of data are carried out in a prescribed order. A sorting network is composed of buses and a number of homogeneous comparators, where each comparator $c(a,b)$ performs the elementary operation that compares the a^{th} and b^{th} buses; if the values are in order, ignore them, otherwise exchange them. We call a sorting network for n inputs an n -bus sorting network. The sorting network problem is to find an n -bus sorting network with minimal comparators for a given input size n .

In the sorting network problem, a sorting network is generally represented by a sequence of comparators [9] [32] [33] [34]. Choi and Moon [9] defined sorting network isomorphism in terms of construction scheme and showed that the sequence enumeration is a redundant encoding. Contrary to the case of the graph partitioning problem, the number of genotypes for each phenotype may be different from phenotype to phenotype.

Figure 6 shows an example of two isomorphic networks. Between the two networks, there is a permutation $\begin{pmatrix} 0 & 1 & 2 & 3 & 4 & 5 \\ 5 & 4 & 3 & 2 & 1 & 0 \end{pmatrix}$ of bus indices. Although they are quite different in appearance, they are the same in function.

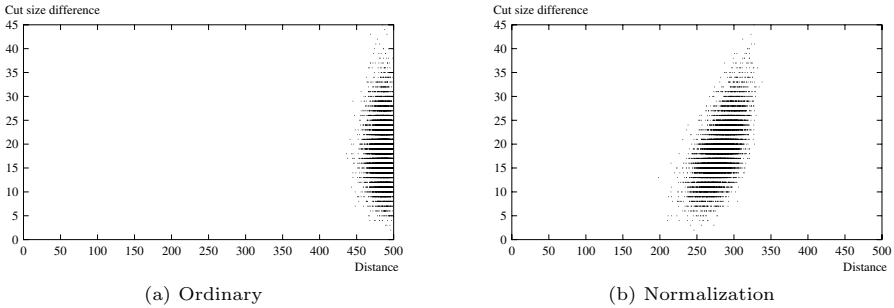


Fig. 5. Normalization effect in 32-way partitioning



Fig. 6. Two isomorphic networks

In sorting networks, it is known that the comparators in the front part (left side) strongly affect the subsequent comparators [34] [35]. This indicates that the comparators in the front part have stronger effects on the qualities of networks than others. In this context, a normalization method was proposed in [9] that transforms a parent so that the front part is as consistent as possible with the other parent.

For example, consider the parents 1 and 2 in Figure 7. Although the subnetworks with the first eight comparators in the two networks are totally different, they are isomorphic. Figure 7(c) shows the network after the parent 2 is normalized with respect to the parent 1. The actual crossover is conducted between the parent 1 and the normalized parent 2.

For 16-bus sorting network problem, we performed similar experiments to those in the graph partitioning problem. In this case, the local heuristic and hybrid genetic algorithm in [9] were used. Figure 8 shows the change of the landscapes obtained from 10,000 local optima generated by the local heuristic.

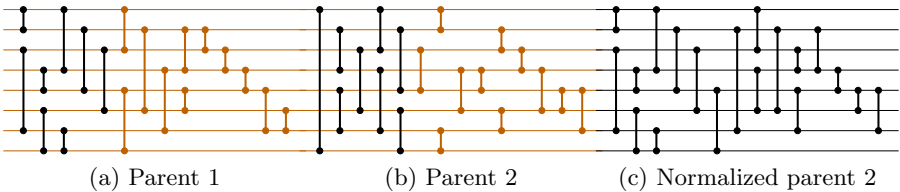


Fig. 7. A normalization example in the 8-bus sorting network problem

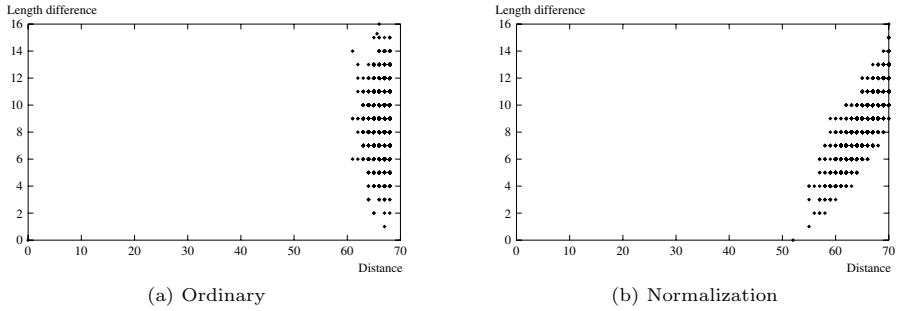


Fig. 8. Normalization effect in 16-bus sorting network problem

Like in the graph partitioning problem, the FDC value considerably increased by normalization (from 0.079 to 0.847) and significant improvement in performance was observed. (Average running time to find 60-comparator networks known as the best decreased from 1,316 to 717 minutes.)

5 Conclusion

We argued that redundant encodings, in which a phenotype is represented by multiple genotypes, lead to severe loss of search power in genetic algorithms, in particular, with respect to traditional crossovers. As a method of alleviating the problem of redundant encodings, normalization considers the genotypes corresponding to a phenotype as a group. We also showed that the normalization alters the search space.

We reviewed the redundant encodings and normalization methods used in the graph partitioning problem and sorting network problem. Through some experiments, we examined the change of search space, the change in problem difficulty, and the performance improvement with respect to genetic algorithms. In both problems, the normalization methods decreased the problem difficulties and led to remarkable improvement in performance.

We are currently working on the effective design of normalization methods for other problems such as neural network optimization and magic tortoise optimization.

Acknowledgements. The authors thank Jong-Pil Kim and Yong-Hyuk Kim for insightful discussions about graph partitioning problem. This work was partly supported by KOSEF through the Statistical Research Center for Complex Systems at Seoul National University (SNU) and Brain Korea 21 Project. The RIACT at Seoul National University provided research facilities for this study.

References

1. N. J. Radcliffe. Forma analysis and random respectful recombination. In *Fourth International Conference on Genetic Algorithms*, pages 222–230, 1991.
2. E. Falkenauer. *Genetic Algorithms and Grouping Problems*. Wiley, 1998.
3. J. Yang, J. Horng, and C. Kao. A continuous genetic algorithm for global optimization. In *Seventh International Conference on Genetic Algorithms*, pages 230–237. Morgan Kaufmann, 1997.
4. Y. K. Kwon and B. R. Moon. A genetic hybrid for CHF function approximation. In *Genetic and Evolutionary Computation Conference*, pages 1119–1125, 2002.
5. M. J. Martin-Bautista and M. A. Vila. A survey of genetic feature selection in mining issues. In *Congress on Evolutionary Computation*, pages 1314–1321, 1999.
6. G. Laszewski. Intelligent structural operators for the k -way graph partitioning problem. In *Fourth International Conference on Genetic Algorithms*, pages 45–52, 1991.
7. T. N. Bui and B. R. Moon. Genetic algorithm and graph partitioning. *IEEE Trans. on Computers*, 45(7):841–855, 1996.
8. S. J. Kang and B. R. Moon. A hybrid genetic algorithm for multiway graph partitioning. In *Genetic and Evolutionary Computation Conference*, pages 159–166, 2000.
9. S. S. Choi and B. R. Moon. Isomorphism, normalization, and a genetic algorithm for sorting network optimization. In *Genetic and Evolutionary Computation Conference*, pages 327–334, 2002.
10. C. Igel and P. Stagge. Effects of phenotypic redundancy in structure optimization. *IEEE Trans. on Evolutionary Computation*, 6(1):74–85, 2002.
11. P. Schuster. Molecular insights into evolution of phenotypes. In J. P. Crutchfield and P. Schuster, editors, *Evolutionary Dynamics — Exploring the Interplay of Accident, Selection, Neutrality and Function*. Oxford Univ. Press, 2002.
12. R. Shipman. Genetic redundancy: Desirable or problematic for evolutionary adaptation. In *Fourth International Conference on Artificial Neural Networks and Genetic Algorithms*, pages 337–344. Springer-Verlag, 1999.
13. M. A. Shackleton, R. Shipman, and M. Ebner. An investigation of redundant genotype-phenotype mappings and their role in evolutionary search. In *Congress on Evolutionary Computation*, pages 493–500, 2000.
14. K. Weicker and N. Weicker. Burden and benefits of redundancy. In *Foundations of Genetic Algorithms*, volume 6, pages 313–333. Morgan Kaufmann, 2001.
15. H. Mühlenbein. Parallel genetic algorithms in combinatorial optimization. In *Computer Science and Operations Research: New Developments in Their Interfaces*, pages 441–453, 1992.
16. R. Dorne and J. K. Hao. A new genetic local search algorithm for graph coloring. In *Parallel Problem Solving from Nature*, pages 745–754. Springer-Verlag, 1998.
17. C. Van Hoyweghen, B. Naudts, and D. E. Goldberg. Spin-flip symmetry and synchronization. *Evolutionary Computation*, 10(4):317–344, 2002.
18. S. Chen. *Is the Common Good ? A New Perspective Developed in Genetic Algorithms*. PhD thesis, Robotics Institute, Carnegie Mellon University, 1999.
19. G. Syswerda. Uniform crossover in genetic algorithms. In *Third International Conference on Genetic Algorithms*, pages 2–9, 1989.
20. S. Chen and S. Smith. Commonality and genetic algorithms. Technical Report CMU-RI-TR-96-27, Robotics Institute, Carnegie Mellon University, 1996.

21. D. E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison Wesley, 1989.
22. L. Davis. *Handbook of Genetic Algorithms*. Van Nostrand Reinhold, 1991.
23. R. A. Watson and J. B. Pollack. Recombination without respect: Schema combination and disruption in genetic algorithm crossover. In *Genetic and Evolutionary Computation Conference*, 2000.
24. S. Forrest and M. Mitchell. Relative building-block fitness and the building-block hypothesis. In *Foundations of Genetic Algorithms*, volume 2, pages 109–126. Morgan Kaufmann, 1993.
25. J. Horn and D. E. Goldberg. Genetic algorithm difficulty and the modality of fitness landscapes. In *Foundations of Genetic Algorithms*, volume 3, pages 243–270. Morgan Kaufmann, 1995.
26. S. A. Kauffman. Adaptation on rugged fitness landscapes. In D. Stein, editor, *Lectures in the Sciences of Complexity*, pages 527–618. Addison Wesley, 1989.
27. T. Jones and S. Forrest. Fitness distance correlation as a measure of problem difficulty for genetic algorithms. In *Sixth International Conference on Genetic Algorithms*, pages 184–192. Morgan Kaufmann, 1995.
28. P. Merz and B. Freisleben. Fitness landscapes, memetic algorithms and greedy operators for graph bi-partitioning. *Evolutionary Computation*, 8(1):61–91, 2000.
29. P. Merz and B. Freisleben. Fitness landscape analysis and memetic algorithms for the quadratic assignment problem. *IEEE Trans. on Evolutionary Computation*, 4(4):337–352, 2000.
30. M. Huynen. Exploring phenotype space through neutral evolution. *Journal of Molecular Evolution*, 43:165–169, 1996.
31. J. P. Kim and B. R. Moon. A hybrid genetic search for multi-way graph partitioning based on direct partitioning. In *Genetic and Evolutionary Computation Conference*, pages 408–415, 2001.
32. W. D. Hillis. Co-evolving parasites improve simulated evolution as an optimization procedure. In C. Langton, C. Taylor, J. D. Farmer, and S. Rasmussen, editors, *Artificial Life II*. Addison Wesley, 1992.
33. G. L. Drescher. Evolution of 16-number sorting networks revisited. Unpublished manuscript, 1994.
34. S. S. Choi and B. R. Moon. A hybrid genetic search for the sorting network problem with evolving parallel layers. In *Genetic and Evolutionary Computation Conference*, pages 258–265, 2001.
35. S. S. Choi and B. R. Moon. A graph-based approach to the sorting network problem. In *Congress on Evolutionary Computation*, pages 457–464, 2001.

Coarse-Graining in Genetic Algorithms: Some Issues and Examples

Andrés Aguilar Contreras¹, Jonathan E. Rowe², and Christopher R. Stephens³

¹ Instituto de Investigación en Matemáticas Aplicadas y Sistemas, UNAM, Circuito Escolar, Ciudad Universitaria, México D.F. 04510

`andresac@uxmcc2.iimas.unam.mx`

² School of Computer Science, University of Birmingham, Birmingham B15 2TT, Great Britain

`J.E.Rowe@cs.bham.ac.uk`

³ Instituto de Ciencias Nucleares, UNAM, Circuito Exterior, A. Postal 70-543, México D.F. 04510.

`stephens@nuclecu.unam.mx`

Abstract. Following the work of Stephens and coworkers on the coarse-grained dynamics of genetic systems, we work towards a possible generalisation in the context of genetic algorithms, giving as examples schemata, genotype-phenotype mappings, and error classes in the Eigen model. We discuss how the dynamics transforms under a coarse-graining, comparing and contrasting different notions of invariance. We work out some examples in the two-bit case, to illustrate the ideas and issues. We then find a bound for the Selection Weighted Linkage Disequilibrium Coefficient for the two-bit onemax problem.

1 Introduction

To model the exact evolution of a genetic algorithm requires us, in general, to track what happens to each possible individual. For example, if the search space is binary strings of length ℓ , we have evolution equations for each of the 2^ℓ possible strings. It may also be of interest to investigate what happens to certain subsets of individuals. There are three reasons for doing this. Firstly, it may be possible to reduce the number of degrees of freedom in the evolution equations and so make a more tractable model. This is particularly true when modelling the appropriate effective degrees of freedom for the dynamics. Secondly, one may be interested in the evolutionary history of one particular individual, and there may be only a limited number of subsets to which its ancestors could have belonged. Thirdly, of course, one may have some intrinsic interest in a certain subset (for example, it may be a subset of high quality individuals or represent some “kinship” or genetically related group such as the individuals associated with a “niche” or a species).

The idea of tracking subsets is the basis of Holland’s schemata [Holland, 1975], Radcliffe’s forma [Radcliffe, 1992] and Vose’s predicates [Vose, 1991]. More recently, Stephens [Stephens and Waelbroek, 1997] has formally studied evolution equations under this kind of coarse-graining using schemata and extended the analysis to other contexts than evolutionary computation in [Stephens, 2003] and

[Stadler and Stephens, 2003]. Van Nimwegen [van Nimwegen et al., 1997] has modelled the dynamics of GAs on royal-road, and other functions, using approximate coarse-grained models. Rowe [Rowe, 1998] has considered the use of unitation classes as a basis for a coarse-grained model of selection-mutation algorithms. We intend to extend this work by looking at possible generalisations and limitations in the context of genetic algorithms.

In [Stephens and Waelbroek, 1997] it was shown that the dynamical equations governing the evolution of a GA with proportional selection, mutation and one-point crossover was *form invariant* under a coarse graining to schemata. This was later extended [Stephens, 2001] to any selection, mutation and homologous crossover operators. This form invariance was later studied by Vose and Wright [Vose and Wright, 2001] who discussed a more restrictive form of the invariance using the notion of *compatibility* [Vose, 1999] between the coarse graining and the genetic operators.

In the following section, we formally define the idea of coarse-graining. We then consider how the dynamics looks after a coarse graining. The ideas and issues are then illustrated by a series of examples. Finally, we apply coarse-graining to help us estimate what happens to the linkage disequilibrium coefficient in the two-bit onemax problem.

2 Coarse-Grained Dynamics

Let $\Omega = \{0, 1, 2, \dots, n-1\}$ be the search space. In the case of binary strings of length ℓ , we identify each string with an integer under standard binary encoding, and $n = 2^\ell$. We represent a population by a vector $p = (p_0, p_1, \dots, p_{n-1})$ in which p_k is the proportion of individual k in the population. Population vectors are elements of the *simplex*

$$\Lambda = \left\{ p \in \mathbb{R}^n : \sum_k p_k = 1, \text{ and } p_k \geq 0 \text{ for all } k \right\}$$

A coarse-graining of Ω will be a collection of subsets of Ω . Given a fitness function $f : \Omega \rightarrow \mathbb{R}$ we wish to define the fitness of a given subset. Notice, that this will, in general, depend on the details of the population. One can think of this situation as being analogous to a *co-evolutionary* model, in which the fitness of an individual depends on the current population. Let $\mathcal{P}(\Omega)$ denote the *power set* of Ω (that is, the set of all subsets of Ω). Then, formally, we have a function

$$F : \Lambda \longrightarrow (\mathcal{P}(\Omega) \longrightarrow \mathbb{R})$$

defined as

$$F(p)(A) = \frac{\sum_{i \in A} p_i f(i)}{\sum_{i \in A} p_i}$$

That is, given a population $p \in \Lambda$, $F(p)$ is a “fitness function” which assigns fitnesses to subsets of Ω . The fitness of a subset A is the average fitness of elements of A in population p .

Definition 1. Let $\Gamma = \{\gamma_i\} \subseteq \mathcal{P}(\Omega)$ be a collection of subsets of the search space that covers the search space. That is,

$$\bigcup \gamma_i = \Omega$$

We call such a collection a coarse-graining of Ω . For any population $p \in \Lambda$ we can assign a fitness to each element of Γ using the function $F(p)$.

A coarse-graining is non-degenerate if Γ is a partition of the search space

$$i \neq j \implies \gamma_i \cap \gamma_j = \emptyset$$

Notice that the fitness of a subset in a coarse-graining depends on the current population and therefore on time, even if the underlying fitness function f is static.

Examples

- 1) **Schemata.** We can associate a schema with the set of all strings which match it. The set of all schemata forms a highly degenerate cover of the search space, however given an arbitrary choice of string then all the schemata that contain the string forms a new basis of the same dimensionality as the original - the Building Block Basis [Stephens, 2003]. However, unless the fitness function is a constant for all strings matching a given schema, then the fitness of the schema itself will be a dynamic quantity (that is, it will depend on the details of the current population).
- 2) **Genotype-phenotype mappings.** Suppose we have a map $\varphi : \Omega \rightarrow \Phi$ which maps genotypes to phenotypes, where Φ is the space of phenotypes. Fitness is then assessed via an individual's phenotype. That is, there is a function $g : \Phi \rightarrow \mathbb{R}$. The fitness of a genotype is then $f = g \circ \varphi$. We can create a non-degenerate coarse-graining by considering subsets of Ω which map to the same phenotype. That is, for each $i \in \Phi$, set

$$\gamma_i = \{a \in \Omega : \varphi(a) = i\}$$

The fitness of such a subset is constant:

$$\begin{aligned} F(p)(\gamma_i) &= \frac{\sum_{j \in \gamma_i} p_j f(j)}{\sum_{j \in \gamma_i} p_j} \\ &= \frac{\sum_{\varphi(j)=i} p_j g \circ \varphi(j)}{\sum_{j \in \gamma_i} p_j} \\ &= \frac{g(i) \sum_{\varphi(j)=i} p_j}{\sum_{j \in \gamma_i} p_j} \\ &= g(i) \end{aligned}$$

This coarse-graining is natural with respect to selection, as we only need to keep track of what happens to the subsets, without worrying about their detailed composition.

- 2.1 **Unitation.** A particular example of a genotype-phenotype mapping is when we have a function of unitation. That is, the search space is binary strings of length ℓ and fitness only depends on the number of ones in a string. The phenotype set is

$$\Phi = \{0, 1, 2, \dots, \ell\}$$

2.2 The Eigen Model (Needle-in-a-haystack). A second example is that of the Eigen model [Eigen, 1971]. In this landscape all the strings have the same fitness except for a special string (the optimum) that has a relatively high fitness. - the so called “master sequence”. In this landscape, the genotype-phenotype coarse-graining creates only two equivalence classes, hence there is a reduction in degrees of freedom from N to one.

3 Exact and Approximate Invariance under a Coarse-Graining

Having motivated the idea of coarse graining and given some simple examples one needs to understand how the evolution equations for the GA look under the coarse graining. As mentioned, it was shown in [Stephens and Waelbroek, 1997] that the canonical GA is form invariant under a coarse graining to schemata, i.e. that the equations have exactly the same functional form after such a coarse graining. This is a highly non-trivial result, as a coarse graining in general will not preserve the functional form, as can be simply seen, for example, in the case of coarse graining from genotype to phenotype in the presence of mutation or crossover. Vose later showed [Vose, 1999] that schemata are the only coarse-grained variable that leave the dynamical equations for homologous crossover invariant and hence form a privileged set. However, Vose also introduced a more restrictive form of invariance under coarse graining - *compatibility* - wherein it was not sufficient that the equations be form invariant. Formally, if an operator $\mathcal{M} : \Lambda \rightarrow \Lambda$ gives the effect of applying an operator to a population, then a coarse-graining (Γ) is compatible with $\mathcal{M}$ if and only if, for any two populations $x, y \in \Lambda$

$$\sum_{j \in \gamma_i} x_j = \sum_{j \in \gamma_i} y_j \implies \sum_{j \in \gamma_i} \mathcal{M}(x)_j = \sum_{j \in \gamma_i} \mathcal{M}(y)_j$$

for all $\gamma_i \in \Gamma$ (see chapter 16-17 of [Vose, 1999]). It is known, for example, that schemata are compatible with crossover (by masks) and that unication classes are compatible with mutation.

A simple example illustrates the difference between the two different notions of invariance and coarse graining. Consider selection only in a two-bit one-max model: The equation of motion for proportional selection is

$$P(h_1 h_2, t+1) = (f(h_1 h_2) / \bar{f}(t)) P(h_1 h_2, t). \quad (1)$$

We pass to the schema h_1* by coarse graining h_2 to find

$$P(h_1*, t+1) = (f(h_1*, t) / \bar{f}(t)) P(h_1*, t) \quad (2)$$

where $f(h_1*, t) = (f(h_1 h_2) P(h_1 h_2, t) + f(h_1 \bar{h}_2) P(h_1 \bar{h}_2, t)) / (P(h_1 h_2, t) + P(h_1 \bar{h}_2, t))$ and $\bar{h}_2$ is the bit complement of h_2 . Clearly (1) and (2) have the same functional form. However, to satisfy compatibility $f(h_1*, t)$ would have to satisfy $f(h_1*, t) = f(h_1)$, where $f(h_1 = 1) = 1$ and $f(h_1 = 0) = 0$. Compatibility will only be valid when the problem exhibits an exact equivalence relation (“symmetry”) and the genetic operators respect this symmetry, such as is the case for the genotype-phenotype map and selection only, or with schemata and crossover only. The existence

of an exact symmetry usually allows for a reduction in the number of degrees of freedom by going to those effective degrees of freedom that are invariant under the symmetry.

However, the utility of coarse graining is not restricted to when it is compatible with the dynamics. For instance, in [Stephens, 2001] the form invariance of the equations of motion was used to prove a generalization of Geiringer's theorem to the case of non-flat landscapes. Further, in the physical sciences, where coarse graining has played an essential role, its utility is precisely for those cases where symmetries are not present, but rather where the coarse-grained dynamics can provide an approximate description of the system. In this case the closer are the coarse-grained variables to the true effective degrees of freedom then the better the approximation. For instance, in a strong selection regime one would expect phenotypes to approximate well the true dynamics with mutation and/or crossover inducing a small "interaction" between different phenotypes. Similarly, in the case of strong crossover and weak selection one would expect one-schemata to approximate well the dynamics with selection inducing a small interaction between the different one-schemata.

4 Crossover and Schemata Coarse Graining

We can track the evolutionary history of the production of a string via crossover, by looking at its constituent schemata, i.e. by using the Building Block basis. Suppose that we only have one-point crossover and no selection or mutation. We adopt the following notation, given that γ is a schema:

- $D(\gamma)$ is the set of indices for which γ has defined bit values. For example

$$D(1 * * 1 1 *) = \{1, 4, 5\}$$

- $L_j(\gamma)$ is the schema which has the same defining bits as γ for all indices $\leq j$, and stars elsewhere. For example, $L_4(1 * * 1 1 *) = 1 * * 1 * *$.
- $R_j(\gamma)$ is the schema which has the same defining bits as γ for all indices $> j$, and stars elsewhere. For example, $R_4(1 * * 1 1 *) = * * * * 1 *$.
- If the population at time t is $p \in \Lambda$, we write $P(\gamma, t) = \sum_{i \in \gamma} p_i$.

Then, following [Stephens and Waelbroek, 1997] we have

$$P(\gamma, t+1) = \sum_{j \in D(\gamma)} P(L_j(\gamma), t) P(R_j(\gamma), t)$$

Notice that this equation also applies to strings, by associating a string with the schema having all the corresponding bits defined.

We can use this equation to see how a given string can be created by crossover over several generations. For example, the string 1 1 1 can be created from the pairs 1 * *, * 1 1 and 1 1 *, * * 1. In the previous generation the schema * 1 1 may have been created from the pair * 1 *, * * 1. We see that there are several different possible "family trees" that can be constructed, the leaves of which are the order-one schemata which match the string at the root. All of the elements of the trees however are elements of the Building Block basis associated with the string of interest.

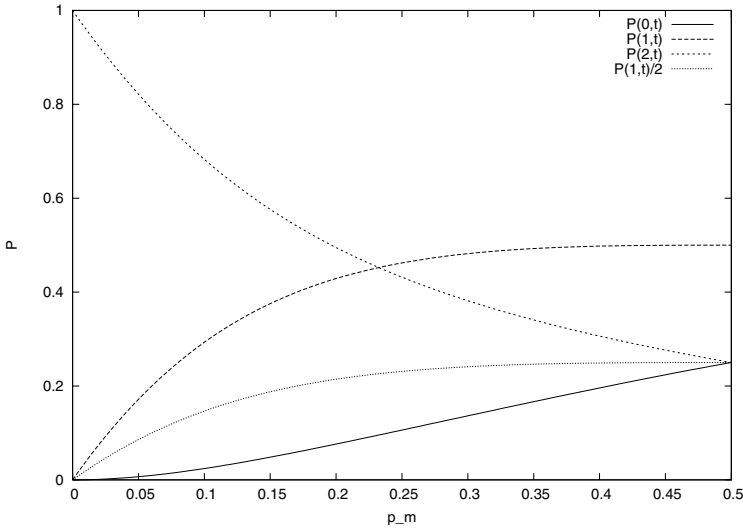


Fig. 1. Asymptotic string proportions in a onemax landscape with mutation rate p_m and no crossover. $P(2)$ decreases as we increase the probability of mutation, while $P(1)$ and $P(0)$ increase with p_m , and when p_m reaches $1/2$, all the strings are equally represented.

5 Mutation and the Genotype-Phenotype Coarse-Graining

The unitation coarse graining is a particular case of the genotype-phenotype coarse-graining in which the fitness of a string is its Hamming weight (or the Hamming distance to the 0 string, $f(i) = w(i) = d(i, 0)$). Now we can write the equivalence classes like $\gamma_j = \{i \in \Omega | w(i) = j\}$. In this particular case the reduction of the search space is huge, going from 2^ℓ degrees of freedom to only $\ell + 1$ effective degrees of freedom.

As an example of the coarse-graining technique in this scenario, consider a GA with probability of crossover zero, probability of mutation μ and $\ell = 2$, with the onemax fitness function. The set of equations describing the evolution of the system are as follows

$$\begin{pmatrix} p_{00}(t+1) \\ p_{01}(t+1) \\ p_{10}(t+1) \\ p_{11}(t+1) \end{pmatrix} = \begin{pmatrix} (1-\mu)^2 & (1-\mu)\mu & (1-\mu)\mu & \mu^2 \\ (1-\mu)\mu & (1-\mu)^2 & \mu^2 & (1-\mu)\mu \\ (1-\mu)\mu & \mu^2 & (1-\mu)^2 & (1-\mu)\mu \\ \mu^2 & (1-\mu)\mu & (1-\mu)\mu & (1-\mu)^2 \end{pmatrix} \begin{pmatrix} p_{00}(t) \\ p_{01}(t) \\ p_{10}(t) \\ p_{11}(t) \end{pmatrix} \quad (3)$$

$$\times \frac{1}{p_{01}(t) + p_{10}(t) + 2p_{11}(t)} \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 2 \end{pmatrix} \begin{pmatrix} p_{00}(t) \\ p_{01}(t) \\ p_{10}(t) \\ p_{11}(t) \end{pmatrix}$$

While in the coarse-grained basis the equations are:

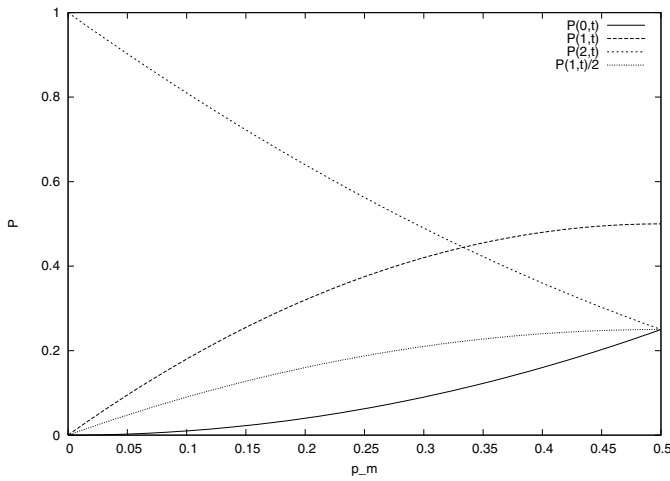


Fig. 2. Asymptotic string proportions in a Eigen model landscape with mutation rate p_m and no crossover.

$$\begin{pmatrix} p(\gamma_0, t+1) \\ p(\gamma_1, t+1) \\ p(\gamma_2, t+1) \end{pmatrix} = \begin{pmatrix} (1-\mu)^2 & (1-\mu)\mu & \mu^2 \\ 2(1-\mu)\mu & (1-\mu)^2 + \mu^2 & 2(1-\mu)\mu \\ \mu^2 & (1-\mu)\mu & (1-\mu)^2 \end{pmatrix} \\
 \times \frac{1}{p(\gamma_1, t) + 2p(\gamma_2, t)} \begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 2 \end{pmatrix} \begin{pmatrix} p(\gamma_0, t) \\ p(\gamma_1, t) \\ p(\gamma_2, t) \end{pmatrix} \quad (4)$$

We can solve this system using a similarity transformation for the state-transition matrix, finding the eigenvectors and replacing the original matrix with a diagonal similar matrix (see, for example, chapter 6 of [Reeves and Rowe, 2001]). The results are shown in figure 1, where we can see the fixed points as a function of the probability of mutation irrespective of the initial conditions of the population.

The unitation coarse-graining allow us to eliminate a redundant variable. Of course, it proves to be more useful as we increase the dimension of the search space [Rowe, 1998].

In contrast, for the Eigen model the genotype-phenotype coarse-graining is not compatible with mutation. Instead, we divide the space into Hamming distance classes from the master sequence. That is $\gamma_j = \{i \in \Omega | d(i, c_{ms}) = j\}$, where c_{ms} is the master sequence. In the case where $c_{ms} = 0$, this gives us the unitation coarse-graining.

Now we can solve the system using the same method as before, considering $f(c_{ms}) \gg f(j), \forall j \neq c_{ms}$. The results in the long-time limit for the two-bit problem are shown in figure 2. Note that qualitatively we obtain a very similar behaviour to the onemax landscape. This means that in spite of the fact that all the strings different to the master sequence have the same low fitness, evolution favours those close to the master sequence as we can see in figure 2. This phenomenon is known in the literature as the formation of a quasi-species.

6 Linkage-Disequilibrium

Let ij be any string of 2 bits, $i, j \in \{0, 1\}$, then the dynamics of the system under proportional selection (with the onemax landscape) and one-point crossover is given by the equations:

$$\begin{aligned} P(ij, t+1) &= (1-p_c)P'(ij, t) + p_c P'(ij, t) [P'(ij, t) + P'(i \not{j}, t) + P'(\not{i}j, t)] \\ &\quad + p_c P'(i \not{j}, t) P'(\not{i}j, t) \\ &= (1-p_c)P'(ij) + p_c P'(ij, t) [1 - P'(\not{i} \not{j}, t)] + p_c P'(i \not{j}, t) P'(\not{i}j, t) \\ &= P'(ij) + p_c [P'(i \not{j}, t) P'(\not{i}j, t) - P'(ij, t) P'(\not{i} \not{j}, t)] \end{aligned}$$

where P' is the proportion after selection, p_c is the probability of crossover and $\not{i}$ is the complement base 2 of i .

$\Delta'(t) = p_c [P'(i \not{j}, t) P'(\not{i}j, t) - P'(ij, t) P'(\not{i} \not{j}, t)]$ is the Selection Weighted Linkage Disequilibrium Coefficient (SWLDC) explicitly introduced in [Stephens, 2001] (and implicit in earlier work) in analogy with the original Linkage Disequilibrium Coefficient (LDC), well known in population biology, that measures how far is the current population from Robbins proportions (in which the bits are distributed independently — Geiringer's Theorem tells us that this is the limit of repeatedly applying crossover [Geiringer, 1944]). We can write

$$w(t+1) = \frac{p_c x(t)y(t)}{\bar{f}^2(t)} \quad (5)$$

$$x(t+1) = \frac{x(t)}{\bar{f}(t)} - \frac{p_c x(t)y(t)}{\bar{f}^2(t)} \quad (6)$$

$$y(t+1) = \frac{y(t)}{\bar{f}(t)} - \frac{p_c x(t)y(t)}{\bar{f}^2(t)} \quad (7)$$

$$z(t+1) = \frac{2z(t)}{\bar{f}(t)} + \frac{p_c x(t)y(t)}{\bar{f}^2(t)} \quad (8)$$

where $w(t) = P(00, t)$, $x(t) = P(01, t)$, $y(t) = P(10, t)$, $z(t) = P(11, t)$, notice that $P(00, t) = \Delta'(t) \forall t > 0$

Substituting the value of $x(t)$ from (6) and $y(t)$ from (7) in (5)

$$w(t+1) = \frac{p_c}{\bar{f}^2(t)} \left[\frac{x(t-1)}{\bar{f}(t-1)} - \frac{p_c x(t-1)y(t-1)}{\bar{f}^2(t-1)} \right] \left[\frac{y(t-1)}{\bar{f}(t-1)} - \frac{p_c x(t-1)y(t-1)}{\bar{f}^2(t-1)} \right] \quad (9)$$

which we can rewrite as:

$$w(t+1) = \frac{1}{\bar{f}^2(t)} \left[w(t) - p_c w(t) \left(\frac{x(t-1) + y(t-1)}{\bar{f}(t-1)} \right) + p_c w^2(t) \right] \quad (10)$$

As there are four genotypes and three phenotypes we now coarse-grain using the genotype-phenotype map, considering the phenotypic variable $b(t) = x(t) + y(t)$, adding (6) and (7) at time t to find

$$\frac{b(t-1)}{\bar{f}(t-1)} = b(t) + 2w(t) \quad (11)$$

which can then be substituted into (10) to obtain

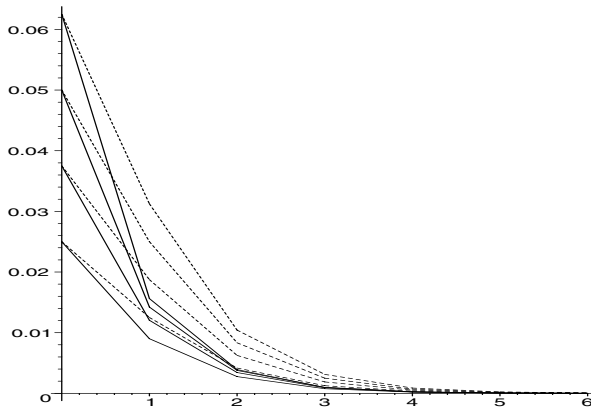


Fig. 3. Linkage Disequilibrium Coefficient $w(t)$ (continuous line) and the bound $w^*(t)$ (dotted line) in a typical run with a random initial population. The initial values correspond to different values of $p_c = \frac{j+2}{5}$, $j \in \{0, 1, 2, 3\}$

$$w(t+1) = \frac{w(t)}{\bar{f}^2(t)} [1 - p_c(w(t) + b(t))] \quad (12)$$

with $\bar{f}(t) = b(t) + 2z(t)$. Now if we assume $p_c = 1$ we can simplify (12) to

$$w(t+1) = \frac{w(t)}{2z(t) + b(t)} \left[\frac{z(t)}{2z(t) + b(t)} \right] \quad (13)$$

Note that this equation, because of the substitution (11), is only valid for $w(t) = 0$, as the latter is a fixed point of (13) but not of (5) (unless $x(t) = y(t) = 0$ as well), and also for $p_c > 0$.

Let $g(t) = \left[\frac{z(t)}{2z(t) + b(t)} \right]$, iterating we find the following solution

$$w(t) = \frac{w(0) \prod_{i=0}^{t-1} g(i)}{2^t z(0) + b(0) + \delta_{t \geq 3} w(0) \sum_{i=1}^{t-2} 2^i \prod_{j=0}^{t-2-i} g(j)} \quad (14)$$

where $\delta_C = 1$ if C is true and 0 otherwise. It is easy to see that $g(t) \leq \frac{1}{2} \forall t$, so we can write

$$\begin{aligned} w(t) &\leq \frac{\left(\frac{1}{2}\right)^t w(0)}{2^t z(0) + b(0) + \delta_{t \geq 3} w(0) \left(\sum_{i=1}^{t-2} 2^i \left(\frac{1}{2}\right)^{t-i-1} \right)} \\ &= \frac{\left(\frac{1}{2}\right)^t w(0)}{2^t z(0) + b(0) + \delta_{t \geq 3} \frac{w(0)}{3} (2^{t-1} - 2^{3-t})} \\ &\leq \frac{\left(\frac{1}{2}\right)^t w(0)}{2^t z(0) + b(0)} = w^*(t) \end{aligned} \quad (15)$$

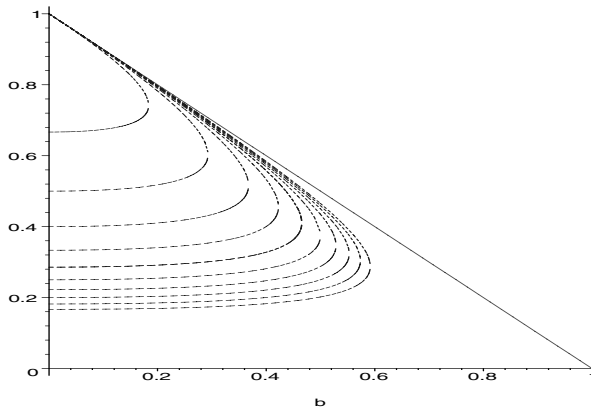


Fig. 4. The dotted lines represent zones with similar SLDC in the simplex as a function of $x(t) + y(t)$. The continuous line is the limit of the simplex.

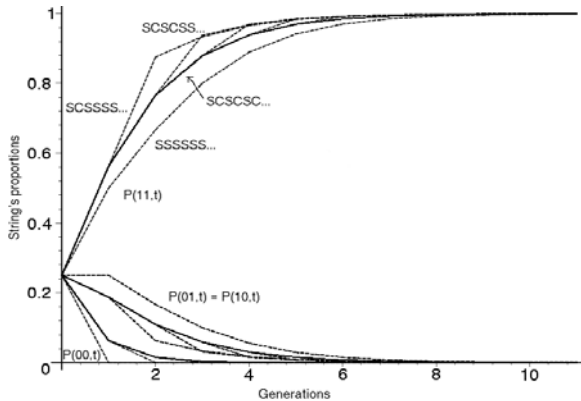


Fig. 5. The dotted lines represent the SCSSSS and the continuous SCSCSCSC

A comparison of the bound and the actual results are shown in fig. 3. Note that we would expect the bound to be better when $b(t) \ll z(t)$, however, for $3b(0) = 2z(0)$ so it is interesting to see that the bound gives reasonable results. Of course, strictly speaking the derived bound is for $p_c = 1$. The weaker the selective difference between 11 and 10 or 01 then the worse the bound. On the contrary, when the selective advantage of 11 over the other strings is large we expect the bound to become better and better. A further simplification lead us to

$$w^*(t) = \frac{\left(\frac{1}{2}\right)^t w(0)}{2^t z(0) + b(0)} \leq \left(\frac{1}{4}\right)^t \frac{w(0)}{z(0)} \quad (16)$$

where we can appreciate more clearly the exponential decay of the SWLDC.

By the definition of $w(t)$ in (5) and its value in (12) when $p_c = 1$ we can conclude that in every generation before selection we have $w(t)z(t) - x(t)y(t) = 0$ i.e. the population is in linkage (in the usual sense). This means that the population follows a path (in the simplex) always in linkage equilibrium, i.e. on the Geiringer manifold, towards a selective linkage equilibrium. Clearly, the latter is of more relevance than the former for the dynamical evolution.

In figure (4) we can see the different “contours” with the same SWLDC as a function of the coarse-grained variable $b(t)$. Notice $b(t) = P(1, t)$ plays an important role in the solution of the equations suggesting again a possible effective degree of freedom.

As the effects of crossover decay in time, as $b(t)$ gets smaller, we might think about approximating the usual iteration sequence SCSCSC... with the sequence SCSSSS.... Where S denotes selection and C crossover. The effects can be seen in Figure 5. Note how the approximation to SCSCSC... uniformly improves in a “perturbative” fashion as we include more Cs in the sequence, four Cs giving a very good approximation.

7 Conclusion

We have discussed the notion of coarse graining in GAs, giving a formal definition and some representative examples such as genotype-phenotype mappings, schemata, error classes. We discussed how the dynamical equations for a GA transform under a coarse graining comparing and contrasting the notions of form invariance and compatibility and discussed some practical issues and potential problems that arise when applying arbitrary coarse-grainings. However, the evolution equations of certain GAs can be simplified with an appropriate choice of subsets. Schemata are natural subsets to consider as the dynamics is form invariant under a schema coarse graining for selection, mutation and crossover, whereas only mutation and crossover are compatible with the dynamics. Unitation classes were also seen to be natural in the case of selection and bitwise mutation. Genotype-phenotype coarse-grainings are natural in the case of selection, since they give rise to constant (that is, static) fitness values for the subsets. We have illustrated these ideas and problems with some simple examples. Finally, we have shown how a genotype-phenotype coarse-graining can help calculate an estimate for the selective linkage disequilibrium coefficient in the two-bit case.

References

- [Eigen, 1971] Eigen, M. (1971). Self-organization of matter and the evolution of biological macromolecules. *Naturwissenschaften*, 58:465.
- [Geiringer, 1944] Geiringer, H. (1944). On the probability theory of linkage in Mendelian heredity. *Annals of Mathematical Statistics*, 15(1):25–57.
- [Holland, 1975] Holland, J. (1975). *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, Michigan.
- [Radcliffe, 1992] Radcliffe, N. J. (1992). The algebra of genetic algorithms. *Annals of Mathematics and Artificial Intelligence*, 10:339–384.
- [Reeves and Rowe, 2001] Reeves, C. R. and Rowe, J. E. (2001). *Genetic Algorithms — Principles and Perspectives*, chapter 6, The Dynamical Systems Model. Kluwer Academic Publishers.

- [Rowe, 1998] Rowe, J. (1998). Population fixed-points for functions of unitation. In Banzhaf, W. and Reeves, C., editors, *Foundations of Genetic Algorithms*, pages 69–84. Morgan Kaufmann.
- [Stadler and Stephens, 2003] Stadler, P. F. and Stephens, C. R. (2003). Landscapes and effective fitness. *Comm. Theor. Biol.* Accepted for publication.
- [Stephens, 2001] Stephens, C. R. (2001). Some exact results from a coarse grained formulation of genetic dynamics. In Spector, L., Goodman, E. D., Wu, A., Langdon, W. B., Voigt, H.-M., Gen, M., Sen, S., Dorigo, M., Pezeshk, S., Garzon, M. H., and Burke, E., editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, pages 631–638, San Francisco, California, USA. Morgan Kaufmann.
- [Stephens, 2003] Stephens, C. R. (2003). The renormalization group and the dynamics of genetic systems. *Acta Phys. Slov.*, 52:515–524.
- [Stephens and Waelbroek, 1997] Stephens, C. R. and Waelbroek, H. (1997). Effective degrees of freedom in genetic algorithms and the block hypothesis. In Bäck, T., editor, *Proceedings of the Seventh International Conference on Genetic Algorithms*, pages 34–41. Morgan Kauffman.
- [van Nimwegen et al., 1997] van Nimwegen, E., Crutchfield, J. P., and Mitchell, M. (1997). Finite populations induce metastability in evolutionary search. *Physics Letters A*, 229:144–150.
- [Vose, 1991] Vose, M. D. (1991). Generalizing the notion of a schema in genetic algorithms. *Artificial Intelligence*, 50:385–396.
- [Vose, 1999] Vose, M. D. (1999). *The Simple Genetic Algorithm: Foundations and Theory*. MIT Press, Cambridge, MA.
- [Vose and Wright, 2001] Vose, M. D. and Wright, A. H. (2001). Form invariance and implicit parallelism. *Evolutionary Computation*, 9(3):355–370.

Building a GA from Design Principles for Learning Bayesian Networks

Steven van Dijk, Dirk Thierens, and Linda C. van der Gaag

Universiteit Utrecht, Institute of Information and Computing Sciences, Decision Support Systems, PO Box 80.089, 3508 TB Utrecht, The Netherlands
{`steven, dirk, linda`}@cs.uu.nl

Abstract. Recent developments in GA theory have given rise to a number of design principles that serve to guide the construction of selecto-recombinative GAs from which good performance can be expected. In this paper, we demonstrate their application to the design of a GA for a well-known hard problem in machine learning: the construction of a Bayesian network from data. We show that the resulting GA is able to efficiently and reliably find good solutions. Comparisons against state-of-the-art learning algorithms, moreover, are favorable.

1 Introduction

Recent developments in GA theory [5, 18, 12, 6, 20] have yielded in-depth insight in the search behavior of selecto-recombinative GAs. This insight stresses the importance of concepts such as linkage, mixing, and disruption. The use of a selecto-recombinative GA for solving a search problem is appropriate if the linkage of the problem can be assessed with reasonable confidence. The linkage then determines the location of the building blocks for good solutions for the problem. It further allows for the design of a crossover operator that mixes well and indicates where disruption can occur, thereby making counter measures easier to take. We use these recent insights to define various design principles to guide the construction of GAs from which we can expect good performance.

To demonstrate the effectiveness of our GA design principles, we apply them to the problem of learning Bayesian networks (BNs) from data. BNs [13] are probabilistic graphical models that capture a joint probability distribution by explicitly modeling the independences between the statistical variables involved in a directed acyclic graph (DAG). The strengths of the relationships between the variables are quantified by probability tables. Figure 1 depicts a hypothetical example of a BN. While Bayesian networks have proven their value as a robust framework for reasoning with uncertainty in a range of applications, their construction can be a daunting task. Especially when having to rely upon extensive collaboration of domain experts to build the graphical structure and to elicit the required probabilities, handcrafting a network is hard and very time consuming. If databases are available that store information about the statistical variables of importance, however, these data can be exploited to construct a Bayesian

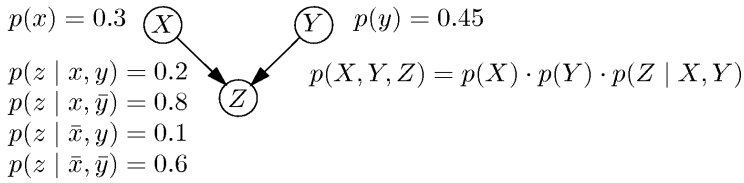


Fig. 1. Bayesian network with three variables ($X = \{x, \bar{x}\}$, $Y = \{y, \bar{y}\}$ and $Z = \{z, \bar{z}\}$).

network automatically. The learned network can then be further improved with the help of domain experts.

The BN learning problem essentially is an optimization problem, where a Bayesian network has to be found that best represents the probability distribution that has generated the data in a given database. The use of maximum-likelihood estimates allows us to take the frequencies in the data for a network's probability tables. The learning problem thereby reduces to the problem of searching the optimum in the space of all DAGs. A trade-off has to be made between the structural complexity of a network and the accuracy with which it describes the data, since complex networks tend to suffer from over-fitting and make the running time of inference algorithms prohibitively large. An oft-used measure that serves to balance accuracy and complexity is the MDL measure, based on the principle of minimal description length from information theory [15,9]. In this paper we solve the BN learning problem for databases with complete cases by searching for a DAG that, after completion to a network with maximum-likelihood estimates, minimizes the MDL score.

The learning problem is a suitable problem to exemplify our GA design principles with since the graphical nature of a BN allows for an easy assessment of the structure of the problem. A learned network should connect two nodes if the corresponding variables are dependent. Even though the reverse need not be true, strong dependences found in the data indicate which pairs of nodes are important for the search. This observation translates into an assessment of the linkage between the genes in an appropriately-chosen representation, and therefore of the location of the building blocks.

Our contributions are the following. We show how to solve an important search problem from GA design principles. We further describe a carefully designed GA for the learning problem that compares favorably against current state-of-the-art learning algorithms. We present results on two real networks in addition to the commonly used Alarm network. The remainder of the paper is organized as follows. In Section 2, we discuss previous work on the BN learning problem. We continue in Section 3 with a discussion of recent developments in GA theory and with a statement of our design principles. In Section 4, we describe our GA for the learning problem. We present results and comparisons in Section 5. We conclude and discuss future work in Section 6.

2 Related Work

Most state-of-the-art algorithms for learning BNs from complete data can be classified as taking one of two approaches: the use of an (in)dependence test such as Pearson χ^2 or mutual information [16,4], and the use of a quality measure such as MDL [2,7,9]. With both approaches, encouraging results have been reported. Both approaches, however, also have their disadvantages.

In the first approach, a statistical test is employed for examining whether or not two variables are (in)dependent given some conditioning set of variables. The *order* of the test is the size of the conditioning set used. By starting with zero-order tests and selectively growing the conditioning set, in theory, all relevant (in)dependences can be extracted from the data and the network can be exactly recovered. In practice, however, the test quickly becomes unreliable for higher orders. For these higher orders, the number of data available for the test decreases exponentially with the order. For example, for binary variables and a database of 1000 cases (a realistic size), a sixth-order independence test would be based, on average, on approximately $1000/2^6 \approx 15$ cases from the database. As a result, the test can be inaccurate, which can affect the quality of the learned network.

In the second approach, an information measure is used for assessing the quality of candidate DAGs. This approach suffers from the size of the search space of DAGs. To efficiently traverse this huge space, often a greedy search algorithm is used to focus attention on the most promising regions. Other algorithms explicitly constrain the search space by assuming a topological ordering on the nodes of candidate DAGs. With both types of algorithm, the optimal DAG may be pruned from the space that is effectively searched.

Larrañaga et al. [11] have proposed a genetic algorithm based upon the latter approach. In their GA, a DAG is represented by a connectivity matrix that is stored as a string (the concatenation of its rows). Recombination is implemented as one-point crossover on these strings; mutation is implemented as random bit-flipping. After children have been generated, the resulting graphs are rendered acyclic by randomly deleting arcs that are part of a cycle. In addition, for nodes with a number of (immediate) predecessors that exceeds a predefined constant, the best subset is chosen that is small enough. In related work, Larrañaga et al. [10] experimented with a GA that searches for an ordering that is passed on to K2, a greedy search algorithm. They concluded that the results were comparable to those of their previous GA. The use of different crossover operators in a GA for the learning problem has recently been explored by Cotta and Muruzábal [3], who concluded that adaptive operators gave the best results.

Recently, Wong et al. [22] proposed the “hybrid evolutionary programming” (HEP) algorithm that combines the use of independence tests with a quality-based search. The search space of DAGs is constrained in the sense that each possible DAG only connects two nodes if they show a strong dependence in the available data. The algorithm evolves a population of DAGs to find a solution that minimizes the MDL score. A new population $Pop(t+1)$ is constructed as follows. First, half the members from $Pop(t)$ are copied and put in an intermediate population. The members from the other half are “merged” with

the members from the previous population $Pop(t-1)$ that were *not* selected. Merging is similar to crossover, except that it selectively uses those parts of the parents that give the largest improvement of the MDL score of the child. The child is then put in the intermediate population. To each individual in this intermediate population, four different mutation operators are applied. The resulting population is then added to $Pop(t)$. Selection consists of ranking the individuals and putting the top half in $Pop(t+1)$.

As mentioned before, the HEP algorithm constrains the search space of DAGs by only allowing edges that connect nodes with a strong dependence between them. More specifically, the algorithm allows an edge between two nodes if there exists a zero-order dependence between them, and no first-order order independences. The (in)dependence test employed makes use of a threshold that indicates the required level of significance. Wong et al. argue that it is better to avoid manually setting the threshold. Instead, they allow the search algorithm to evolve the threshold and constrain the solutions to match their individual threshold. Initially, a threshold is chosen at random for each individual in the population. A new individual receives the mutated threshold of its parent.

Wong et al. compared an earlier version of their algorithm against the GA of Larrañaga et al. [11], and found that it was faster and produced networks of better MDL scores. We will compare our GA against HEP in Section 5.

3 Design Considerations

During the previous decade, GA theory has progressed to the point where GAs giving predictable results can be designed. A descriptive model of the selecto-recombinative GA has emerged from the literature [5, 18, 12, 6, 20] in which the representation of solutions is divided into several partitions¹. A schema of highest fitness in a partition is called a building block. The proportion of building blocks of a certain partition in the population should converge to 1.0. The propagation of building blocks to this end is modeled by competitions between strings and statistical decision making. By mixing on the boundaries of the partitions, the building blocks are assembled into a close-to-optimal solution. In other words, a good solution matches the building block of each partition. The model is applicable especially to problems that involve an additively decomposable fitness function. In the work by Harik et al. [6], more specifically, partitions are separable, that is, any gene is part of a single partition. The partitions themselves then correspond to the subfunctions into which the fitness function can be decomposed. The model was tested on deceptive subfunctions, and proved to be quite accurate. Later it was shown that the model could also predict results for more realistic problems that involve overlapping partitions, such as the map-labeling

¹ A *chromosome* is a string over an alphabet A . A *schema* is a string over $A \cup \{\#\}$, which is matched by a chromosome that has the same symbols in the same positions, except for the $\#$'s. A *partition* is a string over $\{\#, f\}$ that is matched by schemata that have an element from A at positions where the partition has an f .

problem [20], provided that the deviations from the underlying assumptions are not too severe.

The insights yielded by the model of the selecto-recombinative GA can be formulated into the following design principles [21]:

- Use an additively decomposable fitness function.
- Ensure a good building-block supply, either in the initial population or during the run.
- Ensure that the proportions of building blocks increase.
- Ensure good mixing of building blocks.
- Minimize disruption of building blocks.

The first principle allows for a representation of solutions with a direct mapping of the fitness subfunctions to genes. Ensuring a good building-block supply is vital since it is impossible to construct a good solution without building blocks. Building blocks will be scattered throughout the initial population, or will be injected during the run by the genetic operators. Proper selection of building blocks is necessary to ensure their proportions within the population grow. During the run of the algorithm, all building blocks have to be assembled on the same individual, which makes good mixing of building blocks vital. Since mixing tends to be disruptive, however, care needs to be taken that not too many building blocks are lost. Central to most of these principles is the assessment of the linkage, which is the first design issue to be addressed. We next exemplify the various principles by designing a GA for the BN learning problem.

4 The GA

We recall that the learning problem involves searching for a DAG. For the representation of DAGs, we use a list of genes (corresponding with connections between nodes) of fixed length. Each gene can be set to three different alleles corresponding with the two possible arc directions and the absence of an arc. Within this representation, we need to group genes into partitions in such a way that the building blocks of the partitions match a close-to-optimal solution. Recall that each node corresponds with a statistical variable. The arcs to and from that node correspond with dependences that are reflected in the data. Therefore, genes are linked (part of the same partition) if they involve the same node. The linkage of an efficient GA involves partitions whose size is bounded by a small constant. Therefore, we use only connections that correspond with plausible dependences. This effectively constrains the search to plausible DAGs.

The whole algorithm now consists of two phases. First, we construct from the database an undirected graph, called the *skeleton graph*, that represents the search space for the GA. The graph contains a node for each statistical variable present in the database. The edges of the skeleton graph are found by performing conditional dependence tests on the data: nodes are connected by edges if their corresponding variables are found to be dependent. In the next phase, the GA will turn the skeleton into a DAG by evolving a population of DAGs that use

the skeleton as a template. We observe that during the run of the GA, the graphical structures yielded need to be acyclic. Our GA ensures this property by applying a “repair” operator as soon as cycles are introduced during initialization and crossover. This repair operator breaks cycles in a manner that is not too disruptive. In addition, no node can have more than a predefined number of direct predecessors. This constraint is enforced because Bayesian networks with a large number of predecessors are impractical, since performing probabilistic inference on such networks is computationally prohibitive.

4.1 Generation of the Skeleton Graph

We construct from the data a skeleton graph to reduce the search space for the GA. The skeleton graph is constructed from the “important” edges for each node. An edge is deemed important if zero- and first-order tests show that the corresponding variables are dependent. We use just zero- and first-order conditional dependence tests, since higher-order tests quickly become unreliable. If we now consider the Bayesian network with the optimal (lowest) MDL score given the data, then, ideally, the skeleton graph is the underlying undirected graph of the DAG of the network. The skeleton graph, however, can include spurious edges when a higher-order independence test would have been required to remove a connection between two nodes. In our experiments, we found that skeleton graphs were produced that could contain approximately twice as many edges as the original network from which the data was sampled.

To construct the skeleton graph, we start with the empty graph and find for each node the nodes that should be connected to it. An edge is added between two nodes if their corresponding variables are dependent. The neighbors of a node X are found by building a candidate list of neighbors. This list contains all nodes Y that show a dependence on X through a zero-order dependence test. It is possible that two nodes (variables) are dependent without having to be connected, such as the nodes X and Y in Figure 2. These nodes become independent given Z . We therefore try to remove each node Y from the candidate list of neighbors for X by testing for a dependence on X given any other node Z in the list. If such a test shows that X and Y become independent given Z , then Y is removed from X ’s candidate list. For each node Y in the final list of neighbors of X we then add an edge between X and Y in the skeleton graph. The total number of dependence tests to be performed when constructing the skeleton graph is bounded by $O(n^2)$, where n denotes the number of statistical variables (and therefore nodes). In our experiments, up to 45 percent of the total running time was spent on calculating the skeleton graph.

We used the χ^2 -test for determining whether or not two statistical variables are dependent given some conditioning set of variables. This test uses a threshold indicating the largest p-value that would cause it to reject the null-hypothesis that the variables are independent. A large threshold allows weak dependences to be found, but may also result in reporting incorrect dependences. A small threshold makes the test more selective. In practice, we found that the zero-order dependences from the original network could be recovered with a threshold

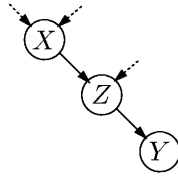


Fig. 2. Variables X and Y are dependent, yet independent given Z .

very close to zero. In fact, the p-values of the conditional tests rise significantly for nodes that are not true neighbors. In our experiments, therefore, we set the threshold as low as 0.005.

4.2 The GA

In the design of our GA for the BN learning problem, we applied the design principles mentioned before. The requirement of using an additively decomposable fitness function is satisfied by using the MDL measure. This measure is derived from information theory and states that the *description length* of a BN and a database is the sum of the size of the network, and the size of the database after it has been compressed using the network. The best network given a database then is the one yielding the smallest description length. We note that a network that is more complex can represent the underlying distribution of the data better and compress it to a smaller size. Complex networks, however, require a larger encoding to specify their arcs and tables. The MDL measure balances these two issues. It is to be minimized and has the following decomposed form [9]:

$$MDL(D, B) = \sum_{i=1}^n MDL_i(D, \pi(i)),$$

where D denotes the database, B is the candidate Bayesian network, n is the number of nodes or variables, and $\pi(i)$ equals the set of predecessors of node i . The measure thus decomposes into local terms per node.

Having chosen an additively decomposable fitness function, we now outline the GA, indicating how the other design principles are satisfied:

Encoding: a string of genes, each of which corresponds with an edge in the skeleton graph, and can be set in one of the following states (the alleles): DIRECTEDFROM, DIRECTEDTO, NONE.

Initialization: an assignment of one of the three alleles is made to each gene at random. In other words, an edge is either removed or turned into an arc. All incoming arcs of nodes that thus end up with too many predecessors (recall that no node can have more than a given number of predecessors) are traversed in random order. An arc is reversed and if it then still causes an infeasible solution, it is removed. Finally, the solution is made acyclic with the operator MAKEACYCLIC.

Selection scheme: for selection and replacement, the incremental elitist recombination scheme [17] is used.

Mutation: no mutation in the traditional sense is used. Traditionally, random mutation is used to (re)generate building blocks during the run. In our GA, we ensure a good building-block supply by choosing an appropriately sized population and as a side-effect of the genetic operators (recombination, REPAIR and MAKEACYCLIC).

Recombination: to guarantee good mixing of building blocks, recombination is performed by crossover on the level of partitions. Through the crossover operator, we try to transfer schemata from the relevant partitions intact from parent to child. Since partitions overlap, this may still cause possible building blocks to be disrupted. Disruption is dealt with after crossover. For the crossover operator, a node is picked at random. The genes corresponding with the edges to its neighbors in the skeleton graph are added to the selection. The operator continues picking nodes until half the number of genes are selected. The first child now inherits the selection of genes from the first parent and its complement from the second one. The other child is constructed by taking the selection of genes from the second parent and its complement from the first one. The genetic operator REPAIR is applied to the edges in both children that are part of a possibly disrupted building block. Finally, MAKEACYCLIC is applied to both children.

Fitness function: the MDL measure is used to compute the fitness of solutions. It is minimized to find the best solution.

Stop criterion: the GA is halted when the difference between the average fitness in the population and the fitness of the best individual is smaller than a certain threshold (we used 0.001), or when the last 10% of the recombinations performed did not improve the average score by 1.

We describe the REPAIR and MAKEACYCLIC operators in more detail.

The REPAIR operator is used to handle the disruption of possible building blocks after crossover. We consider a node in one of the parents before recombination. The genes that correspond with the edges in the skeleton graph connected to that node can store a building block, that is, a combination of alleles for those genes that matches a close-to-optimal solution. If the node is picked by the crossover operator, these genes are transferred together to a child and no disruption occurs. If, on the other hand, the child inherits alleles for these genes from both parents, a building block may be disrupted. We can repair the building block by returning the genes inherited from one of the parents to their former setting. Since we don't know which settings are building blocks, we rely on the MDL score to find the best change. For our implementation, we make the simplifying assumption, for reasons of efficiency, that we can retrieve the original building block by changing genes one at a time. Moreover, to avoid changing a child too much, we try to find a minimal set of edges to be changed. The results obtained with the simplifying assumption are quite satisfactory.

The REPAIR operator now proceeds as follows. For every node, it investigates the genes that correspond with the edges connected to that node in the

skeleton graph. Suppose that the first parent donates x alleles and the second parent donates y alleles to a child. If $x < y$, the operator puts the x (indices to the) edges in a set S , otherwise the y edges. It then continues with subsequent nodes, disregarding edges already in S . For example, if the second node that is investigated has seven incident edges, of which two are already in S and three are inherited from the first parent, the remaining two edges are put in S . After all nodes have been visited, each edge in S is investigated for each child. If the allele for the gene corresponding with the edge can be changed to another allele with an improvement of the MDL score, the best change is chosen. In a typical run of the GA, for a skeleton of 95 edges, on average eight edges are changed in a child in the early iterations of the run. This number gradually drops to zero as the population converges.

The function MAKEACYCLIC operator is used to render a graph acyclic. It first finds all cycles in the graph by performing a recursive descent from each node of a spanning tree. Next, it finds the change for any edge that is part of a cycle that deteriorates the MDL score the least. Note that any change will break at least one cycle. This is repeated until all cycles have been resolved. If a change caused a new cycle or causes an arc to point to a node with the maximum number of predecessors, the operator reverts the change and marks it as forbidden. The process is then repeated (but forbidden states are avoided) until all cycles have been broken. Note that termination is guaranteed since a change to a deleted edge will break a cycle but not cause new conflicts.

It is worthwhile to point out that the running time of the GA can be dramatically optimized by caching the local score for a node with a certain predecessor set. Since the calculation of the MDL score is computationally intensive and has to be done by any MDL-based learning algorithm, the use of a population-based search method gives a relatively minor overhead. For example, a typical run with a population size of 150 for a database of 10000 cases took 14 minutes; a run with a population size of 1000—a more than six-fold increase—took only 35. In our experiments, the GA (using an incremental selection scheme) performed on average 2385 recombinations until convergence, corresponding with about 32 generations in a generational scheme.

5 Results

To study the performance of the resulting GA, we conducted experiments on various databases that were sampled from three real-life Bayesian networks by means of logic sampling [8]. The Alarm network [1] (37 nodes, 46 arcs) was originally built to help anesthetists monitor their patients, and is widely used for evaluating the performance of BN learning algorithms. The Oesoca network [19] (42 nodes, 59 arcs) was developed at Utrecht University, in collaboration with The Netherlands Cancer Institute, to aid gastroenterologists in staging oesophageal cancer and predicting the outcome of different treatment alternatives. The VSD network [14] (38 nodes, 52 arcs) also comes from the medical domain and was developed for treatment planning for a congenital heart disease.

We compared our GA against a rather straightforward hillclimber (HC), for a baseline performance. The hillclimber starts with the empty graph and considers pairs of nodes that are connected in the skeleton graph. In each step, it finds the change (remove, insert, or flip an arc) that improves the MDL score the most. This is continued until no progress is made, upon which MAKEACYCLIC is called to produce the result. We also compared our GA against the original implementation of the HEP algorithm by Wong et al. In our experiments, we used databases of 1000 and 10000 cases. The GA and the HEP algorithm were run ten times. All algorithms allowed a maximum of five predecessors per node. The GA used a population size of 150. The HEP algorithm used the default settings as suggested by Wong et al. The results are presented in Table 1.

Table 1. Results of the experiments. Below the name of the database, the MDL score of the original network is shown. In each table, the first column denotes the algorithm, the second gives the average MDL score with standard deviation, and the third column shows the best MDL score found in the ten runs.

	algo	avg±sd	best		algo	avg±sd	best
Alarm-1000 (17834.3)	GA	16757.4±12.7	16747.1	Alarm-10000 (139266.9)	GA	139240.4±89.7	139097.5
	HEP	16591.9±40.4	16567.2		HEP	140196.6±590.6	139441.7
	HC	17387.3			HC	140946.4	
Oesoca-1000 (26070.0)	GA	24065.0±0.0	24065.0	Oesoca-10000 (213765.7)	GA	213089.8±0.0	213089.8
	HEP	23800.3±72.8	23748.5		HEP	213478.8±1096.1	212715.1
	HC	24676.6			HC	215192.8	
VSD-1000 (25790.6)	GA	23186.3±0.0	23186.3	VSD-10000 (207647.4)	GA	206296.1±15.56	206251.2
	HEP	24641.7±52.8	24551.5		HEP	220160.1±958.3	219423.3
	HC	23318.8			HC	209101.0	

From the table we observe that both the GA and HEP yield networks with MDL scores that are close to the score of the original network. In fact, for all networks and associated databases, the GA found an average score that was better than the score of the original network. For all but two databases (Alarm-10000 and VSD-10000), the same observation holds for HEP. It may seem surprising that the original network is not the network of the best MDL score. The difference originates from the fact that the database is a finite sample, subject to sampling error, that does not reflect all the dependences from the original network accurately. The probability distribution observed in the data, therefore, is likely to differ from the distribution captured by the original network.

A striking difference in the results yielded by the two algorithms is that the GA shows a much smaller standard deviation than HEP. The GA can thus be seen as the more reliable of the two algorithms as it is more likely to always give results of similar quality. We observe, however, that for half the number of databases (Alarm-1000, Oesoca-1000 and Oesoca-10000) HEP's best result is slightly better than the GA's. We further observe that the hillclimber also

performs quite well, thereby giving an indication of the benefit from the skeleton graph. The average MDL score yielded by the GA is always better than that by the HC, whereas HEP's best solution can be worse than the result from the HC.

The GA and the HEP algorithm yielded their results fairly quickly, within a maximum of 20 minutes per run. For the databases of 1000 cases, the longest running time was four minutes. HEP was generally faster than the GA, with a maximal difference in running time by a factor eight. It is important to realize, however, that for the BN learning problem, the time spent is not a critical factor. Once a network is learned, it is either used directly for daily problem solving, or inspected and improved manually. The time spent on learning the network, therefore, is well within reasonable bounds for both algorithms.

6 Conclusions

Building upon recent developments in GA theory we discussed various principles for GA design. We demonstrated how to apply these design principles, rather straightforwardly, to derive a GA for learning Bayesian networks from data that performs at least comparably to the state-of-the-art algorithms. We feel that the robustness of the resulting GA is a confirmation of the feasibility of our approach. For future work, we would like to put the model from which our design principles were derived on a more formal footing. Specifically, it is worth investigating how overlap between partitions influences the predictions of the model.

Acknowledgments. This research was supported by the Netherlands Organisation for Scientific Research (NWO). We would like to thank Wong, Lee and Leung for graciously providing us with an implementation of their HEP algorithm.

References

1. I. A. Beinlich, H. J. Suermondt, R. M. Chavez, and G. F. Cooper. The Alarm monitoring system: a case study with two probabilistic inference techniques for belief networks. In J. Hunter et al., editors, *Proceedings of the Second Conference on Artificial Intelligence in Medicine*, pages 247–256. Springer, 1989.
2. G. F. Cooper and E. Herskovits. A Bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9:309–347, 1992.
3. C. Cotta and J. Muruzábal. Towards more efficient evolutionary induction of Bayesian networks. In J.-J. M. Guervós et al., editors, *Lecture Notes in Computer Science, Volume 2439: Proceedings of the Parallel Problem Solving from Nature VII Conference*, pages 730–739. Springer-Verlag, 2002.
4. L. M. de Campos and J. F. Huete. A new approach for learning belief networks using independence criteria. *International Journal of Approximate Reasoning*, 24(1):11–37, 2000.
5. D. E. Goldberg, K. Deb, and J. H. Clark. Genetic algorithms, noise, and the sizing of populations. *Complex Systems*, 6(4):333–362, 1992.

6. G. Harik, E. Cantú-Paz, D. E. Goldberg, and B. L. Miller. The gambler's ruin problem, genetic algorithms, and the sizing of populations. *Evolutionary Computation*, 7(3):231–253, 1999.
7. D. Heckerman, D. Geiger, and D. M. Chickering. Learning Bayesian networks: The combination of knowledge and statistical data. *Machine Learning*, 20(3):197–243, 1995.
8. M. Henrion. Propagating uncertainty in Bayesian networks by probabilistic logic sampling. In J. F. Lemmer and L. N. Kanal, editors, *Proceedings of the Second Conference on Uncertainty in Artificial Intelligence*, pages 149–163. Elsevier, 1988.
9. W. Lam and F. Bacchus. Using causal information and local measures to learn Bayesian networks. In D. Heckerman and A. Mamdani, editors, *Proceedings of the Ninth Conference on Uncertainty in Artificial Intelligence*, pages 243–250. Morgan-Kaufmann, 1993.
10. P. Larrañaga, C. M. H. Kuijpers, R. H. Murga, and Y. Yurramendi. Learning Bayesian network structures by searching for best ordering with genetic algorithm. *IEEE Transactions on Systems, Man, and Cybernetics*, 26(4):487–493, 1996.
11. P. Larrañaga, M. Poza, Y. Yurramendi, R. Murga, and C. Kuijpers. Structure learning of Bayesian networks by genetic algorithms: A performance analysis of control parameters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(9):912–926, 1996.
12. B. L. Miller and D. E. Goldberg. Genetic algorithms, selection schemes, and the varying effects of noise. *Evolutionary Computation*, 4(2):113–131, 1996.
13. J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 1988.
14. N. Peek and J. Ottenkamp. Developing a decision-theoretic network for a congenital heart disease. In E. Keravnou et al., editors, *Proceedings of the Sixth European Conference on Artificial Intelligence in Medicine*, pages 157–168. Springer, 1997.
15. J. J. Rissanen. Modelling by shortest data description. *Automatica*, 14:465–471, 1978.
16. P. Spirtes and C. Glymour. An algorithm for fast recovery of sparse causal graphs. *Social Science Computer Review*, 9(1):62–73, 1991.
17. D. Thierens. Selection schemes, elitist recombination, and selection intensity. In T. Bäck, editor, *Proceedings of the Seventh International Conference on Genetic Algorithms and their Applications*, pages 152–159. Morgan-Kaufmann, 1997.
18. D. Thierens and D. E. Goldberg. Mixing in genetic algorithms. In S. Forrest, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms and their Applications*, pages 38–45. Morgan-Kaufmann, 1993.
19. L. C. van der Gaag, S. Renooij, C. Witteman, B. Aleman, and B. Taal. Probabilities for a probabilistic network: A case-study in oesophageal cancer. *Artificial Intelligence in Medicine*, 25(2):123–148, 2002.
20. S. van Dijk, D. Thierens, and M. de Berg. Scalability and efficiency of genetic algorithms for geometrical applications. In M. Schoenauer et al., editors, *Lecture Notes in Computer Science, Volume 1917: Proceedings of the Parallel Problem Solving from Nature VI Conference*, pages 683–692. Springer-Verlag, 2000.
21. S. van Dijk, D. Thierens, and M. de Berg. On the design and analysis of competent GAs. Technical Report TR-2002-15, Utrecht University, 2002.
22. M. L. Wong, S. Y. Lee, and K. S. Leung. A hybrid data mining approach to discover Bayesian networks using evolutionary programming. In W. B. Langdon et al., editors, *Proceedings of the Genetic and Evolutionary Computation Conference*. Morgan-Kaufmann, 2002.

A Method for Handling Numerical Attributes in GA-Based Inductive Concept Learners

Federico Divina, Maarten Keijzer, and Elena Marchiori

Department of Computer Science
Vrije Universiteit
De Boelelaan 1081a, 1081 HV Amsterdam
The Netherlands
{divina,mkeijzer,elena}@cs.vu.nl

Abstract. This paper proposes a method for dealing with numerical attributes in inductive concept learning systems based on genetic algorithms. The method uses constraints for restricting the range of values of the attributes and novel stochastic operators for modifying the constraints. These operators exploit information on the distribution of the values of an attribute. The method is embedded into a GA based system for inductive logic programming. Results of experiments on various data sets indicate that the method provides an effective local discretization tool for GA based inductive concept learners.

1 Introduction

Inductive Concept Learning (ICL) [14] constitutes a central topic in Machine Learning. The problem can be formulated in the following manner: given a description language used to express possible hypotheses, a background knowledge, a set of positive examples, and a set of negative examples, one has to find a hypothesis which covers all positive examples and none of the negative ones (cf. [12, 15]). The so learned concept can be used to classify previously unseen examples. Concepts are induced because obtained from the observation of a limited set of training examples. When hypotheses are expressed in (a fragment of) first order logic, ICL is called Inductive Logic Programming (ILP).

Many learning problems use data containing numerical attributes. Numerical attributes affect the efficiency of learning and the accuracy of the learned theory. The standard approach for dealing with numerical attributes in inductive concept learning is to discretize them into intervals that will be used instead of the continuous values. The discretization can be done during the learning process (*local* discretization), or beforehand (*global* discretization). Discretization methods that employ the class information of the instances are called *supervised* methods, while if they do not use this information they are called *unsupervised* methods. The simplest way is to use an equal interval width method. In this way, the continuous values are simply divided into n equal sized bins, where n is a parameter. A better way for discretizing numerical attributes was proposed by Fayyad and Irani [9]. This method uses a recursive entropy minimization

algorithm and employs the Minimum Description Length principle in the stopping criterion. In [16] a variant of the Fayyad and Irani's method is used for discretizing numerical attributes in an Inductive Logic Programming system. In [1,2] methods using adaptive discrete intervals are used within a GA based system for classification. Another approach for local discretization is proposed by Kwedlo and Kretowski. In their work, information on a subset of all thresholds of a numerical attribute is used as to determine thresholds in evolved decision rules [13].

The aim of this paper is to introduce an alternative method for dealing with numerical attributes in evolving classifiers where the actual discretization is determined at run time (i.e. local discretization). An unsupervised, global method is used to determine the density of data. Due to the use of unsupervised methods, unlabeled data or a priori knowledge about the distribution of data can be used to fine-tune the density estimation. The information gathered in the global pre-processing step is used to guide the genetic operators to make density controlled operations in the search process.

The method introduced here is general, in that guiding the genetic operators using estimated, unlabeled or a priori information about the distribution of data can be used in any evolutionary classifying system. In order to assess the benefits of the method we use one particular system: the evolutionary ILP system ECL [7,6]. We run experiments on different data sets and compare the results obtained using the original ECL system (where numerical attributes are treated as nominal), ECL with numerical attributes discretized using Fayyad and Irani's method, and ECL with numerical attributes treated using the novel method based on constraints. The results of the experiments indicate that the proposed method allows ECL to find better solutions which are comparable or better than those found using Fayyad and Irani's method.

The paper is structured in the following way. In section 2 we describe the method for dealing with numerical attributes. In section 3 we introduce the main features of the ECL system. In section 4 we perform experiments and discuss the results and finally in section 5 some conclusions and future work are given.

2 Handling Numerical Attributes Using Constraints

We propose to handle numerical attributes by using constraints of the form $a \leq X \leq b$, where X is a variable relative to a numerical attribute, and a, b are attribute values. During the execution of a GA based inductive concept learner, a constraint for a numerical attribute is generated when that attribute is selected. Constraints are then modified during the evolutionary process by using the novel operators defined in the following sections. These novel operators use information on the distribution of the values of attributes in order to update the interval boundaries of the constraints. Information about the distribution of the values of attributes is obtained by clustering the values of each attribute using a mixture of Gaussian distribution.

We will first briefly introduce the clustering algorithm that is used, and then we will describe how constraints are modified. In the sequel we will use Prolog syntax, where a variable starts with uppercase character while a constant starts with a lowercase character.

2.1 Clustering Attribute Values

We cluster the values of each attribute in order to get information about the distribution of the data, and use this information in the operators for modifying constraints. Clustering is performed using the Expectation-Maximization (EM) algorithm [5] (in the experiments we use WEKA implementation [17]). For each attribute the EM algorithm returns n clusters described by means μ_i and standard deviations σ_i , $1 \leq i \leq n$ of Gaussian distributions.

A begin (b_{cl_i}) and end (e_{cl_i}) of a cluster $cluster_i$ are generated by intersecting the distributions of $cluster_i$ with the one of $cluster_{i-1}$ and $cluster_{i+1}$, respectively. Special cases are $b_{cl_1} = -\infty$ and $e_{cl_n} = +\infty$.

The boundaries a, b of each constraint $a \leq X \leq b$ are contained in one cluster.

2.2 Operators

Within each cluster, we use constraints for restricting the range of values of an attribute variable.

A constraint can be modified either by enlarging its boundaries, or by shrinking, or by shifting the boundaries, or by changing the cluster of its boundaries, or by grounding the constraint (i.e., restricting its range to a single value).

Formally, consider the constraint $C : a \leq X \leq b$, and let b_{cl} and e_{cl} be the begin and the end of the cluster cl containing C .

Enlarge. This operator applied to C returns a constraint $C' = a' \leq X \leq b'$ where $a' \leq a$ and $b \leq b'$. The new bounds a', b' are computed in the following way:

1. let $min = \text{minimum} \{P(b_{cl} \leq X \leq a), P(b \leq X \leq e_{cl})\}$ the minimum of the probability that X is between b_{cl} and a and the probability that X is between b and e_{cl} .
2. generate randomly p with $0 \leq p \leq min$;
3. find two points a', b' such that $p = P(a' \leq X \leq a)$ and $p = P(b \leq X \leq b')$.

Bounds are enlarged by generating probabilities instead of random points inside the cluster because in this way we can exploit the information about the distribution of the data values in an interval.

Shrink. This operator applied to C returns $C' = a' \leq X \leq b'$ where $a' \geq a$ and $b' \leq b$. a' and b' are computed by randomly choosing $p \leq P(a, b)$ such that $p = P(a \leq X \leq a') = P(b' \leq X \leq b)$, and $a' \leq b'$.

Ground. This operator, applied to C returns $C' = a' \leq X \leq a'$, with a' in the cluster containing a, b .

Shift. This operator, applied to C returns $C' = a' \leq X \leq b'$ where a', b' are points in the cluster containing a, b such that $P(a' \leq X \leq b') = P(a \leq X \leq b)$.

Change Cluster. This operator, applied to $C = a \leq X \leq b$ returns $C' = a' \leq X \leq b'$ where a', b' belong to a different cluster. The new cluster is chosen at random. Once the new cluster has been chosen, a pair a', b' with $a' \leq b'$ is randomly generated. In general, $P(a' \leq X \leq b')$ is not equal to $P(a \leq X \leq b)$.

3 ECL: A GA Based Inductive Concept Learner

In order to test the effectiveness of our discretization method, we embed it in the ILP system ECL. Like many ILP systems, ECL treats numerical attributes as if they were nominal, therefore it can be used as a platform for testing and comparing our local discretization method with the global discretization method by Fayyad and Irani.

In figure 1 a scheme of ECL is given.

```

ALGORITHM ECL
Sel = positive_examples
repeat
    Select partial Background Knowledge
    Population = {}
    while (not terminate) do
        Adjust examples weights
        Select n chromosomes using Sel
        for each selected chromosome chrm
            Mutate chrm
            Optimize chrm
            Insert chrm in Population
        end for
    end while
    Store Population in Final_Population
    Sel = Sel - { positive examples
                  covered by clauses in Population }
until max_iter is reached
Extract final theory from Final_Population

```

Fig. 1. The overall learning algorithm ECL

The system takes as input a background knowledge (BK), and a set of positive and negative examples, and outputs a set of Horn clauses that covers many positive examples and few negative ones.

Recall that a Horn clause is of the form $p(X, Y) : \neg r(X, Z), q(Y, a)$. with head $p(X, Y)$ and body $r(X, Z), q(Y, a)$. A clause has a declarative interpretation: $\forall X, Y, Z (r(X, Z), q(X, a) \rightarrow p(X, Y))$ and a procedural one: *in order to solve $p(X, Y)$ solve $r(X, Z)$ and $q(Y, a)$* . Thus a set of clauses forms a logic program, which can directly (in a slightly different syntax) be executed in the programming language Prolog. The background knowledge used by ECL contains ground facts (i.e. clauses of the form $r(a, b) \leftarrow .$ with a, b constants). The training set contains facts which are true (positive examples) and false (negative examples) for the target predicate. A clause is said to *cover an example* if the theory formed by the clause and the background knowledge logically entails the example.

In the repeat statement of the algorithm a **Final_population** is iteratively built from the empty one. Each iteration performs the following actions: part of the background knowledge is randomly selected, an evolutionary algorithm that uses that part of BK is run and the resulting set of Horn clauses is joined to the actual **Final_population**.

The evolutionary algorithm evolves a **Population** of Horn clauses starting from an empty population, where an individual represents a clause, by the repeated application of selection, mutation (the system does not use any crossover operator) and optimization in the following way.

At each generation n individuals are selected using a variant of the US selection operator [10]. Roughly, the selection operator selects a positive example and performs a roulette wheel on the set of individuals in the **Population** that cover that example. If that example is not covered by any individual then a new clause is created using that example as seed.

Each selected individual undergoes mutation and optimization.

Mutation consists of the application of one of the following four generalization/specialization operators. A clause is generalized either by deleting an atom from its body or by turning a constant into a variable, and it is specialized by either adding an atom or turning a variable into a constant. Each operator has a degree of greediness. In order to make a mutation, a number of mutation possibilities is considered, and the one yielding the best improvement, in terms of fitness, is applied.

Optimization consists of the repeated application of one of the mutation operators, until the fitness of the individual increases, or a maximum number of optimization steps has been reached.

Individuals are then inserted in the population. If the population is not full then the individuals are simply inserted. If the population has reached its maximum size, then n tournaments are made among the individuals in the population and the resulting n worst individuals are substituted by the new individuals.

The fitness of an individual x is given by the inverse of its accuracy:

$$fitness(x) = \frac{1}{Acc(x)} = \frac{P+N}{p_x+(N-n_x)}$$

In the above formula P and N are respectively the total number of positive and negative examples, while p_x and n_x are the number of positive and negative examples covered by the individual x . We take the inverse of the accuracy, because ECL was originally designed to minimize a fitness function.

When the **Final_population** is large enough (after **max_iter** iterations) a subset of its clauses is extracted in such a way that it covers as many positive examples as possible, and as few negative ones as possible. To this aim an heuristic algorithm for the weighted set covering is used.

3.1 Clu-Con: ECL Plus Local Discretization

We consider the following variant of ECL, called Clu-Con (Clustering and Constrain), which incorporates our method for handling numerical values.

When a new clause is built using a positive example as a seed, or when a clause is specialized, atoms of the background knowledge are added to its body. Each time an atom describing the value of a numerical attribute is introduced in a clause, a constraint relative to that attribute is added to the clause as well. For example, consider the following clause for example $c23$:

$$Cl = p(c23) : -q(c23, a), t(c23, y).$$

Suppose now that we would like to add the atom $r(c23, 8)$ stating that in example $c23$ attribute r has value 8. Then we obtain the clause

$$p(c23) : -q(c23, a), t(c23, y), r(c23, X), 8 \leq X \leq 8.$$

The operators for handling constraints, introduced in section 2.2, are used as mutation operators. When an operator is chosen then it is applied to a constraint a number $n\_choices$ of times, where $n\_choices$ is a user supplied parameter. In this way $n\_choices$ new constraints are generated and the one yielding the best fitness improvement is selected.

Shrink and Ground. These two operators are applied when specializing a clause. More precisely, when the system is specializing a clause by turning a variable into a constant, if the selected variable occurs in a constraint then either Shrink or Ground are applied to that constraint.

Enlarge. This operator is applied when the system decides to generalize a clause. ECL has two generalization operators: *delete an atom* and *constant into variable* operators. When *delete an atom* is selected and the atom chosen for deletion describes the value of a numerical attribute, then both the atom and the constraint relative to the described attribute are deleted. If *delete an atom* is not selected and there are constraints in the body of the clause chosen for mutation, then the system randomly selects between the *constant into variable* and *enlarge* operators.

Change Cluster and Shift. The standard operators and the above described operators are applied inside a *mutate* procedure. Before calling this procedure, a test is performed to check if the selected individual has got constraints in the body of its clause. If this is the case, then either the *change cluster* or the *shift* operator is applied with a probability pc (typical value 0.2), otherwise the *mutate* procedure is called.

3.2 Ent_MDL: ECL Plus Global Discretization

The other variant of ECL we consider, called Ent_MDL (Entropy minimization plus Minimum Description Length principle), incorporates the popular Fayyad and Irani's method for discretizing numerical values [9]. In [8,11] a study of some discretization methods is conducted, and it emerged that Fayyad and Irani's method represents a good way for globally discretizing numerical attributes.

This supervised recursive algorithm uses the class information entropy of candidate intervals to select the boundaries of the bins for discretization. Given a set S of instances, an attribute p , and a partition bound t , the class information entropy of the partition induced by t is given by

$$E(p, t, S) = Entropy(S_1) \frac{|S_1|}{|S|} + Entropy(S_2) \frac{|S_2|}{|S|}$$

where S_1 is the set of instances whose values of p are in the first half of the partition and S_2 the set of instances whose values of p are in the second half of the partition. Moreover, $|S|$ denotes the number of elements of S and $Entropy(S) = -p_+ \log_2(p_+) - p_- \log_2(p_-)$ with p_+ the proportion of positive examples in S and p_- is the proportion of negative examples in S .

For a given attribute p the boundary t^* which minimizes $E(p, t, S)$ is selected as a binary discretization boundary. The method is then applied recursively to both the partitions induced by t^* until a stopping criterion is satisfied. The Minimum Description Length principle is used to define the stopping criterion. Recursive partitions within a set of instances stops if $Entropy(S) - E(p, t, S)$ is smaller than $\log_2(N - 1)/N + \Delta(p, t, S)/N$, where $\Delta(p, t, S) = \log_2(3^k - 2) - [k \cdot Entropy(S) - k_1 \cdot Entropy(S_1) - k_2 \cdot Entropy(S_2)]$, and k_i is the number of class labels represented in S_i .

The method is used as a pre-processing step for ECL, where the domains of numerical attributes are split into a number of intervals, and each interval is considered as one value of a nominal attribute.

4 Experiments

In order to asses the goodness of the proposed method for handling numerical attributes, we conduct experiments on benchmark datasets using ECL and the two variants Clu.Con and Ent_MDL described previously. The characteristics of the datasets are shown in table 1. Datasets 1 to 6 are taken from the UCI repository [3], while dataset 7 originates from [4]. These datasets are chosen

Table 1. Features of the datasets. The first column shows the total number of training examples with, between brackets, the number of positive and negative examples. The second and third columns show the number of numerical and nominal attributes. The fourth column shows the number of elements of the background knowledge.

	Dataset	Instances (+,-)	Numerical	Nominal	BK
1	Australian	690 (307,383)	6	8	9660
2	German	1000 (700,300)	24	0	24000
3	Glass2	163 (87,76)	9	0	1467
4	Heart	270 (120,150)	13	0	3510
5	Ionosphere	351 (225,126)	34	0	11934
6	Pima-Indians	768 (500,268)	8	0	6144
7	Mutagenesis	188 (125,63)	6	4	13125

Table 2. Parameter settings: pop_size = maximum size of population, mut_rate = mutation rate, n = number of selected clauses, max_gen = maximum number of GA generations, max_iter = maximum number of iterations, N(1,2,3,4) = parameters of the genetic operators, p= probability of selecting a fact in the background knowledge, l = maximum length of a clause.

	Australian	German	Glass2	Heart	Ionosphere	Pima-Indians	Mutagenesis
pop_size	50	200	150	50	50	60	50
mut_rate	1	1	1	1	1	1	1
n	15	30	20	15	15	7	15
max_gen	1	2	3	1	6	5	2
max_iter	10	30	15	10	10	10	10
N(1,2,3,4)	(4,4,4,4)	(3,3,3,3)	(2,8,2,9)	(4,4,4,4)	(4,8,4,8)	(2,5,3,5)	(4,8,2,8)
nb	∞	200	8	20	20	8	8
p	0.4	0.3	0.8	1	0.2	0.2	0.8
l	6	9	5	6	5	4	3

because they contain mainly numerical attributes. The parameters used in the experiments are shown in table 2 and are obtained after a number of preliminary experiments. We use ten-fold cross validation. Each dataset is divided in ten disjoint sets of similar size; one of these sets is used as test set, and the union of the remaining nine forms the training set. Then ECL is run on the training set and it outputs a logic program, whose performance on new examples is assessed using the test set. Three runs with different random seed are performed on each dataset.

Table 3 contains the results of the experiments. The performance of the GA learner improves when numerical attributes are discretized. In particular, Clu-Con seems to perform best on these datasets, being outperformed by Ent-MDL only in two cases, namely on the Ionosphere and the Pima-Indians datasets.

Table 3. Results of experiments. Average accuracy, with standard deviation between brackets. In the first column the method adopting clusters and constraints described in the paper is employed. In the second column numerical attributes are treated as nominal. In the third column the Fayyad and Irani’s discretization algorithm was used for globally discretizing numerical attributes.

	Dataset	Clu-Con	ECL	Ent-MDL
1	Australian	0.79 (0.03)	0.71 (0.05)	0.62 (0.06)
2	German	0.73 (0.02)	0.63 (0.07)	0.59 (0.08)
3	Glass2	0.78 (0.07)	0.51 (0.06)	0.69 (0.11)
4	Heart	0.74 (0.09)	0.67 (0.11)	0.68 (0.10)
5	Ionosphere	0.81 (0.07)	0.41 (0.03)	0.85 (0.06)
6	Pima-Indians	0.65 (0.09)	0.61 (0.05)	0.69 (0.07)
7	Mutagenesis	0.90 (0.04)	0.83 (0.05)	0.85 (0.07)

5 Conclusions and Future Work

In this paper we have proposed an alternative method for dealing with numerical attributes. The method uses standard density estimation to obtain a view of the distribution of data, and this information is used to guide the genetic operators to find local discretizations that are optimal for classification. By using an unsupervised method for density estimation, the method can make use of either unlabeled data or a priori knowledge on the density of the numeric attribute. In section 4 we have tested the method in the system ECL on some datasets containing numerical attributes. In a previous version of the system, no particular method for dealing with numerical attributes was implemented, so numerical attributes were treated as if they were nominal. We have shown that not only the proposed method improves the performance of the system, but also that the proposed method is in general more effective than a global discretization by means of Fayyad and Irani’s algorithm. We believe that the proposed method could be profitably applied to other learning systems for dealing with numerical attributes.

Currently, the density estimation procedure only works with single attributes. However, when there is reason to believe that numeric attributes are covariant (for instance weight and height of a person), the entire method can be converted to use multiple attributes. For this, a multivariate variant of the expectation maximization algorithm can be used which will find the covariance matrix Σ instead of a single standard deviation σ . Because our genetic operators are defined to use only density information derived from the clusters, the operators can be easily adapted to mutate two or more numerical attributes at the same time in the context of the globally estimated covariance matrices. This is however left for future work.

We are currently investigating the possibility of changing the operators that act on constraints. In particular we would like to generate the probabilities p_i in the enlarge and shrink operator not at random, but in a way that could take

into consideration the actual coverage of the constraint. We are also considering the possibility of enlarging or shrinking a constraint asymmetrically.

References

1. J. BACARDIT AND J. M. GARREL, *Evolution of adaptive discretization intervals for a rule-based genetic learning system*, in GECCO 2002: Proceedings of the Genetic and Evolutionary Computation Conference, New York, 9–13 July 2002, Morgan Kaufmann Publishers, p. 677.
2. —, *Evolution of multi-adaptive discretization intervals for a rule-based genetic learning system*, in Proceedings of the 7th Iberoamerican Conference on Artificial Intelligence (IBERAMIA2002), 2002, p. to appear.
3. C. BLAKE AND C. MERZ, *UCI repository of machine learning databases*, 1998.
4. A. DEBNATH, R. L. DE COMPADRE, G. DEBNATH, A. SCHUSTERMAN, AND C. HANSCH, *Structure-Activity Relationship of Mutagenic Aromatic and Heteroaromatic Nitro Compounds. Correlation with molecular orbital energies and hydrophobicity*, Journal of Medical Chemistry, 34(2) (1991), pp. 786–797.
5. A. P. DEMPSTER, N. M. LAIRD, AND D. B. RUBIN, *Maximum likelihood from incomplete data via the EM algorithm*, Journal of the the Royal Statistical Society, 39 (1977), pp. 1–38.
6. F. DIVINA, M. KEIJZER, AND E. MARCHIORI, *Non-universal suffrage selection operators favor population diversity in genetic algorithms*, in Benelearn 2002: Proceedings of the 12th Belgian-Dutch Conference on Machine Learning (Technical report UU-CS-2002-046), Utrecht, The Netherlands, 4 December 2002, pp. 23–30.
7. F. DIVINA AND E. MARCHIORI, *Evolutionary concept learning*, in GECCO 2002: Proceedings of the Genetic and Evolutionary Computation Conference, New York, 9–13 July 2002, Morgan Kaufmann Publishers, pp. 343–350.
8. J. DOUGHERTY, R. KOHAVI, AND M. SAHAMI, *Supervised and unsupervised discretization of continuous features*, in International Conference on Machine Learning, 1995, pp. 194–202.
9. U. FAYYAD AND K. IRANI, *Multi-interval discretization of continuous attributes as pre-processing for classification learning*, in Proceedings of the 13th International Joint Conference on Artificial Intelligence, Morgan Kaufmann Publishers, 1993, pp. 1022–1027.
10. A. GIORDANA AND F. NERI, *Search-intensive concept induction*, Evolutionary Computation, 3 (1996), pp. 375–416.
11. R. KOHAVI AND M. SAHAMI, *Error-based and entropy-based discretization of continuous features*, in Proceedings of the Second International Conference on Knowledge Discovery and Data Mining, 1996, pp. 114–119.
12. M. KUBAT, I. BRATKO, AND R. MICHALSKI, *A review of Machine Learning Methods*, in Machine Learning and Data Mining, R. Michalski, I. Bratko, and M. Kubat, eds., John Wiley and Sons Ltd., Chichester, 1998.
13. W. KWEDLO AND M. KRETOWSKI, *An evolutionary algorithm using multivariate discretization for decision rule induction*, in Principles of Data Mining and Knowledge Discovery, 1999, pp. 392–397.
14. T. MITCHELL, *Generalization as search*, Artificial Intelligence, 18 (1982), pp. 203–226.
15. —, *Machine Learning*, Series in Computer Science, McGraw-Hill, 1997.

16. W. VAN LAER, *From Propositional to First Order Logic in Machine Learning and Data Mining – Induction of first order rules with ICL*, PhD thesis, Department of Computer Science, K.U.Leuven, Leuven, Belgium, June 2002. 239+xviii pages.
17. I. H. WITTEN AND E. FRANK, *Weka machine learning algorithms in java*, in Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations, Morgan Kaufmann Publishers, 2000, pp. 265–320.

Analysis of the (1+1) EA for a Dynamically Bitwise Changing ONEMAX

Stefan Droste*

LS Informatik 2, Universität Dortmund, 44221 Dortmund, Germany
`stefan.droste@udo.edu`

Abstract. Although evolutionary algorithms (EAs) are often successfully used for the optimization of dynamically changing objective functions, there are only very few theoretical results for EAs in this scenario. In this paper we analyze the (1+1) EA for a dynamically changing ONEMAX, whose target bit string changes bitwise, i. e. possibly by more than one bit in a step. We compute the movement rate of the target bit string resulting in a polynomial expected first hitting time of the (1+1) EA asymptotically exactly. This strengthens a previous result, where the dynamically changing ONEMAX changed only at most one bit at a time.

1 Introduction

Evolutionary Algorithms (EAs) optimize objective functions heuristically by following principles of natural evolution (see [BFM97] for an in-depth coverage). A great variety of EAs is successfully used in practice and the experimental knowledge about EAs is immense. In comparison, theory of EAs is still in its beginnings. Many results are based on empirical observations, precluding generalizations. Other results, like the basic forms of the schema theorem ([Hol75]) are limited to the short-time behaviour of EAs. Such results cannot be carried over to other dimensions of the search space, even if the objective function is fixed. This can only be achieved by a rigorous mathematical analysis showing how the first hitting time of the EA (or any other measure of interest) depends on the dimension of the search space.

As there are no theoretical foundations, the EAs and objective functions rigorously analyzed so far are simple in comparison to those used in practice. One of the most thoroughly examined EAs is the (1+1) EA which was analyzed by many different researchers (e. g., see [Müh92], [Rud97], or [DJW02]). The techniques developed while analyzing this simple-structured EA have led to analyses of more complex EAs using crossover and populations (see [JW99] or [JW01]), making it a good starting point for the analysis of dynamic EAs.

So far theoretical analyses of EAs almost completely focus on static optimization problems, i. e. objective functions that do not change over time. Although

* This research was partly supported by the Deutsche Forschungsgemeinschaft as part of the Collaborative Research Center “Computational Intelligence” (531).

different kinds of EAs are successfully used for a broad range of dynamical optimization problems (see [Bra01] for an overview), there are only very few theoretical results. Most investigations are limited to experimental work or heuristic approximations. A first more theoretical approach is [SD99], where the transition probabilities of a (1+1) EA for a dynamic variant of the ONEMAX-problem are numerically evaluated and compared to empirical data, but not analyzed mathematically.

In this paper, we strengthen the first rigorous mathematical analysis of an EA for a dynamic optimization problem presented in [Dro02]. There the standard (1+1) EA with a mutation probability of $1/n$ is analyzed on a dynamic variant of ONEMAX, where in every step with probability p' exactly one uniformly chosen bit of the target bit string changes. Hence, in contrast to the static case, the Hamming distance of the current search point to the target bit string can increase by one. It is shown in [Dro02] that the expected first hitting time of the (1+1) EA is polynomial if and only if p' is asymptotically of growth rate at most $\log(n)/n$, i. e. $O(\log(n)/n)$ (see [MR95] for the notation of growth rates).

Here, we look at the more interesting case of changing the target bit string by the same type of random process as the current search point of the (1+1) EA, i. e. each bit is flipped with probability p' . We prove that the expected first hitting time of the (1+1) EA is with high probability super-polynomial if p' grows faster than $\log(n)/n^2$, i. e. $p' = \omega(\log(n)/n^2)$. In the light of [Dro02] the result may not be surprising, but its analysis is significantly more complicated, because we have to consider a process where the Hamming distance between the current search point and the target bit string can increase by more than one in one step. By showing how such a process can be upper bounded by a random process where the distance increases by at most one in one step we prove the desired result. This bounding technique (although it should not be new, we have not found it in the literature) may be applicable in other settings and therefore be of more general interest.

In the next section we define the (1+1) EA for dynamic objective functions, the dynamic variant of ONEMAX, and the first hitting time, our measure of efficiency. In Sect. 3 we show that for $p' = \omega(\log(n)/n^2)$ the first hitting time of the (1+1) EA is polynomial only with super-polynomially small probability, implying a super-polynomial expected first hitting time. In Sect. 4 we show how stochastic processes that can move away from the target in one step by more than one can be related to processes where this difference can only increase by one in one step. Using this result we show that the expected first hitting time for $p' = O(\log(n)/n^2)$ is polynomial in n . The paper ends with some conclusions.

2 The (1+1) EA and a Dynamic ONEMAX

The (1+1) EA is probably the most simple EA for maximizing an objective function $f: \{0, 1\}^n \rightarrow \mathbb{R}$ as it uses only one individual being effected by mutation only. The commonly used mutation operator for bit strings is the bitwise mutation, i. e. each bit is flipped independently with probability $p \in [0, 1]$. In most

cases p is $1/n$. Selection chooses between the old bit string and the mutant, where the mutant is only chosen if its f -value is at least as high as that of the old bit string. This mutation-selection scheme is repeated until a stopping criterion is fulfilled. Since we are interested in the number of steps until an optimum of f is evaluated for the first time, we omit this stopping criterion.

If f is not static but changes over time, we model it by a family $(f_t: \{0, 1\}^n \rightarrow \mathbb{R})_{t \in \mathbb{N}_0}$ of functions, where f_t is the current objective function after t steps of the (1+1) EA. In this situation the (1+1) EA should evaluate even the old bit string after each step, as its fitness might have changed since the last evaluation. So, the (1+1) EA for dynamic optimization problems is defined as follows:

Definition 1 ((1+1) EA for dynamic optimization problems).

1. Set $t := 0$ and choose $x_t \in \{0, 1\}^n$ randomly uniformly.
2. Set $x' := x_t$ and flip each bit of x' with probability $1/n$.
3. If $f_t(x') \geq f_t(x_t)$, set $x_{t+1} := x'$, else $x_{t+1} := x_t$.
4. Set $t := t + 1$ and go to step 2.

There are a number of different measures for the performance of an algorithm for dynamic optimization, e. g. Hamming distance to the nearest optimum. Here we concentrate on the number T_f of steps until the (1+1) EA for the first time evaluates a point x' where this distance is zero:

Definition 2 (First hitting time of the (1+1) EA).

Let $f = (f_t: \{0, 1\}^n \rightarrow \mathbb{R})_{t \in \mathbb{N}_0}$ be a dynamically changing objective function. The first hitting time T_f of the (1+1) EA on f is defined as

$$T_f := \min\{t \in \mathbb{N}_0 \mid f_t(x_t) = \max\{f_t(x) \mid x \in \{0, 1\}^n\}\}.$$

One of the first functions the (1+1) EA has been analyzed for in the static case is ONEMAX. Here the expected first hitting time is $O(n \log(n))$ (see [Müh92] for an approximate analysis and [DJW98] for a rigorous one). ONEMAX simply counts the number of ones in its argument $x \in \{0, 1\}^n$. This can also be interpreted as the number of *matching* bits with the *target bit string* $(1, \dots, 1)$. Hence, maximizing ONEMAX is equivalent to minimizing the Hamming distance to this target bit string. Using this interpretation, a dynamic ONEMAX-variant where the target bit string is changed by a random process is obvious. Let $M: \{0, 1\}^n \rightarrow \{0, 1\}^n$ represent this random process (formally it is a function $M: \{0, 1\}^n \times \Omega \rightarrow \{0, 1\}^n$, where Ω together with a function $P: \mathcal{P}(\Omega) \rightarrow [0, 1]$ forms a probability space, but we omit these technicalities) and $H(x, y)$ denote the Hamming distance between bit strings x and y :

Definition 3 (ONEMAX_{t,M}).

The family $(\text{ONEMAX}_{t,M}: \{0, 1\}^n \rightarrow \mathbb{N}_0)_{t \in \mathbb{N}_0}$ of functions is defined as

$$\text{ONEMAX}_{t,M}(x) := n - H(x, y_t),$$

where $y_0 := (1, \dots, 1)$ and $y_{t+1} := M(y_t)$ for $t \geq 0$.

(As the initial individual x_0 is chosen uniformly at random, initializing y_0 with any other bit string makes no difference.) In [Dro02] the random operator M_1 flips exactly one uniformly chosen bit with probability p' and it is shown that the expected first hitting time of the (1+1) EA is polynomial if and only if $p' = O(\log(n)/n)$. Here we look at a more natural random operator M_n by allowing M_n to flip each bit independently with probability p' :

Definition 4 (Bitwise operator M_n).

Let $p' \in [0, 1]$. Then the random operator $M_n: \{0, 1\}^n \rightarrow \{0, 1\}^n$ is defined for all $y, y' \in \{0, 1\}^n$ by:

$$P(M_n(y) = y') := (p')^i \cdot (1 - p')^{n-i} \text{ if } H(y, y') = i.$$

The parameter p' is called the *movement rate* of the target bit string as it influences the number of steps in which the target moves and also the number of bits that flip. In the following we will denote $(\text{ONEMAX}_{t, M_n})_{t \in \mathbb{N}_0}$ shortly by ONEMAX_D to indicate the dynamic variant of ONEMAX . To separate more easily between the changes made by the mutation of the (1+1) EA and those of the random operator M_n , we call bits flipped by mutation *mutating bits* and bits changed by M_n *moving bits*. In the next section we show that for $p' = \omega(\log(n)/n^2)$ the (1+1) EA has a super-polynomial expected first hitting time.

3 A Super-Polynomial Lower Bound for Large Movement Rates

In this section we assume that p' grows asymptotically faster than $\log(n)/n^2$, i. e. $p' = \omega(\log(n)/n^2)$. We show that the probability of the (1+1) EA reaching the optimum of ONEMAX_D in polynomially many steps is super-polynomially small, i. e. at most $1/\beta(n)$, where $\beta(n)$ grows faster than any polynomial in n .

First, we consider $p' = \Omega(1/n^{1-\varepsilon})$ for a constant $\varepsilon > 0$. Then the probability of any specific movement of the target bit string is super-polynomially small, implying that in the last step before reaching the optimum of ONEMAX_D a super-polynomially unlikely movement of the target bit string has to happen (here we assume that $p' \leq 1/2$ as otherwise more than half of the bits of the target would move on average in each step, making efficient optimization impossible):

Theorem 1. Let $p' = \Omega(1/n^{1-\varepsilon})$ for a constant $\varepsilon \in]0, 1[$. Then the first hitting time of the (1+1) EA on ONEMAX_D is $\exp(o(n^\varepsilon))$ only with exponentially small probability.

Proof. We look at an arbitrary run of the (1+1) EA that reaches the optimum after exactly $T > 0$ steps. Then it is necessary that in the last step the movement of the target bit string from y_{T-1} to y_T leads to $y_T = x'$. For $H(y_{T-1}, x') = i$, the probability of such a movement equals $(p')^i \cdot (1 - p')^{n-i}$.

If i grows linearly in n , the factor $(p')^i$ is $\exp(-O(n))$. If $n - i$ grows linearly in n , the second factor is asymptotically at most (where $n - i = \beta(n) = \Theta(n)$)

$$\left(1 - \frac{1}{n^{1-\varepsilon}}\right)^{\beta(n)} = \left(1 - \frac{n^\varepsilon}{n}\right)^{(n/n^\varepsilon) \cdot (n^\varepsilon/n) \cdot \beta(n)},$$

which is upper bounded by $\exp(-c \cdot n^\varepsilon)$ for a constant $c > 0$. Since i or $n - i$ grows linearly and the probability of initializing with $x_0 \neq y_0$ is exponentially close to 1, we know that in order to reach the optimum an event with probability at most $\exp(-c \cdot n^\varepsilon)$ has to happen. $\square$

We have to argue more carefully if p' is not that large but still $\omega(\log(n)/n^2)$. Although the proof follows the lines of the super-polynomial lower bound in [Dro02], it must be adjusted to take into account that the target bit string can move by more than one bit at a time step. First, we show the following lemma:

Lemma 1. *Let a Bernoulli experiment with success probability $p \in]0, 1]$ be repeated independently $a + b$ -times ($a, b \in \mathbb{N}$). The probability that in the first a experiments the number of successes is larger than the number of successes in the last b experiments is at most a/b .*

Proof. Let the random variables A resp. B denote the number of successes in the first a resp. last b experiments and define $Y := A/(B + 1)$. Then $E(Y) = pa/(pb+1) \leq a/b$. If and only if the number of successes in the first a experiments is larger than in the last b experiments Y is at least 1. Using the Markov bound ([MR95]) we get $P(Y \geq 1) \leq E(Y) \leq a/b$. $\square$

Theorem 2. *Let $p' = \omega(\log(n)/n^2)$. Then the first hitting time of the (1+1) EA on ONEMAX_D is polynomial only with super-polynomially small probability.*

Proof. Because of Theorem 1 we concentrate on values of p' with $p' \leq 1/n^{6/7}$. Let $\alpha(n) = \min \{n/\log(n), p' \cdot n^2/\log(n)\}$ be the minimum of $n/\log(n)$ and the factor by which p' grows faster than $\log(n)/n^2$. Then $\lim_{n \rightarrow \infty} \alpha(n) = \infty$.

Our first assumption on the (1+1) EA is that the initial bit string contains at most $n - 2G$ matching bits, where G grows slightly faster than $\log(n)$, i. e. $G := \log(n) \cdot \alpha(n)^{4/7}$. As $G = o(n)$, Chernoff bounds ([MR95]) guarantee that initializing with a bit string with at most $n - 2G$ matching bits has a probability super-polynomially close to one, i. e. this event is *almost sure*. We show that the (1+1) EA reaches the optimum in polynomially many steps only via a search point whose number of matching bits is between $n - 2G$ and $n - G$ almost surely.

Since the mutation rate of the (1+1) EA is $1/n$, almost surely at most $\log(n)$ matching bits are mutated in one step. If the movement rate p' of the target bit string is $O(1/n)$, the number of moving bits of the target bit string in one step is also at most $\log(n)$ almost surely. Hence, in one step the ONEMAX_D-value of the current search point changes by at most $2\log(n)$ almost surely. As G is growing asymptotically faster than $\log(n)$, it is at least $2\log(n)$ for n large enough. Hence, any run of the (1+1) EA with polynomial many steps starting with at most $n - 2G$ matching bits reaches the optimum only via a search point with at least $n - 2G$ and at most $n - G$ matching bits almost surely.

If $p' = \omega(1/n)$, we have $\alpha(n) = n/\log(n)$. As p' is $O(1/n^{6/7})$ by assumption, Chernoff bounds yield that the number of moving bits in one step is at most $cn^{1/7}$ for a constant $c > 1$ almost surely. Since G is $n^{4/7} \cdot \log(n)^{3/7}$ in this case, i. e. asymptotically larger than the number of moving or mutating bits in one

step, we again know that the (1+1) EA reaches the optimum in polynomial time only via a search point with at least $n - 2G$ and at most $n - G$ bits almost surely.

Let I_t denote the current state of the (1+1) EA, i.e. the number of matching bits of x_t . The preceding argumentation shows that if state n is reached in polynomially many steps, almost surely two points in time $t_1 < t_2$ exist, where $I_{t_1} \in \{n - 2G, \dots, n - G\}$, $I_t \in \{n - G + 1, \dots, n - 1\}$ for all $t \in \{t_1 + 1, \dots, t_2 - 1\}$, and $I_{t_2} = n$. In the following we show that for every $t = t_2 - t_1 \in \mathbb{N}$ it is super-polynomially unlikely that such a sequence $I_{t_1}, \dots, I_{t_2}$ exists (a similar technique is used in [RRS95]).

We exclude the possible, but very unlikely event that the (1+1) EA bridges the gap, i.e. comes from $n - G$ to n ones, using only few but far reaching steps. Therefore, we show that in one step at most $L := \alpha(n)^{1/7}$ non-matching bits flip almost surely. The probability of flipping at least $L + 1$ of the at most $2G$ non-matching bits during one step by mutation and/or movement of the target bit string is at most

$$\binom{2G}{L+1} \cdot \left(\frac{1}{n} + p'\right)^{L+1} \leq \left(\frac{4G}{n^{6/7}}\right)^{L+1},$$

which is super-polynomially small as $G \leq n^{4/7} \log(n)^{3/7}$ and L grows with n to infinity. Hence, we can rule out mutations increasing the number of ones by more than L . The error introduced by this assumption has only super-polynomially small probability if we concentrate on polynomially many steps.

Now, we upper resp. lower bound the probability p_i^+ resp. p_i^- , that the number of matching bits increases resp. decreases during one step when the current number of matching bits is i . The number of matching bits decreases if one matching bit moves, but no non-matching bit moves and no bit mutates. Assuming that $p' \leq 1/n$ we can lower bound p_i^- by (where $c > 0$ is a constant):

$$(1 - (1 - p')^i) \cdot (1 - p')^{n-i} \cdot \left(1 - \frac{1}{n}\right)^n \geq i \cdot p' \cdot c \geq \frac{c}{2} \cdot \alpha(n) \cdot \frac{\log(n)}{n}.$$

If p' is larger than $1/n$, there is a constant probability that at least one of the i matching bits, but none of the $n - i$ non-matching bits moves, while no bit mutates. Hence, the inequality $p_i^- \geq (c/2) \cdot \alpha(n) \cdot (\log(n)/n)$ is still valid for an appropriate constant $c > 0$, because $\alpha(n) = n/\log(n)$ for all $p' \geq 1/n$.

For the upper bound p_i^+ we notice that a step increasing the number of matching bits implies that the mutation resp. the movement of the target bit string flips more non-matching bits than matching bits after the movement resp. mutation has occurred. According to Lemma 1 both events have probability at most $(n - i + L)/(i - L)$ because at most L matching bits change during movement resp. mutation by assumption. Consequently, $p_i^+ \leq 2(n - i + L)/(i - L)$. Since i is at least $n - 2G = n - 2\log(n)\alpha(n)^{4/7}$ and $L = o(G)$, we know that for all $n \geq n_0$ for a constant n_0 (i.e. n is large enough):

$$p_i^+ \leq 2 \cdot \frac{2G + L}{n - 2G - L} \leq 2 \cdot \frac{3G}{n - 3G} \leq 12 \cdot \frac{\alpha(n)^{4/7} \log(n)}{n}.$$

Since it is sufficient for us that decreasing the number of matching bits is by a factor $\alpha(n)^{2/7}$ more likely than increasing it, we lower resp. upper bound p_i^- resp. p_i^+ for n large enough by

$$\tilde{p}_i^- := \frac{c}{2} \cdot \frac{\alpha(n) \log(n)}{n} \quad \text{resp.} \quad \tilde{p}_i^+ := \frac{c}{2} \cdot \frac{\alpha(n)^{5/7} \log(n)}{n}.$$

Hence, the assumption that the probabilities of steps decreasing resp. increasing the number of matching bits are exactly $\tilde{p}_i^-$ resp. $\tilde{p}_i^+$ and that every improvement leads us L bits closer to the target while every deterioration only changes one bit, results in a “faster process” (a proof would follow the line of Lemma 2 in the next section). Then the probability that the next mutation changing the number of matching bits (called *effective mutation*) increases the number of matching bits equals

$$\frac{\frac{c}{2} \cdot \frac{\alpha(n)^{5/7} \log(n)}{n}}{(\alpha(n) + \alpha(n)^{5/7}) \cdot \frac{c}{2} \cdot \frac{\log(n)}{n}} = \frac{\alpha(n)^{5/7}}{\alpha(n) + \alpha(n)^{5/7}} \leq \alpha(n)^{-2/7}.$$

Consequently, the expected number of increasing mutations during t effective mutations is at most $t/\alpha(n)^{2/7}$ and we can follow the argumentation presented in [Dro02]. Again, by assuming that the expected number of increasing mutations is exactly $t/\alpha(n)^{2/7}$ the process can only become faster. To obtain a superpolynomially small upper bound by Chernoff bounds we show that

1. the expected number of increasing mutations grows significantly faster than $\log(n)$, i.e. $t/\alpha(n)^{2/7} = \omega(\log(n))$ and
2. the number of increasing mutations necessary to reach the optimum is by a constant factor larger than the expected number of increasing mutations.

To prove these claims we argue that t , the length of the sequence $I_{t_1}, \dots, I_{t_2}$, is at least $G/L = \log(n)\alpha(n)^{3/7}$, because every increasing mutation raises the number of matching bits by at most $L = \alpha(n)^{1/7}$. Hence, $t/\alpha(n)^{2/7} = \omega(\log(n))$, i.e. only sequences of some minimum length are candidates for $I_{t_1}, \dots, I_{t_2}$. The necessary number of increasing mutations can be lower bounded as follows: if t^+ denotes the number of increasing and t^- the number of decreasing mutations of $t = t^+ + t^-$ effective mutations, the number of matching bits increases by $t^+ \cdot L - t^-$. In order to reach the global optimum this value has to be at least G . This is equivalent to

$$t^+ \cdot L - t^- \geq G \iff t^+ \geq \frac{G + t - t^+}{L} \iff t^+ \geq \frac{G + t}{L + 1}.$$

For n large enough we have

$$\frac{G + t}{L + 1} = \frac{\log(n)\alpha(n)^{4/7} + t}{\alpha(n)^{1/7} + 1} \geq 2 \cdot \frac{t}{\alpha(n)^{2/7}}.$$

In other words, to reach the global optimum in t steps the number of increasing mutations necessary is at least by a factor 2 larger than the expected number

$t/\alpha(n)^{2/7}$ of increasing mutations. Hence, using Chernoff bounds the probability of a sequence $I_{t_1}, \dots, I_{t_2}$ of length t with the desired properties is bounded above by $\exp\left(-\frac{t}{\alpha(n)^{2/7}} \cdot \frac{4}{3}\right)$. Because t is at least $\alpha(n)^{3/7} \log(n)$, this is upper bounded by $\exp\left(-\Omega(\alpha(n)^{1/7} \log(n))\right) = n^{-\Omega(\alpha(n)^{1/7})}$.

All in all, we have shown that the (1+1) EA finds the optimum of ONEMAX_D with $p' = \omega(\log(n)/n^2)$ in polynomially many steps only, if a super-polynomially unlikely event happens. $\square$

4 A Polynomial Upper Bound for Small Movement Rates

We now show that the expected first hitting time of the (1+1) EA for ONEMAX is polynomial for $p' = O(\log(n)/n^2)$. While doing this we have to analyze a process where the distance from the target (the number of non-matching bits) can increase in one step by more than one. The crucial idea is to show how this process can be replaced by a slower one (i. e. how to *upper bound the process*), where the distance from the target in one step increases by at most one.

In the following we describe the (1+1) EA and other processes by their transition probabilities $p_{i,j}$ ($i, j \in \{0, \dots, n\}$) of changing from a point with i matching bits in one step to a point with j matching bits. As all considered processes are Markovian, i. e. time-independent, and operating on $\{0, \dots, n\}$ the transition probabilities describe them completely. Hence, we denote the process with transition probabilities $p_{i,j}$ shortly by $(p, \cdot)$. The random variable $T_{i,j}$ denotes the first hitting time to come from state i to state j . The names are sometimes adapted suitably, e. g. $\tilde{T}_{i,j}$ denotes the first hitting time of the process $(\tilde{p}, \cdot)$.

The outline of the proof is as follows: firstly, we “cut off” all transitions from states i to states $j \geq i + 2$ (we will call processes with $p_{i,j} = 0$ for all i and $j \geq i + 2$ briefly ≤ 1 -processes in the following). This ≤ 1 -process is replaced by one with a stronger tendency towards lower states. Finally, we replace this process by one changing its state from i only to $i - 1$, i , or $i + 1$ and analyze it by standard methods.

To show that all these steps lead to processes having at least the same expected first hitting time, we need a number of lemmata:

Lemma 2. *Let $(p, \cdot)$ be a process. For the ≤ 1 -process $(\tilde{p}, \cdot)$ defined by*

$$\tilde{p}_{i,j} := \begin{cases} p_{i,j} & \text{if } j < i \text{ or } j = i + 1, \\ p_{i,i} + \sum_{j=i+2}^n p_{i,j} & \text{if } j = i, \end{cases}$$

we have for all $i \in \{0, \dots, n-1\}$, $k \geq i$ and all $t \in \mathbb{N}_0$: $P(\tilde{T}_{i,n} \geq t) \geq P(T_{k,n} \geq t)$.

Proof. By induction over $t \in \mathbb{N}_0$: for $t = 0$ the statement is trivial. For the step from t to $t + 1$, we lower bound $P(\tilde{T}_{i,n} \geq t + 1)$ according to two cases:

$$1. \quad k > i : \sum_{j=0}^{i+1} \tilde{p}_{i,j} \cdot P(\tilde{T}_{j,n} \geq t) \geq \sum_{j=0}^{i+1} \tilde{p}_{i,j} \cdot P(T_{k,n} \geq t) \geq P(T_{k,n} \geq t + 1).$$

$$\begin{aligned}
2. \quad k = i : & \sum_{j=0}^{i-1} \tilde{p}_{i,j} \cdot P(\tilde{T}_{j,n} \geq t) + \tilde{p}_{i,i} \cdot P(\tilde{T}_{i,n} \geq t) + \tilde{p}_{i,i+1} \cdot P(\tilde{T}_{i+1,n} \geq t) \\
& \geq \sum_{j=0}^{i-1} p_{i,j} \cdot P(T_{j,n} \geq t) + p_{i,i} \cdot P(T_{i,n} \geq t) + \sum_{j=i+2}^n p_{i,j} \cdot P(T_{j,n} \geq t) \\
& \quad + p_{i,i+1} \cdot P(T_{i+1,n} \geq t) = P(T_{i,n} \geq t + 1).
\end{aligned}$$

□

Since $E(T_{i,n}) = \sum_{t=1}^{\infty} t \cdot P(T_{i,n} = t) = \sum_{t=1}^{\infty} P(T_{i,n} \geq t)$, Lemma 2 tells us that by “cutting off” all improvements by more than one, the expected first hitting time does not decrease. A ≤ 1 -process has two useful properties (for the sake of brevity we omit the easy proofs by induction):

Lemma 3. *Let $(p, \cdot)$ be a ≤ 1 -process. Then we have for all $i \in \{0, \dots, n\}$ and $t \in \mathbb{N}_0$: $P(T_{i,n} \geq t) \geq P(T_{i+1,n} \geq t)$ (implying $E(T_{i,n}) \geq E(T_{i+1,n})$).*

Lemma 4. *Let $(p, \cdot)$ and $(\tilde{p}, \cdot)$ be ≤ 1 -processes. If for all $i \in \{0, \dots, n-1\}$ and all $j < i$ we have $\tilde{p}_{i,j} \geq p_{i,j}$ and $\tilde{p}_{i,i+1} \leq p_{i,i+1}$ then $P(\tilde{T}_{i,n} \geq t) \geq P(T_{i,n} \geq t)$ (implying $E(\tilde{T}_{i,n}) \geq E(T_{i,n})$) for all $i \in \{0, \dots, n\}$ and $t \in \mathbb{N}_0$.*

In general the expected first hitting time of a ≤ 1 -process is difficult to upper bound as it might jump to lower states with high probability. But if this is not too likely we can find a Markov process jumping only from i to $i-1$, i , or $i+1$ (called a $\{-1, 0, 1\}$ -process) having at least the same expected first hitting time. The key idea is to interpret a number of steps of a given $\{-1, 0, 1\}$ -process as a single step of a ≤ 1 -process. If we do not “forget” any transitions of the $\{-1, 0, 1\}$ -process we get a ≤ 1 -process that lower bounds the original $\{-1, 0, 1\}$ -process:

Lemma 5. *Let $(p, \cdot)$ resp. $(\tilde{p}, \cdot)$ be a ≤ 1 - resp. a $\{-1, 0, 1\}$ -process where*

1. $\forall i \in \{1, \dots, n-1\} : p_{i,0} = \prod_{k=1}^i \tilde{p}_{k,k-1}$,
2. $\forall i \in \{1, \dots, n-1\} : \forall j \in \{1, \dots, i-1\} : p_{i,j} = \tilde{p}_{j,j} \prod_{k=j+1}^i \tilde{p}_{k,k-1}$,
3. $\forall i \in \{0, \dots, n-1\} : p_{i,i+1} = \tilde{p}_{i,i+1}$.

Then for all $i \in \{0, \dots, n-1\}$: $E(T_{i,i+1}) \leq E(\tilde{T}_{i,i+1})$.

Proof. Assume that the current state of the process $(\tilde{p}, \cdot)$ is i . Consider the next at most i transitions of $(\tilde{p}, \cdot)$. According to the following cases we split a prefix from this sequence and consider it as a one step of the process $(p, \cdot)$:

- All i transitions decrease the number of the current state by exactly one. This event has probability $\prod_{k=1}^i \tilde{p}_{k,k-1}$ and afterwards the process is in state 0.
- The first $i-j \in \{1, \dots, i-1\}$ transitions decrease the number of the current state by exactly one, while the $(i-j+1)$ -th transition stays in the current state. This event has probability $\tilde{p}_{j,j} \prod_{k=j+1}^i \tilde{p}_{k,k-1}$ and afterwards the process is in state j .
- The first transition leads from state i to $i+1$. This has probability $\tilde{p}_{i,i+1}$ and afterwards the process is in state $i+1$.

- In all other cases, i.e. when the considered i transitions start with at least one but at most $i-1$ decreasing transitions directly followed by an increasing transition, the process is in state at most i . By assuming that the process is exactly in state i , the expected first hitting time does not increase according to Lemma 3, because the described process is a ≤ 1 -process.

These cases show how the $\{-1, 0, 1\}$ -process $(\tilde{p}, \cdot)$ can be considered as a ≤ 1 -process $(p, \cdot)$ with exactly the transition probabilities given above. By interpreting the considered transition(s) as a single transition (observe that their number varies according to the respective case), we get the claimed lower bound. $\square$

Lemma 5 can hardly be applied directly as it only correlates processes where the transition probabilities fulfill some equalities. Using Lemma 4 we can weaken the conditions of Lemma 5 to inequalities:

Corollary 1. *Let $(p, \cdot)$ resp. $(\tilde{p}, \cdot)$ be a ≤ 1 -process resp. a $\{-1, 0, 1\}$ -process on $\{0, \dots, n\}$. If the following conditions hold*

1. $\forall i \in \{1, \dots, n-1\} : p_{i,0} \leq \prod_{k=1}^i \tilde{p}_{k,k-1}$,
2. $\forall i \in \{1, \dots, n-1\} : \forall j \in \{1, \dots, i-1\} : p_{i,j} \leq \tilde{p}_{j,j} \prod_{k=j+1}^i \tilde{p}_{k,k-1}$,
3. $\forall i \in \{0, \dots, n-1\} : p_{i,i+1} \geq \tilde{p}_{i,i+1}$,

then $E(T_{i,n}) \leq E(\tilde{T}_{i,n})$ for all $i \in \{0, \dots, n\}$.

Now we can upper bound the expected first hitting time of the $(1+1)$ EA by repeatedly replacing the original process by simpler ones and using the preceding results to show that the new process is not faster than the former one. As we are interested in asymptotical results, the processes may be defined in a way that only makes sense for n large enough (e.g. some probabilities may be negative for small n). At the end we come up with a $\{-1, 0, 1\}$ -process we can analyze by standard methods.

Theorem 3. *The expected first hitting time of the $(1+1)$ EA on ONEMAX_D with $p' = O(\log(n)/n^2)$ is polynomial.*

Proof. 1. Let $(p^1, \cdot)$ denote the random process exactly describing the $(1+1)$ EA on ONEMAX_D . According to Lemma 2, for the ≤ 1 -process defined by

$$p_{i,j}^2 := \begin{cases} p_{i,j}^1 & \text{if } j < i \text{ or } j = i+1, \\ p_{i,i}^1 + \sum_{j=i+2}^n p_{i,j}^1 & \text{if } j = i, \end{cases}$$

$E(T_{i,n}^2) \geq E(T_{i,n}^1)$ for all $i \in \{0, \dots, n\}$. Hence, from now on we consider the simpler process $(p^2, \cdot)$ instead of $(p^1, \cdot)$.

2. The ≤ 1 -process $(p^2, \cdot)$ is replaced by the ≤ 1 -process $(p^3, \cdot)$, where for all $i \in \{0, \dots, n-1\}$ and $k \in \{1, \dots, i\}$ (remember that $p' = O(\log(n)/n^2)$):

$$p_{i,i-k}^3 := \binom{i}{k} \cdot (p')^k \quad \text{and} \quad p_{i,i+1}^3 := \frac{n-i}{n} \cdot \frac{2}{3 \exp(1)}$$

$(p_{i,i}^3)$ is implicitly defined as the “remaining probability” to stay in state i). According to Lemma 4, this process $(p_{\cdot,\cdot}^3)$ does not have smaller expected first hitting times than the process $(p_{\cdot,\cdot}^2)$, because for all $i \in \{0, \dots, n-1\}$ and $k \in \{1, \dots, i\}$

$$p_{i,i-k}^2 = p_{i,i-k}^1 \leq \binom{i}{k} \cdot (p')^k = p_{i,i-k}^3 \quad \text{and for } n \text{ large enough}$$

$$p_{i,i+1}^2 = p_{i,i+1}^1 \geq \binom{n-i}{1} \cdot \frac{1}{n} \cdot \left(1 - \frac{1}{n}\right)^{n-1} \cdot (1-p')^n \geq p_{i,i+1}^3$$

as $p' = O(\log(n)/n^2)$. Hence, from now on we consider the process $(p_{\cdot,\cdot}^3)$.

3. The $\{-1, 0, 1\}$ -process $(p_{\cdot,\cdot}^4)$ is defined by (for all $i \in \{0, \dots, n-1\}$)

$$p_{i,i-1}^4 := 2 \cdot i \cdot p' \quad \text{and} \quad p_{i,i+1}^4 := \frac{n-i}{n} \cdot \frac{1}{2 \exp(1)}.$$

(Note that $p_{i,i}^4$ is only implicitly defined as $1 - p_{i,i-1}^4 - p_{i,i+1}^4$, which is at least $1/2$ for n large enough, since $p' = O(\log(n)/n^2)$.) Corollary 1 yields that $(p_{\cdot,\cdot}^4)$ upper bounds $(p_{\cdot,\cdot}^3)$: Condition 1 of Corollary 1 is fulfilled, since $p_{i,0}^3 = (p')^i \leq \prod_{k=1}^i 2kp' = \prod_{k=1}^i p_{k,k-1}^4$ for $i \in \{1, \dots, n-1\}$.

Condition 2 holds because for all $i \in \{1, \dots, n-1\}$ and $j \in \{1, \dots, i-1\}$:

$$p_{i,j}^3 \leq p_{j,j}^4 \prod_{k=j+1}^i p_{k,k-1}^4 \iff \binom{i}{i-j} (p')^{i-j} \leq p_{j,j}^4 \prod_{k=j+1}^i 2kp',$$

which is valid, since $p_{j,j}^4 \geq 1/2$ for n large enough. Condition 3 follows directly by $p_{i,i+1}^3 > p_{i,i+1}^4$ for all $i < n$. Now the $\{-1, 0, 1\}$ -process $(p_{\cdot,\cdot}^4)$ can be analyzed with standard methods. It is well-known (see e.g. [DJW00]) that

$$E(T_{i,i+1}^4) = \sum_{k=0}^i \frac{1}{p_{k,k+1}^4} \cdot \prod_{l=k+1}^i \frac{p_{l,l-1}^4}{p_{l,l+1}^4}.$$

Filling in our values for $p_{i,j}^4$, we get the following upper bound on $E(T_{i,n}^4)$ for $p' \leq c \cdot \log(n)/n^2$ (see [Dro02] for a similar calculation):

$$\begin{aligned} & \sum_{k=0}^i \frac{2 \exp(1) \cdot n}{n-k} \cdot \prod_{l=k+1}^i 2 \cdot l \cdot \frac{c \cdot \log(n)}{n^2} \cdot \frac{2 \exp(1) \cdot n}{n-l} \\ &= \sum_{k=0}^i \frac{2 \exp(1) \cdot n}{n-k} \cdot \left(\frac{4c \exp(1) \log(n)}{n} \right)^{i-k} \cdot \frac{i!}{k!} \cdot \frac{(n-i-1)!}{(n-k-1)!} \\ &\leq \frac{2 \exp(1) \cdot n}{n-i} \cdot \sum_{k=0}^i \left(\frac{4c \exp(1) \log(n)}{n} \right)^{i-k} \cdot \binom{i}{k} \Big/ \binom{n-k-1}{n-i-1} \\ &\leq 2 \exp(1) \cdot \frac{n}{n-i} \cdot \left(1 + \frac{4c \exp(1) \log(n)}{n} \right)^i \end{aligned}$$

Using the estimation $(1 + 1/x)^x \leq \exp(1)$, $E(T_{i,i+1}^A)$ is bounded above by:

$$2 \exp(1) \cdot \frac{n}{n-i} \cdot \exp\left(\frac{4c \log(n) \exp(1)}{n} \cdot i\right) \leq 2 \exp(1) \cdot \frac{n}{n-i} \cdot n^{\frac{4c \exp(1)}{\ln(2)}}.$$

Hence, by linearity of expectation and pessimistically assuming that we initialize in $(0, \dots, 0)$,

$$E(T_{\text{ONEMAX}_D}) \leq 2 \exp(1) \cdot n^{\frac{4c \exp(1)}{\ln(2)} + 1} \cdot \sum_{i=1}^n \frac{1}{i} = O\left(n^{4c \cdot \frac{\exp(1)}{\ln(2)} + 1} \cdot \log(n)\right).$$

□

5 Conclusions

We have analyzed the $(1+1)$ EA on a dynamic variant of ONEMAX where every bit of the target flips with probability p' . The critical growth rate of p' where the expected first hitting time of the $(1+1)$ EA changes from polynomial to super-polynomial was shown to be $\Theta(\log(n)/n^2)$. Besides this result, the techniques developed may be of more general interest. They show how a Markov process which can move away from the target by more than one in one step can be replaced by a slower Markov process that can move away from the target by at most one in one step (Lemma 5 resp. Corollary 1). This method might help in the analysis of other EAs or randomized processes in general.

Acknowledgements. I thank Jens Jägersküpper, Ingo Wegener, and Carsten Witt for their valuable advice and help while preparing this paper.

References

- [BFM97] Th. Bäck, D.B. Fogel, and Z. Michalewicz, editors. *Handbook of Evolutionary Computation*. Institute of Physics Publishing, 1997.
- [Bra01] J. Branke. *Evolutionary Optimization in Dynamic Environments*. Kluwer Academic, 2001.
- [DJW98] S. Droste, Th. Jansen, and I. Wegener. A rigorous complexity analysis of the $(1+1)$ EA for linear functions with Boolean inputs. In *Proceedings of ICEC 1998*, pages 499–504, 1998.
- [DJW00] S. Droste, Th. Jansen, and I. Wegener. Dynamic parameter control in simple evolutionary algorithms. In *Proceedings of FOGA 2000*, pages 275–294, 2000.
- [DJW02] S. Droste, Th. Jansen, and I. Wegener. On the analysis of the $(1+1)$ EA. *Theoretical Computer Science*, (276):51–81, 2002.
- [Dro02] S. Droste. Analysis of the $(1+1)$ EA for a dynamically changing OneMax-variant. In *Proceedings of CEC 2002*, pages 55–60, 2002.
- [Hol75] J.H. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, 1975.

- [JW99] Th. Jansen and I. Wegener. On the analysis of evolutionary algorithms – a proof that crossover really can help. In *Proceedings of ESA 1999*, pages 184–193, 1999. LNCS 1643.
- [JW01] Th. Jansen and I. Wegener. Real royal road functions – where crossover is provably essential. In *Proceedings of GECCO 2001*, pages 1034–1041, 2001.
- [MR95] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995.
- [Müh92] H. Mühlenbein. How genetic algorithms really work: I. mutation and hill-climbing. In *Proceedings of PPSN 1992*, pages 15–26, 1992.
- [RRS95] Y. Rabani, Y. Rabinovich, and A. Sinclair. A computational view of population genetics. In *Proceedings of STOC 1995*, pages 83–92, 1995.
- [Rud97] G. Rudolph. *Convergence Properties of Evolutionary Algorithms*. Verlag Dr. Kovač, 1997.
- [SD99] S.A. Stanhope and J.M. Daida. (1+1) GA fitness dynamics in a changing environment. In *Proceedings of the CEC 1999*, pages 1851–1858, 1999.

Performance Evaluation and Population Reduction for a Self Adaptive Hybrid Genetic Algorithm (SAHGA)

Felipe P. Espinoza¹, Barbara S. Minsker¹, and David E. Goldberg²

¹ University of Illinois, Department of Civil and Environmental Engineering,
205 N. Matthews Ave, Urbana IL 61801
(fespinoz, minsker)@uiuc.edu

² University of Illinois, Department of General Engineering,
104 N. Matthews Ave, Urbana IL 61801
deg@uiuc.edu

Abstract. This paper examines the effects of local search on hybrid genetic algorithm performance and population sizing. It compares the performance of a self-adaptive hybrid genetic algorithm (SAHGA) to a non-adaptive hybrid genetic algorithm (NAHGA) and the simple genetic algorithm (SGA) on eight different test functions, including unimodal, multimodal and constrained optimization problems. The results show that the hybrid genetic algorithm substantially reduces required population sizes because of the reduction in population variance. The adaptive nature of the SAHGA algorithm together with the reduction in population size allow for faster solution of the test problems without sacrificing solution quality.

1 Introduction

Hybrid genetic algorithms (HGAs) are a natural extension to genetic algorithms (GA), combining the genetic algorithm's global search capabilities with the strengths of local search algorithms. These algorithms have been used in a number of different fields, including transportation engineering [8], water resources modeling [4], operations research [12], and groundwater management [11] to name a few. For example in the so-called "Bicriteria Transportation Problem" presented in [8], the GA was combined with the traditional simplex method to solve linear problems to create the HGA. Another example is the groundwater management problem. The HGA presented in [11] was created by combining the GA with constrained differential dynamic programming. In these applications and others, the HGA and the GA solved the problem using the same population size, and the local search step is applied in most of the cases to a small number of individuals in the population generation after generation. In this study, we analyze the effect of the local search component of the HGA on the search and on reduction of the search space. This reduction reduces the population size necessary to solve the HGA problem relative to the SGA for the same solution reliability, thereby decreasing the computational burden.

Section 2 gives an overview of the HGA algorithms analyzed in this work and Section 3 presents the test functions that were used to evaluate performance of the

HGA algorithms in Section 4. Finally, Section 5 presents our conclusions and recommendations.

2 SAHGA and NAHGA Algorithms

The two hybrid genetic algorithms (HGA) used in this work are NAGHA (non adaptive hybrid genetic algorithm) and SAGHA (self adaptive hybrid genetic algorithm) were originally proposed by [7]. A brief description of the algorithms is presented next. The full details of these algorithms can be found in [7].

2.1 Non-adaptive Hybrid Genetic Algorithm (NAHGA)

The NAHGA algorithm combines a simple genetic algorithm (SGA) with local search. The local search step is defined by three basic parameters: local search frequency, probability of local search, and number of local search iterations. Local search frequency determines how frequently local search is invoked (e.g., every 3 iterations); probability of local search represents the fraction of individuals in the population that undergo local search at each invocation; and number of local search iterations represents the number of local search steps performed at each invocation.

2.2 Self-Adaptive Hybrid Genetic Algorithm (SAHGA)

The SAHGA algorithm works with the same operators as the NAHGA algorithm: local search frequency, probability of local search and number of local search iterations. The major difference in the approaches is that the SAHGA adapts in response to recent performance of the algorithm as it converges to the solution. In other words, the operators are used only when they can provide new information to the search. The adaptation process for the three parameters is different. The global-local switch is adapted by evaluating the ratio of the coefficient of variation of the fitness between two consecutive generations. Local search is performed when this ratio is greater than a specified local search threshold. This approach ensures that local search is only performed when the coefficient of variation is significantly increasing, which indicates that a new area of decision space is being searched and local search is needed. The second adaptive parameter is the probability of local search, which is decreased from the initial value at the beginning of every local search step. Finally, the number of local search steps is controlled by comparing the improvement attained in the local search step with the improvement attained by global search. When local search no longer improves average fitness more than the most recent global search iteration, the search reverts to global search.

3 Test Functions

In order to evaluate the performance of the algorithm, the following 8 test functions were selected from the GA literature:

- Unimodal problems: De Jong's 1 and De Jong's 2 [5]
- Multimodal problems with the same local minimum at different locations: Branin [3] and Six Hump [6]
- 2 multimodal problems with different optima: Schwefel [17] and Griewank [1]
- 2 constrained unimodal problems: Test04 [13] and Bracken & McCormick [2]

4 Evaluation of HGA Performance

This section is divided in 5 sub-sections: local search algorithms, population size evaluation for the SGA, standard deviation reduction, population size for the HGA, and analysis of results.

4.1 Local Search Algorithm

The local search operator attempts to find the best solution starting at a previously selected point, in this case a solution in the SGA population. For this analysis, 5 local search algorithms were selected. These algorithms are:

- Random Walk with Uniform Distribution (LS1): In general, the random walk is simply the movement from one point of the decision space to a new point randomly selected using a uniform distribution from a neighborhood around the starting point [18]. One iteration of this algorithm requires one fitness function evaluation.
- Random Walk with Normal Distribution (LS2): This algorithm is similar to the uniform distribution discussed previously, but the change of location is evaluated with a normal distribution instead of a uniform distribution. For this reason, the points located near the starting point are more likely to be selected than those located closer to the boundary of the search area. Again, one iteration requires one function evaluation.
- (1+1)-Evolutionary Strategy (LS3): This algorithm, proposed by [14] and [16], randomly selects a new location using a normal distribution with variable standard deviation. The standard deviation changes following the so-called $\frac{1}{5}$ success rule based on the evaluation of success of the search. This algorithm also requires one fitness function evaluation per local search iteration.
- Random Derivative (LS4): This algorithm randomly selects a search direction, and using this direction, the location of a new point is evaluated. This algorithm needs 2 fitness function evaluation for every local search iteration (one for the evaluation of the coordinates of the new point and one for its fitness).
- Steepest Descent (LS5): The algorithm evaluates the new point following the direction of the gradient of the function at the starting point. This algorithm performs one function evaluation for every one of the decision variables of the

particular test problem (to numerically evaluate gradient) and one function evaluation to evaluate the fitness of the new individual.

4.2 Population Size for Genetic Algorithm

One of the critical elements for optimal performance of a GA is the population size. For the population size evaluation, a relationship derived from the random walk model proposed by [9] is used. In their approach the population size is given by:

$$N \geq -2^{K-1} \ln(\alpha) \left(\frac{\sigma_f}{d} \right) \quad (1)$$

In Eq. (1), K represents the building block (BB) order, α is the reliability (i.e., the probability that the GA finds the optimal solution); σ_f is the standard deviation of the fitness function, and d is the signal difference between the best and second best solution. The parameters σ_f and d are estimated using a large, random initial population. For these test problems, d was estimated from the differences in fitness of the best members of the population. In this case the analysis was performed by means of a probabilistic approach based on frequency analysis theory [10], in which the histogram of the fitness function is evaluated. This histogram represents different “classes” of fitness that can be statistically identified. Using the histogram, “d” is evaluated as the size of the first class of the histogram. The building block order, K, is unknown but can be assumed to vary between 1 and 5 [15].

Using this information, a range of initial population sizes for the SGA was estimated from Eq. (1) for all 8 test problems for different values of parameter K, as shown in Table 1.

The final step in the analysis is the selection of the starting population from the values given in Table 1. For this analysis, the convergence time must be less than the drift time. The convergence time is evaluated with the relation $t \approx 2l$ (where l is the chromosome length) [19]. The drift time is given by $t_{\text{drift}} \approx 1.4 N$ (where N is the population size) [20]. Imposing the condition that convergence time must be less than drift time to obtain the correct solution, the population size must satisfy the relation presented in Eq. (2) (lower boundary for population size):

$$N > 1.42 l \quad (2)$$

Table 1. Population size (SGA) for different values of K

Building Block Order (K)	Test Function							
	DJ1	DJ2	Branin	Six-Hump	Schwefel	Griewank	Test04	B&M
1	20	12	20	10	35	20	16	24
2	40	24	40	20	70	40	32	48
3	80	48	80	40	140	80	64	96
4	160	96	160	80	280	160	128	192
5	320	192	320	160	560	320	256	384

Table 2 shows the chromosome length for each test function and the corresponding lower boundary for the population evaluated with Eq. (2). The table also includes the

minimum building block order that satisfies Eq. (2) (see Table 1) and the adopted population for the SGA runs.

Table 2. Adopted SGA Population for all 8 problems

Function	Chromosome Length	Lower Boundary	Minimum Building Block Order (K)	Adopted Population
DJ1	150	213	5	320
DJ2	60	83	4	96
Branin	60	83	4	160
Six-Hump	60	83	5	160
Schwefel	150	213	4	280
Griewank	60	86	4	160
Test04	60	83	4	128
B&M	60	86	3	96

4.3 Standard Deviation Reduction: Local Search Effect on Population Size

For the HGA case, Eq. (1) is modified to include the effect of local search. During local search, the diversity of the initial population is decreased because multiple members of the population usually have the same nearby local optima. The following example contains the results of local search on the multimodal function Griewank for a population of 20 individuals. Fig. 1 shows the fitness for a random population before and after local search (uniform random and gradient). The line represents the average population fitness. From Fig. 1, it is clear that the standard deviation of the fitness for the population after local search is substantially reduced. Initially the average is 30 and the standard deviation is 27.1. After 3 iterations (60 function evaluations) uniform random search reduces the standard deviation to 24.6 and the average to 25.2 (see open squares and dashed line). Steepest descent local search performed 1 iteration (60 function evaluations) and changed the standard deviation to 7.3 and reduced the average to 12.1 (see open circles and solid line). Steepest descent local search lowered the standard deviation and the average by an amount larger than uniform random local search while using the same 60 function evaluations. Both methods of local search performed the same number of function evaluations, however the gradient method only did 1 iteration while the uniform random method did 3 iterations in order to evaluate the results for the same amount of effort. These results show that gradient search decreases diversity faster than random search.

The local search reduction effect can be modeled as $\sigma_n = \beta \sigma_f$, where $0 \leq \beta < 1$ is the standard deviation reduction by local search. Eq. (1) can be rewritten as:

$$N \geq -2^{K-1} \ln(\alpha) \left(\beta \frac{\sigma_f}{d} \right) \quad (3)$$

Eq. (3) shows that the population for the HGA algorithms should be directly proportional to the SGA population.

The parameter β can be estimated by applying local search to the initial random population for a predefined number of iterations. For the current application, local search was applied to a population of 1,000 individuals for a total of 10 iterations.

Fig. 2 shows the change of standard deviation with respect to the standard deviation of the initial population for the selected test problem (Griewank). Since most of the local search steps involved fewer than 6 iterations, the value used for posterior evaluations was the average value of β across the first 5 iterations.

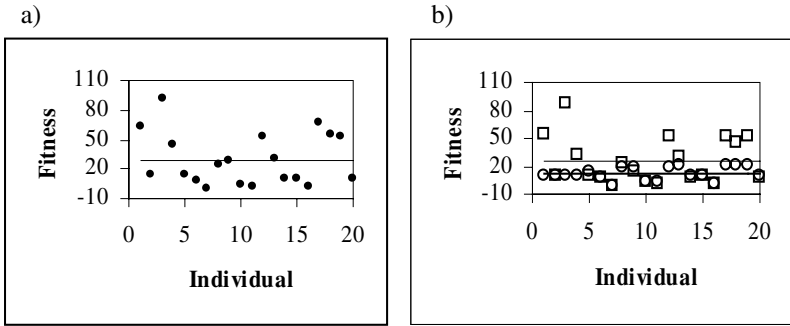


Fig. 1. Local search effect on fitness: a) before local search, b) after local search

In Fig. 2, gradient (LS5) has the largest effect on standard deviation; 1 iteration of this algorithm is equivalent to 3 iterations of LS1, LS2 or LS3. The random derivative (LS4) method is the second most effective local search method at decreasing the standard deviation of the population. The first three local search methods also have substantial effects on the standard deviation, especially if the amount of effort involved is taken into consideration.

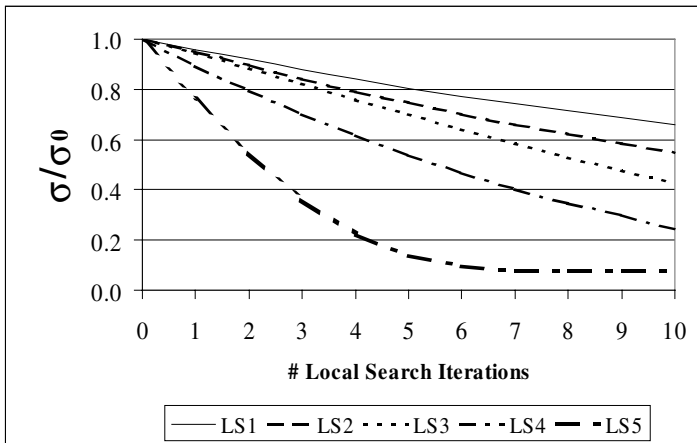


Fig. 2. Standard Deviation Reduction, β , for Griewank Function

4.4 Population Size for Hybrid Algorithms

Using Eq. (3) and the value of β estimated for every one of the 5 algorithms, Table 3 displays the HGA population results for every test problem and local search algorithm. From the results, it is clear that the effect of the different local search algorithms is the same for DJ1 and Six-Hump. On the other hand, for Griewank the population size is different for every one of the 5 local search algorithms in consideration because of the different reductions in standard deviation.

Table 3. HGA Population

Test Function	Local Search Algorithm				
	LS1	LS2	LS3	LS4	LS5
DJ1	288	288	288	288	288
DJ2	64	64	64	40	40
Branin	96	96	80	48	48
Six-Hump	96	96	96	96	96
Schwefel	240	224	216	160	160
Griewank	144	136	128	112	64
Test04	120	112	112	112	80
B&M	92	88	88	48	48

4.5 HGA Performance

The HGA performance results consist of 2 types of data: results evaluated for one population for a default set of parameters defined in [7], and results evaluated using Monte Carlo simulation for different combinations of parameters also defined in [7].

The first set of results for Griewank can be presented in two graphs. The first graph (Fig. 3) plots the best fitness in the population during the SGA and SAHGA search processes, with SAHGA using the 5 local search techniques. The results are presented in terms of number of function evaluations because the adaptive local search uses different numbers of function evaluations in each generation. The results presented also only include the results attained by the application of SAHGA because SAHGA performance is consistently better for the 8 test problems.

Fig. 3 shows that the SGA is inferior to SAHGA with all five local search algorithms. SGA required the most function evaluations with a total of 8,800. Steepest descent (LS5) performed only 4,896 function evaluations with a population of 64, a reduction of 44%. The steepest descent method's speed and its ability to reduce the required population offset the fact that it is the most expensive method. In fact, its performance is better than the performance of normal random search (LS2) and (1+1)-ES (LS3).

The second result, shown in Fig. 4, is the convergence ratio, which is defined as the number of individuals with fitness equal to or near the fitness of the best individual. Fig. 4 shows that the results attained by SAHGA are again better than the SGA for all 5 local search approaches. The results from the other 7 test functions are not shown because they present the same trend, with SAHGA performing better than the SGA by 5% to 33%.

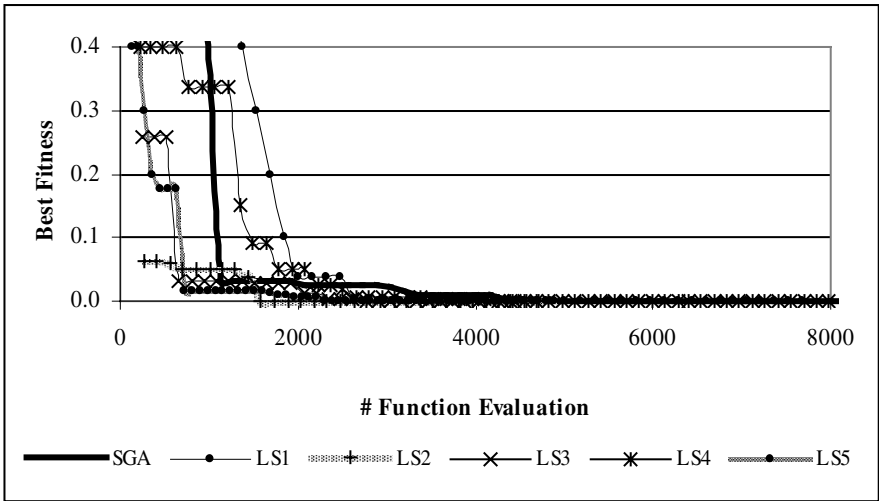


Fig. 3. Best Fitness for Griewank

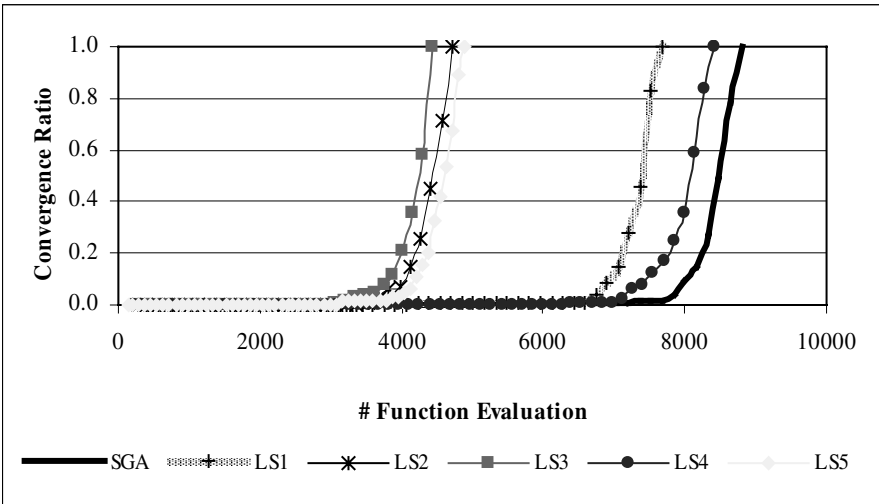


Fig. 4. Convergence Ratio for Griewank

The solution quality of the algorithms was also evaluated by performing 100 Monte Carlo simulations over the initial population for every one of the parameter combinations described in [7]. Table 4 shows the results of these simulations. The results are presented for the best, default and worst parameter combinations. For SAHGA, the average number of function evaluations for the default and best set of parameters are always better than the results attained with the SGA, and the difference between the best set of results and the default varies between 2% and 8%. On the other hand, for NAHGA sometimes the results associated with the default set

of parameters are worse than the results attained with the SGA, and the difference between the best set of results and the default set of results varies between 10% and 25% for different local search algorithms. This shows that parameter estimation is critical to attaining good performance for the NAHGA, but the adaptive capabilities of SAHGA reduce the need for careful parameter selection. For both algorithms, the worst set of results was attained when all the individuals are undergoing local search.

Table 4. Monte Carlo Simulation Results
Average Number of Function Evaluations

Local Search	SGA	SAHGA			NAHGA		
		Worst	Default	Best	Worst	Default	Best
LS1	4,326	6,851	4,020	3,906	9,462	4,373	3,936
LS2	4,326	6,470	3,800	3,720	8,936	4,131	3,717
LS3	4,326	6,402	3,775	3,660	8,723	4,036	3,478
LS4	4,326	13,776	3,964	3,666	14,337	4,645	3,508
LS5	4,326	7,905	3,503	3,292	8,961	4,153	3,071

For the other test problems in this study, the results are similar. The only difference is that the best local search algorithm sometimes varies among gradient, (1+1)-ES, uniform random walk, normal random walk, and random derivative. This phenomenon is related to the differences in standard deviation reduction of each algorithm for different test functions. For example, when the differences in standard deviation reduction between steepest descent and random search is not significant (see Table 3 for functions DJ1 and Six-Hump), random search is more likely to perform better than steepest descent given the big difference in the effort necessary to apply the algorithms. On the other hand, when the difference is significant (see Griewank function in Table 3), steepest descent is better than random walk.

Fig. 5 shows the reliability of SGA, SAHGA and NAHGA for the default set of parameters. SAGHA reliability was in the range of 98-100%, while solving the problem over a much smaller population than the SGA, which means it is less expensive. NAGHA is very efficient as well, with reliability values of 96% to 100%, again with smaller population sizes than the SGA. SAHGA and NAHGA have somewhat lower reliabilities for LS4 and LS5 because the extensive standard deviation reduction associated with the gradient-based algorithms sometimes causes premature convergence due to loss of diversity in the population.

5 Recommendations and Conclusions

The results presented in this paper indicate that the local search capabilities of the HGA algorithm enabled robust solution of complex, unimodal, multi-modal, and constrained problems with less effort than the SGA. Of the two hybrid algorithms, SAHGA is always superior to NAHGA because near-optimal performance is attained for a broad range of parameters. Another result is related to the importance of the

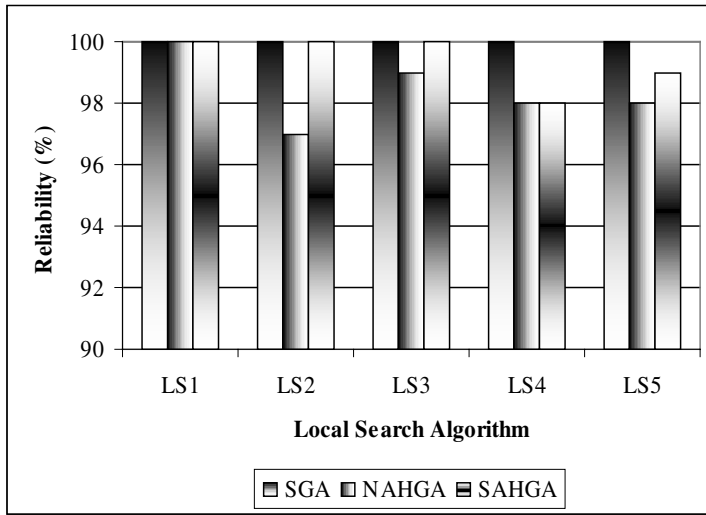


Fig. 5. SGA and HGA Reliabilities for Default Set of Parameters

local search algorithm selected. The results show that the best performance is not always attained for the same local search algorithm across all test functions, mostly because of the difference in effort necessary to apply the different local search algorithms. For any function, the most suitable local search algorithm can be pre-selected using only the population sizing analysis shown in Table 3. When the difference between populations (for different local search algorithms) is not significant, it is more likely that a random search algorithm (which requires only 1 function evaluation per local search iteration) will perform better than steepest descent (which requires $n+1$ function evaluations per iteration, with n the number of decision variables).

Another important result derived from this research is related to the setting of population sizing in Section 4. This process allows us to evaluate a good estimate of the population to achieve optimal performance, but there is a possibility that the same level of reliability is achieved with a smaller population. With respect to the HGA, the analysis presented shows clearly how local search reduces the standard deviation of fitness in the population, which in turn reduces the required population size to achieve the same level of reliability. This reduced population together with the HGA reduces the total number of function evaluations by 44% on average. In this way, the HGA is almost twice as fast as the original GA.

The next step of this research will be to analyze possible ways to improve the performance of the algorithm with a more detailed study of the processes involved. Finally, the algorithm will be applied to solve real world problems.

Acknowledgments. This material is based upon work supported by the National Science Foundation under Grant No. BES 97-34076 CAR and the U. S. Army Research Office under Grant No. DAAD 19-00-7-0389.

Disclaimer. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the author(s) and do not necessarily reflect the views of the National Science Foundation or the U. S. Army Research Office.

References

1. Bäck, T., D. Fogel and Z. Michalewicz, (eds): Handbook of Evolutionary Computation, Bristol and New York. Institute of Physics Publishing Ltd and Oxford University Press (1997)
2. Bracken, J. G. P. McCormick: Ausgewählte Anwendungen Nichtlinearer Programmierung. Berliner Union and Kohlhammer, Stuttgart (1970)
3. Branin, F. K.: A Widely Convergent Method for Finding Multiple Solutions of Simultaneous Nonlinear Equations. IBM J. Res. Develop., pp. 504-522. (1972)
4. Cai, X., McKinney, D. and Lasdon. L.: Solving Nonlinear Water Management Models Using a Combined Genetic Algorithm and Linear Programming Approach. Advances in Water Resources, 24, 667-676. (2001)
5. De Jong, K. A.: An Analysis of the Behavior of a Class of Genetic Adaptive Systems. Ph.D. Dissertation, University of Michigan, Ann Arbor, MI. (1975)
6. Dixon, L. C. W. and Szego, G. P.: The Optimization Problem: An Introduction. In Dixon, L. C. W. and Szego, G. P. (Eds.): Towards Global Optimization II, New York: North Holland. (1978)
7. Espinoza, F., B. S. Minsker, and D. Goldberg. (2001). "A Self Adaptive Hybrid Genetic Algorithm". L. Spector, E. Goodman, A. Wu, W.B. Langdon, H.-M. Voigt, M. Gen, S. Sen, M. Dorigo, S. Pezeshk, M. Garzon, and E. Burke, editors. 2001. Proceedings of the Genetic and Evolutionary Computation Conference, GECCO'2001. San Francisco, Morgan Kaufmann Publishers.
8. Gen, M., Ida, K., and Li, Y.: Bicriteria Transportation Problem by Hybrid Genetic Algorithm. Computers & Industrial Engineering, 35(1-2), 363-366. (1998)
9. Harik, G.R., Cantú-Paz, E., Goldberg, D. E., and Miller, B. L.: The Gambler's Ruin Problem, Genetic Algorithms, and the Sizing of Populations." In Proceedings of the 1997 IEEE Conference on Evolutionary Computation, pp. 7-12, IEEE Press, New York, NY. (1997)
10. Hogg, R., and Craig: A. Introduction to Mathematical Statistics. Macmillan Publishing Co., Inc., New York. (1978)
11. Hsiao, C. and Chang, L.: Dynamic Optical Groundwater Management With Inclusion Of Fixed Costs. Journal of Water Resources Planning and Management, ASCE, 128(1), 57-65. (2002)
12. Lin, W., Delgado-Frias, J, Gause, D., and Vassiliadis, S.: Hybrid Newton-Raphson Genetic Algorithm for the Traveling Salesman Problem. Cybernetics & Systems, 26(4), 387-412. (1995)
13. Kim, J-H. and H. Myung: Evolutionary Programming Techniques for Constrained Optimization Problems. Evolutionary Computation, 1(2), 129-140. (1997)
14. Rechenberg, I.: Ecolutionsstrategie: Optimierung Technischer Systeme Nach Prinzipien Der Biologischen Evolution. Frommann-Iolzboog Verlag, Stuttgart. (1973)

15. Reed, P., Minsker, B. S., and Goldberg, D. E.: Designing a Competent Simple Genetic Algorithm for Search and Optimization. *Water Resources Research*, 36(12), 3757-3761. (2000)
16. Schwefel, H. P.: *Evolutionsstrategie und Numerische Optimierung*. PhD Dissertation, Department of Process Engineering, Technical University of Berlin, Berlin, Germany. (1975)
17. Schwefel, H. P.: *Numerical Optimization of Computer Models*. John Wiley & Sons, Chichester-New York-Brisbane-Toronto, (1981)
18. Spitzer, F.: *Principles of random walk*. D. Van Nostrand Company, Inc. (1964)
19. Thierens, D., Goldberg, D. E., and Guimaraes Pereira: A. Domino Convergence, Drift, and the Temporal-Salience Structure of Problems. In *The 1998 IEEE International Conference on Evolutionary Computation Proceedings*, pp. 535-540, IEEE Press, New York, NY, (1998)

Schema Analysis of Average Fitness in Multiplicative Landscape

Hiroshi Furutani

Kyoto University of Education,
Fushimi-ku, Kyoto, 612-8522 Japan
furutani@kyokyo-u.ac.jp

Abstract. By applying the schema theorem, we study the effects of crossover in Genetic Algorithms with the multiplicative fitness function. On this landscape, the analytical expression of the exact schema theorem can be obtained, and this makes it possible to carry out the mathematical investigation of genetic operators. We consider the average fitness under the action of selection, mutation and crossover. To do this, we give the expressions for the average and variance of fitness in terms of schema frequencies. The theoretical results are compared with numerical experiments.

1 Introduction

It is in general very difficult to know the effects of crossover and mutation in Genetic Algorithms (GAs). Even now, we cannot predict the behavior of a population in a given GA calculation under crossover and mutation. In this paper, we consider this problem by applying the method of schema analysis [1].

The schema theorem was proposed by Holland [2], and used in the theoretical analysis of GAs. However, there have been also given many criticisms on its effectiveness. For example, it only takes into account the destructive nature of crossover and mutation on schemata without considering their constructive roles [1].

There are several attempts to make the schema theory more quantitative. Stephens and Waelbroeck obtained exact schema evolution equations for mutation and crossover [3,4]. Vose has pointed out a close relationship between the schema frequency and the Walsh transform of genotype frequency [5]. Wright derived a version of exact schema theorem by using the Walsh transformation method of Vose [6].

Recently, we have also developed a method to derive a schema theorem from a evolution equation of genotype frequency by applying Walsh transformation [7]. Fortunately, it was found recently that evolution equations for mutation and crossover can be expressed in very simple forms by Walsh transformation [5, 8]. Therefore, it is not difficult to obtain the schema evolution equations for mutation and crossover [1].

Selection is the most important operator in genetics and GA, and plays the role of a driving force in evolution. In this paper, we consider the evolution of

a population on the multiplicative landscape. Though there are many papers on this landscape [9,10,11], we study this problem again from the viewpoint of schema evolution. Previously, we have shown that the evolution equation for that population can be solved exactly [12]. In this paper, we apply the schema theorem to this problem, and study the roles of genetic operators microscopically. We derive an exact evolution equation of schemata evolving under the action of selection, mutation and crossover. The effect of linkage on the average fitness is studied mathematically. The theoretical results are compared with numerical experiments.

2 Mathematical Model

2.1 Model

In this paper, we study the action of selection, mutation and crossover in GAs. We use the fitness proportionate selection and uniform crossover. A population is assumed to evolve in discrete and non-overlapping generations, and the process is described by a set of difference equations. We consider GAs of infinite population size, and neglect the effect of random sampling.

An individual is represented by a binary string of length ℓ , and thus there are $n = 2^\ell$ possible genotypes, or strings. The i th genotype B_i is defined as $B_i = \langle i(\ell), \dots, i(1) \rangle$. The integer i is sometimes identified with B_i . We use the notation $|i|$ for the number of bit ones in i

$$|i| = \sum_{k=1}^{\ell} i(k). \quad (1)$$

We use the relative frequency $x_i(t)$ of the genotype B_i at generation t for the analysis. The relative frequencies satisfy the normalization condition

$$\sum_{i=0}^{n-1} x_i(t) = 1. \quad (2)$$

2.2 Walsh Transformation

The evolution of a GA system can also be described by the Walsh transform of $x_j(t)$

$$\tilde{x}_i(t) \equiv \sum_{j=0}^{n-1} W_{ij} x_j(t), \quad (3)$$

where Walsh function W_{ij} is

$$W_{ij} \equiv W_i(j) = \prod_{k=1}^{\ell} (-1)^{i(k) \cdot j(k)}, \quad (4)$$

and we call $\tilde{x}_i$ as Walsh coefficient. When it is necessary to show the number of bit ones and their positions in i

$$k = |i|, \quad 1 \leq b_1 < \dots < b_k \leq \ell,$$

we use the notation

$$\tilde{x}_i(t) = \tilde{x}^{(k)}[b_1, \dots, b_k](t), \quad (5)$$

where k is the degree of Walsh coefficient.

Substituting $W_{0j} = 1$ into equation (3), and using the normalization condition (2), we have the zeroth order Walsh coefficient $\tilde{x}_0$

$$\tilde{x}_0(t) = \sum_{j=0}^{n-1} x_j(t) = 1. \quad (6)$$

The following relation will be used later

$$\tilde{x}^{(k)}[b_1, b_2 \dots b_k] = \sum_{j=0}^{n-1} \prod_{m=1}^k (-1)^{j(b_m)} x_j, \quad (7)$$

where $j(b_1), \dots, j(b_k)$ stand for the bit values of j at the positions of ones in i .

2.3 Evolution Equation

In proportionate selection, the frequency of genotype B_i at generation $t + 1$ is given in terms of frequencies at generation t

$$x_i(t+1) = \frac{f_i}{\bar{f}(t)} x_i(t) \quad (i = 0, \dots, n-1), \quad (8)$$

where f_i is a fitness of B_i , and the average fitness of the population $\bar{f}(t)$

$$\bar{f}(t) = \sum_{i=0}^{n-1} f_i x_i(t). \quad (9)$$

There is a very important relation for the selection process. The change in average fitness per generation

$$\Delta \bar{f}(t) = \bar{f}(t+1) - \bar{f}(t),$$

is given by

$$\begin{aligned} \Delta \bar{f}(t) &= \frac{1}{\bar{f}(t)} V(f), \\ V(f) &= \sum_i f_i^2 x_i(t) - \bar{f}(t)^2. \end{aligned} \quad (10)$$

This is a version of Fisher's fundamental theorem of natural selection [13]. Fisher's theorem means that the increase in the average fitness is proportional to the ratio of the variance of fitness $V(f)$ to its average.

The Walsh transform of the evolution equation (8) becomes

$$\widehat{S}\tilde{x}_i(t) = \frac{1}{n\bar{f}(t)} \sum_{j=0}^{n-1} \tilde{f}_j \tilde{x}_{i \oplus j}(t), \quad (11)$$

where $\oplus$ is the bitwise exclusive-or operation, and $\widehat{S}$ shows the effect of selection symbolically. The fitness $\tilde{f}_i$ is the Walsh transform of f_i

$$\tilde{f}_i = \sum_{j=0}^{n-1} W_{ij} f_j. \quad (12)$$

The average fitness is given by

$$\bar{f}(t) = \frac{1}{n} \sum_{j=0}^{n-1} \tilde{f}_j \tilde{x}_j(t). \quad (13)$$

The change in the frequency under mutation is

$$x_i(t+1) = \sum_{j=0}^{n-1} M_{ij} x_j(t), \quad (14)$$

where the mutation matrix M_{ij} stands for the probability of mutation from B_j to B_i over one generation. An (i, j) th element of M is a function of the Hamming distance $d(i, j)$ between strings i and j .

$$M_{ij} = (1-p)^{\ell-d(i,j)} p^{d(i,j)}, \quad (15)$$

where p denotes the mutation rate at one bit position over one generation.

The matrix M_{ij} can be diagonalized by Walsh functions

$$\sum_{i'=0}^{n-1} \sum_{j'=0}^{n-1} W_{ii'} M_{i'j'} W_{j'j} = n (1-2p)^{|i|} \delta_{i,j}.$$

Thus the Walsh transform of the evolution equation under mutation (14) is

$$\widehat{M}\tilde{x}_i(t) = (1-2p)^{|i|} \tilde{x}_i(t), \quad (16)$$

where $\widehat{M}$ symbolically shows the effect of mutation.

The evolution equation under crossover can be given in terms of crossover tensor C

$$x_k(t+1) = \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} C(k|i, j) x_i(t) x_j(t). \quad (17)$$

Using the Walsh transformation, we can express the effect of crossover in a very simple form [5,8]

$$\widehat{C} \tilde{x}_k(t) = \sum_{i=0}^{n-1} c_{i,i \oplus k} \tilde{x}_i(t) \tilde{x}_{i \oplus k}(t), \quad (18)$$

where $\widehat{C}$ denotes the effect of crossover. The factor c_{ij} is given by

$$c_{ij} = (1 - \chi) \frac{\delta(i) + \delta(j)}{2} + \chi c_{ij}(\chi = 1), \quad (19)$$

where χ is a crossover rate, and $c_{ij}(\chi = 1)$ is the value of c_{ij} with $\chi = 1$. The discrete δ function is defined for integer m

$$\delta(m) = \begin{cases} 1 & (m = 0) \\ 0 & (m \neq 0). \end{cases}$$

For one-point crossover, we obtain [5]

– $i = 0$ or $j = 0$

$$c_{ij}(\chi = 1) = \frac{1}{2(\ell - 1)} \{ \text{lo}(i) - \text{hi}(i) + \text{lo}(j) - \text{hi}(j) \}. \quad (20)$$

The functions $\text{hi}(i)$ and $\text{lo}(i)$ are defined as

$$\begin{aligned} \text{hi}(i) &= \begin{cases} 1 & (i = 0) \\ i_{\max} & (\text{otherwise}), \end{cases} \\ \text{lo}(i) &= \begin{cases} \ell & (i = 0) \\ i_{\min} & (\text{otherwise}), \end{cases} \end{aligned}$$

where $i_{\max}$ and $i_{\min}$ stand for the leftmost and rightmost ones in i , respectively.

– $i \neq 0$ and $j \neq 0$

$$c_{ij}(\chi = 1) = \begin{cases} (i_{\min} - j_{\max})/2(\ell - 1) & (i_{\min} > j_{\max}) \\ (j_{\min} - i_{\max})/2(\ell - 1) & (j_{\min} > i_{\max}) \\ 0 & (\text{otherwise}). \end{cases} \quad (21)$$

For uniform crossover, we have a very simple expression

$$c_{ij} = (1 - \chi) \frac{\delta(i) + \delta(j)}{2} + \chi \prod_{m=1}^{\ell} \frac{\delta\{i(m)\} + \delta\{j(m)\}}{2}. \quad (22)$$

3 Schema Theorem

After introducing schemata and Holland's schema theorem [2], we describe the new schema theorem obtained by Walsh transformation [7].

A schema $\mathcal{H}$ represents a set of genotypes. It is given by three types of symbols, 0, 1 and *. The bits 0 and 1 are defining bits, and * is a wild card that allows both 0 and 1. The order of schema $\mathcal{O}(\mathcal{H})$ is the number of defining bits, and the defining length $\mathcal{L}(\mathcal{H})$ is the distance between the rightmost and leftmost defining bits.

The most famous example of the schema theorem is Holland's one with one-point crossover [2].

$$h(\mathcal{H}, t+1) \geq h(\mathcal{H}, t) \frac{f(\mathcal{H})}{\bar{f}(t)} \left\{ 1 - \chi \frac{\mathcal{L}(\mathcal{H})}{\ell-1} - p \mathcal{O}(\mathcal{H}) \right\}, \quad (23)$$

where $h(\mathcal{H}, t)$ is the relative frequency of the schema $\mathcal{H}$ at generation t , and $f(\mathcal{H})$ is the average fitness of genotypes included in $\mathcal{H}$.

We also use the notation showing explicitly the order of schema, the positions of defining bits, and their bit values,

$$\mathcal{H} = \mathcal{H}^{(k)}[i(b_1), i(b_2), \dots, i(b_k)],$$

Here, $k = \mathcal{O}(\mathcal{H})$, and $1 \leq b_1 < b_2 < \dots < b_k \leq \ell$ are positions of defining bits. In a similar manner, we use the notation for the relative frequency $h(\mathcal{H})$,

$$h(\mathcal{H}) = h^{(k)}[i(b_1), i(b_2), \dots, i(b_k)].$$

Then we derive the new schema theorem by using the Walsh transformation. When i and j take binary values, 0 and 1, the condition of $i = j$ is given by

$$\delta(i - j) = \frac{1}{2} \{ 1 + (-1)^i (-1)^j \}.$$

From this, we can obtain an expression for the frequency $h(\mathcal{H})$ of the first order schema $\mathcal{H}^{(1)}[i(b_1)]$. Using the normalization condition (2) and definition (7), we have

$$\begin{aligned} h^{(1)}[i(b_1)] &= \sum_{j=0}^{n-1} \delta(i(b_1) - j(b_1)) x_j \\ &= \sum_{j=0}^{n-1} \frac{1}{2} \{ x_j + (-1)^{i(b_1)} (-1)^{j(b_1)} x_j \} \\ &= \frac{1}{2} \{ 1 + (-1)^{i(b_1)} \tilde{x}^{(1)}[b_1] \}. \end{aligned}$$

For the L th order schema, noting

$$h^{(L)}[i(b_1), i(b_2), \dots, i(b_L)] = \sum_{j=0}^{n-1} \prod_{m=1}^L \delta(i(b_m) - j(b_m)) x_j,$$

and expanding the products of the delta functions, we have a new expression for the schema frequency in terms of the Walsh coefficients. Giving the positions of all defining bits $S = \{b_1, \dots, b_L\}$, and its subset $S' = \{b'_1, \dots, b'_k\}$, we have

$$\begin{aligned}
& h^{(L)}[i(b_1), \dots, i(b_L)] \\
&= \frac{1}{2^L} \sum_{S' \in \mathcal{P}(S)} (-1)^{i(b'_1) + \dots + i(b'_k)} \tilde{x}^{(k)}[b'_1, \dots, b'_k],
\end{aligned} \tag{24}$$

where $\mathcal{P}(S)$ is the power set of the set S . The power set is a set of all subsets of a given set. Thus the summation is taken over all subsets S' of S .

For example, the second order term is

$$\begin{aligned}
& h^{(L)}[i(b_1), i(b_2)] \\
&= \frac{1}{4} \{ 1 + (-1)^{i(b_1)} \tilde{x}^{(1)}[b_1] + (-1)^{i(b_2)} \tilde{x}^{(1)}[b_2] \\
&\quad + (-1)^{i(b_1) + i(b_2)} \tilde{x}^{(2)}[b_1, b_2] \}
\end{aligned}$$

The inverse transformation is given by

$$\begin{aligned}
& (-1)^{i(b_1) + \dots + i(b_L)} \tilde{x}^{(L)}[b_1, \dots, b_L] \\
&= \sum_{S' \in \mathcal{P}(S)} (-1)^{L-k} 2^k h^{(k)}[i(b'_1), \dots, i(b'_k)].
\end{aligned} \tag{25}$$

It becomes possible to derive the schema evolution equation for genetic operators. From the evolution equation of the Walsh coefficient under mutation (16), we obtain

$$\begin{aligned}
& \widehat{M} h^{(L)}[i(b_1), \dots, i(b_L)] \\
&= \sum_{S' \in \mathcal{P}(S)} (1 - 2p)^k p^{L-k} h^{(k)}[i(b'_1), \dots, i(b'_k)].
\end{aligned} \tag{26}$$

For the first order Walsh coefficients, we have

$$\widehat{M} h^{(1)}[i(b_1)] = (1 - 2p) h^{(1)}[i(b_1)] + p. \tag{27}$$

For crossover, we give here the final results of the schema theorem under uniform crossover. The process of derivation is described in [7]. We use an integer $i(\mathcal{H})$ as another representation of the schema $\mathcal{H}$

$$\{0, 1\} \rightarrow 1, \quad \{*\} \rightarrow 0 \tag{28}$$

For example, the new representation of $\mathcal{H} = *10*$ is $i(\mathcal{H}) = \langle 0, 1, 1, 0 \rangle$. Though this representation does not distinguish between 0 and 1, it may not bring any confusion in the case of crossover process.

The schema theorem for crossover can be given by

$$\widehat{C} h(k) = \sum_{i=0}^{n-1} c_{i, i \oplus k} h(i) h(i \oplus k), \tag{29}$$

The integers i and k in $h(k)$ and $h(i)$ are the binary expression of schema (28), and please consider that

$$i = i(\mathcal{H}), \quad k = k(\mathcal{H}).$$

The coefficient $c_{i,i \oplus k}$ is zero when i and $i \oplus k$ both take the value of one at the same bit position.

We will use a shorthand notation

$$h^{(1)}[i_k = 1] \rightarrow h(1_k), \quad h^{(1)}[i_k = 0] \rightarrow h(0_k).$$

In this analysis, the notion of linkage [14] is very important, and the linkage disequilibrium coefficient D is defined as

$$D[k, m] = h^{(2)}[1_k, 1_m] - h[1_k]h[1_m]. \quad (30)$$

This quantity shows the correlation between alleles at different loci. When each gene evolves independently, a population is in linkage equilibrium, while if there are any correlations among genes at different loci, it is in linkage disequilibrium. When the population is in linkage equilibrium, all D coefficients are zero, $D[k, m] = 0$.

If we set the crossover rate $\chi = 1$, the effect of crossover is given by [8]. For uniform crossover,

$$\hat{C} D[k, m] = \frac{1}{2} D[k, m]. \quad (31)$$

We note that crossover has an effect of reducing the magnitude of D . The action of mutation on D is given by

$$\hat{M} D[k, m] = (1 - 2p)^2 D[k, m], \quad (32)$$

and we see that mutation also reduces the magnitude of D .

4 Multiplicative Landscape

This section gives some mathematical results for GAs on the multiplicative landscape obtained by the schema theorem. The fitness function of multiplicative form is defined as

$$f_i = (1 - s)^{|z|} = \prod_{k=1}^{\ell} (1 - s)^{i^{(k)}}, \quad (0 < s < 1), \quad (33)$$

where s is a parameter of the strength of selection. We consider a maximization problem on this landscape. In [12], we have shown that the evolution equation of genotypes can be solved exactly for GAs under the action of selection with the multiplicative fitness and mutation. We will obtain here an exact schema evolution.

4.1 Schema Analysis

We assume that the population is in linkage equilibrium state at generation t , and it will later become clear that this is an essential assumption in the following analysis. Thus we have

$$x_i = \prod_{m=1}^{\ell} h\{i(m)\}, \quad (34)$$

and the average fitness is

$$\bar{f}(t) = \bar{f}^{(eq)},$$

where the average fitness in equilibrium is also given in the product form

$$\bar{f}^{(eq)} = \prod_{m=1}^{\ell} \{h(0_m) + (1-s)h(1_m)\} = \prod_{m=1}^{\ell} \{1 - s h(1_m)\}. \quad (35)$$

Then we may define the fitness function at each bit as

$$g_k = 1 - s h(1_k).$$

The Walsh transform of the fitness function is obtained as

$$\tilde{f}_i = \prod_{m=1}^{\ell} \{1 + (-1)^{i(m)} (1-s)\} \quad (36)$$

In the next subsection, without assuming linkage equilibrium, we will obtain the exact expression for the average fitness using this equation.

Under this assumption, we can obtain the schema equation of the first order schemata for selection. Substituting equation (36) into evolution equation for selection (11), we have

$$\hat{S}\tilde{x}_i(t) = \prod_{m=1}^{\ell} \frac{\beta\{i(m)\}}{\beta(0_m)}. \quad (37)$$

Here, the function $\beta\{i(m)\}$ is defined as

$$\beta\{i(m)\} = h(0_m) + (-1)^{i(m)} (1-s)h(1_m).$$

Then we can derive the schema equation of the L th order schemata. With the set of defining bits $S = \{b_1, \dots, b_L\}$, we have

$$\hat{S}h^{(L)}[i(b_1), \dots, i(b_L)] = \prod_{m \in S} \frac{a\{i(m)\}h\{i(m)\}}{h(0_m) + (1-s)h(1_m)}, \quad (38)$$

where $a(0) = 1$ and $a(1) = 1 - s$. For $h(1_k)$, the effect of selection is

$$\hat{S}h(1_k) = \frac{(1-s)h(1_k)}{1 - sh(1_k)}.$$

It is important to note that the schemata after selection also satisfy the condition of linkage equilibrium. From equations (26) and (29), we can also verify that the schemata after mutation and crossover satisfy the condition of equilibrium. Thus, if the population is in linkage equilibrium at $t = 0$, then it always satisfies the condition of equilibrium.

The change in the fitness function g_k under the action of selection is

$$\Delta g_k(t) \equiv \hat{S}g_k(t) - g_k(t),$$

and we have

$$\Delta g_k(t) = \frac{v_k(t)}{g_k(t)}, \quad (39)$$

where the variance v_k is

$$v_k = s^2 h(1_k)\{1 - h(1_k)\}.$$

4.2 Schema Representation of Average Fitness

We will discuss here the direct effects of crossover and mutation on the average fitness. By using equations (13) and (25), the average fitness $\bar{f}(t)$ can be given in terms of schema frequencies. After some calculations, we have

$$\bar{f}(t) = \bar{f}^{(eq)} + s^2 \Delta^{(2)} - s^3 \Delta^{(3)} + \dots + (-s)^\ell \Delta^{(\ell)}. \quad (40)$$

The contribution of the m th order term is given by

$$\Delta^{(m)} = \sum_{k_1} \sum_{< k_2} \dots \sum_{< k_m} D[k_1, k_2, \dots, k_m],$$

where the linkage equilibrium coefficient of the order m is given by the schema frequencies

$$D[k_1, k_2, \dots, k_m] = h^{(m)}(1_{k_1}, 1_{k_2}, \dots, 1_{k_m}) - h(1_{k_1})h(1_{k_2}) \dots h(1_{k_m}). \quad (41)$$

When we use a small value of s , the equation is approximately given by

$$\bar{f}(t) \approx \bar{f}^{(eq)} + s^2 \sum_{k_1} \sum_{< k_2} D[k_1, k_2]. \quad (42)$$

We have shown in the paper [15] that the variance V of Hamming distance from the optimum solution can be given by the sum of additive variance V_a and epistatic variance V_e

$$V = V_a + V_e, \quad (43)$$

where

$$V_a = \sum_{k=1}^{\ell} h(1_k) \{1 - h(1_k)\},$$

and

$$V_e = 2 \sum_{k_1} \sum_{< k_2} D[k_1, k_2].$$

Though we use the term “variance”, the epistatic variance V_e can take negative values.

If we use a small value of s , we obtain the variance of the fitness $V(f)$

$$V(f) = s^2(V_a + V_e) + s^3(\dots) + \dots \approx s^2(V_a + V_e). \quad (44)$$

The variance $V(f)$ in the linkage equilibrium is given by

$$V^{(eq)} = \bar{f}^{(eq)}|_{s \rightarrow 2s-s^2} - \bar{f}^{(eq)} \cdot \bar{f}^{(eq)}. \quad (45)$$

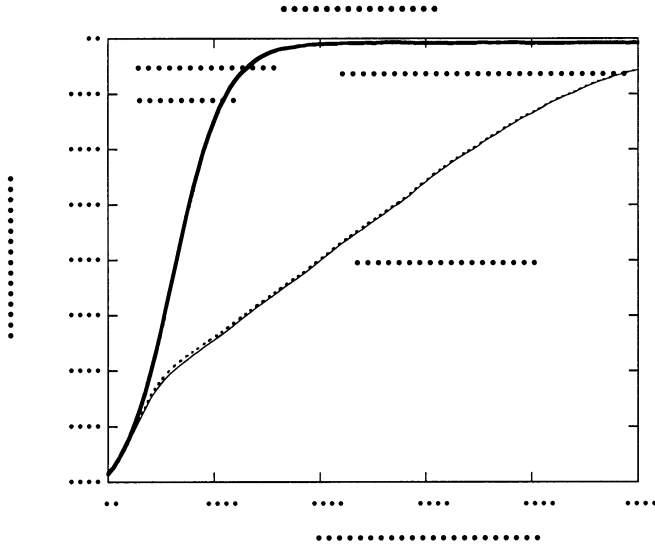


Fig. 1. Numerical experiments for $\bar{f}(t)$. Thick lines for the experiments with crossover, and thin lines without crossover. Solid lines for $\bar{f}(t)$, and dotted lines for $\bar{f}^{(eq)}$. The mutation rate $p = 0.001$, selection parameter $s = 0.2$, and $N = 200$.

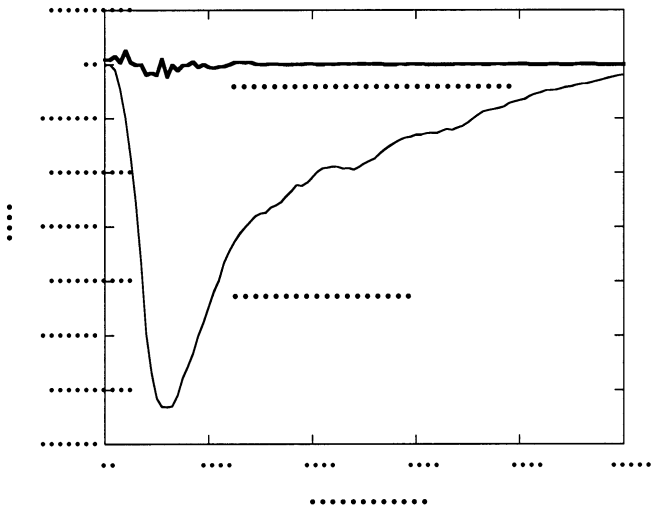


Fig. 2. The sum of the second order linkage disequilibrium coefficients, $s^2 V_e$, with ($\chi = 1$) and without ($\chi = 0$) crossover.

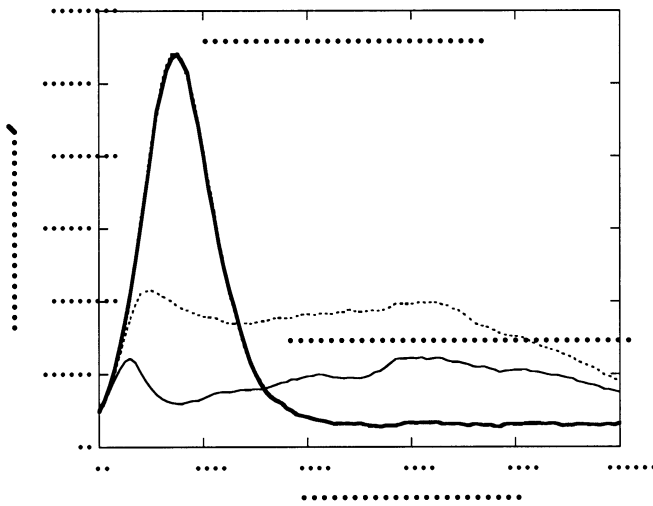


Fig. 3. Variance of fitness. $V(f)$ and $V^{(eq)}$ with ($\chi = 1$) and without ($\chi = 0$) crossover. Solid lines and dotted lines show $V(f)$ and $V^{(eq)}$, respectively.

5 Experiments

We carried out numerical experiments of GA on the multiplicative landscape, and their results were compared with the theory in the previous section. The population size N was 200, and bit length was $\ell = 8$. The selection parameter was $s = 0.2$. We used the mutation rates p of 0.001, and uniform crossover. The effect of crossover was studied by comparing the experiments with the crossover rates $\chi = 0$ and 1. At $t = 0$, we used the initial value of $h(1_k) = 1/8$. The calculations were performed repeatedly with the same parameters, and their results were averaged over 100 runs.

Figure 1 demonstrates the average fitness obtained in four types of calculations. Thick solid line shows $\bar{f}(t)$ with crossover $\chi = 1$, and thin solid line without crossover $\chi = 0$. Thick dotted line is for $\bar{f}^{(eq)}$ with crossover, and thin dotted line without crossover. We cannot observe the thick dotted line because it is masked by the thick solid line. The effect of crossover is very significant in these experiments. We see that crossover works as a beneficial operator. The effect of linkage disequilibrium is very small, and $\bar{f}(t) \approx \bar{f}^{(eq)}$ in experiments with and without crossover. This result seems to suggest that the effect of linkage disequilibrium is negligibly small. However, this is not the case.

Figure 2 gives s^2V_e with and without crossover. V_e shows the degree of linkage disequilibrium of the second order. When crossover is active, V_e is almost 0. On the other hand, if $\chi = 0$, V_e takes negative values of large magnitude.

In Fig. 3, the variance of fitness $V(f)$ is shown. The variances in linkage equilibrium $V^{(eq)}$ calculated by equation (45) are given in dotted lines. There is

large difference in the results with (thick lines) and without (thin lines) crossover. The GA with crossover demonstrates the large value of $V(f)$, and the dotted line for $V^{(eq)}$ is masked by the solid line for $V(f)$. When crossover is absent, the effect of linkage disequilibrium becomes large, and as a result $V(f)$ is small. The contribution of linkage is negative, and we observe large difference between $V(f)$ and $V^{(eq)}$. Fisher's theorem states that the large variance of fitness can give good performance [15], and this explains the fast increase of $\bar{f}(t)$ in GA with crossover in Fig.1.

6 Conclusion

In this paper, we have shown that the exact schema theorem can be obtained for the GA on the multiplicative landscape. In this problem, if we ignore the effect of random sampling, the assumption of linkage equilibrium holds at all generations when the initial state is in linkage equilibrium. Therefore, the system is completely determined by the first order schemata $h(1_k)$. However, in the practical calculations, there appears the stochastic effect, and we have to take into account the effect of linkage disequilibrium.

Crossover works as a beneficial operator in this problem. It reduces the magnitude of D coefficients, and as a result accelerates the speed of evolution by increasing the total variance $V(f)$. This is an indirect effect of crossover.

On the other hand, there is a direct effect of linkage given by equation (40). However, this effect is very small as shown in Fig.1 in our experiments.

References

1. H. Furutani: Schema Analysis of OneMax Problem – Evolution Equation for First Order Schemata –, Foundations of Genetic Algorithms 7, pp.19–36, Morgan Kaufmann (2003).
2. J.H. Holland: Adaptation in Natural and Artificial Systems, MIT press (1992).
3. C. R. Stephens and H. Waelbroeck: "Effective Degrees of Freedom in Genetic Algorithms," *Physical Review E*, **57** (1998) 3251–3264.
4. C. R. Stephens and H. Waelbroeck: "Schemata Evolution and Building Blocks," *Evolutionary Computation*, **7** (1999) 109–124.
5. M.D. Vose: The Simple Genetic Algorithms, MIT Press (1999).
6. A. H. Wright: "The Exact Schema Theorem," Technical report, University of Montana, Missoula, MT59812 (1999).
7. H. Furutani: "Derivation of Schema Theorem for Mutation and Crossover by Walsh Transformation," *IPSIJ Journal*, **43** (2002) 1050–1060.
8. H. Furutani: "Walsh Analysis of Crossover in Genetic Algorithms," *IPSIJ Journal*, **42** (2001) 2270–2283.
9. G. Woodcock and P.G. Higgs: "Population Evolution on a Multiplicative Single-Peak Fitness Landscape," *Journal of Theoretical Biology*, **179** (1996) 61–73.
10. A. Prügel-Bennett: "Modelling Evolving Populations," *Journal of Theoretical Biology*, **185** (1997) 81–95.
11. J. Rowe: "A Normal Space of Genetic Operators," *Evolutionary Computation*, **9** (2001) 25–42.

12. H. Furutani: "Study of Evolution in Genetic Algorithms by Eigen's Theory Including Crossover Operator," Proceedings of the Simulated Evolution and Learning Conference, SEAL'00, (2000) 2696–2703.
13. R.A. Fisher: The Genetical Theory of Natural Selection, Clarendon Press (1930). Reprint: Dover Publications (1959).
14. J. Maynard Smith: Evolutionary Genetics, Oxford University Press (1998).
15. H. Furutani: "Study of Crossover in One Max Problem by Linkage Analysis," L. Spector et al. (Eds.) Proceedings of the Genetic and Evolutionary Computation Conference, GECCO-2001 Morgan Kaufmann, (2001) 320–327.

On the Treewidth of NK Landscapes

Yong Gao and Joseph Culberson

Department of Computing Science, University of Alberta
Edmonton, Alberta, Canada, T6G 2E8
{ygao, joe}@cs.ualberta.ca

Abstract. The concepts of treewidth and tree-decomposition on graphs generalize those of the trees. It is well established that when restricted to instances with a bounded treewidth, many NP hard problems can be solved polynomially. In this paper, we study the treewidth of the NK landscape models. We show that the NK landscape model with adjacent neighborhoods has a constant treewidth, and prove that for $k \geq 2$, the treewidth of the NK landscape model with random neighborhoods asymptotically grows with the problem size n .

1 Introduction

NK landscapes have been widely used in the study of genetic algorithms and computational biology [1]. There are basically two classes of NK landscape models: the NK landscape model with adjacent neighborhood and the NK landscape model with random neighborhood. Both of the two models have been analyzed and characterized from the perspectives of statistics and computational complexity [2,3,4,5].

In [2], it was shown that even though the NK landscape model with adjacent neighborhoods can be solved polynomially and the NK landscape model with the random neighborhood is usually NP complete, the two classes of NK models share almost identical statistical characteristics such as the average number of local minima and the average height of the local minima. This has puzzled researchers in this field for a while. In [4], it was shown that the decision versions of NK landscapes with random neighborhoods are easy to solve with probability asymptotic to 1 under two commonly used probabilistic settings. This is more or less in contrast to the common observation that NK landscapes with random neighborhoods are usually hard for genetic algorithms.

In the study of constraint satisfiability problems and algorithmic graphs [6,7], it is well-known that problems with an underlying tree structure can be solved linearly. The concepts of treewidth and tree-decomposition of graphs generalize the concept of trees and measure the degree to which a graph behaves like a tree [7]. It is well established that many NP complete problems, when restricted to instances with a bounded treewidth structure, can be solved polynomially via dynamic programming or some other deterministic algorithms [8].

In this paper, we study the treewidth of NK landscapes in an effort to further understand the differences between the two classes of NK models and the reasons why they appear to be hard for genetic algorithms.

In Section 2, we introduce the NK landscape model, its interaction graph, and the concepts related to the treewidth and tree decomposition of graphs. In Section 3, we study the treewidth of the NK landscape models and prove that the adjacent neighborhood NK landscape model has a fixed treewidth and the treewidth of the random neighborhood NK landscape model grows linearly with the problem size n . In Section 4, we discuss the implications of our results and future work.

2 NK Landscape Models

An NK landscape

$$f(x) = \sum_{i=1}^n f_i(x_i, \Pi(x_i)), \quad (1)$$

is a real-valued function defined on binary strings of fixed length, where $n > 0$ is a positive integer and $x = (x_1, \dots, x_n) \in \{0, 1\}^n$. It is the sum of n *local fitness functions* f_i , $1 \leq i \leq n$. Each local fitness function $f_i(x_i, \Pi(x_i))$ depends on the *main variable* x_i and its neighborhood

$$\Pi(x_i) \subset \mathcal{P}_k(\{x_1, \dots, x_n\} \setminus \{x_i\}) \quad (2)$$

where $\mathcal{P}_k(X)$ denotes the set of all subsets of size k from X . The most important parameters of an NK landscape are the number of variables n , and the size of the neighborhood $k = |\Pi(x_i)|$.

In an NK landscape, the neighborhood $\Pi(x_i)$ can be chosen in two ways: the *random neighborhood*, where the k variables are randomly chosen from the set $\{x_1, \dots, x_n\} \setminus \{x_i\}$, and the *adjacent neighborhood*, where k variables with indices nearest to i (modulo n) are chosen. To simplify the discussion, we assume in this paper that the adjacent neighborhoods are defined as follows: for each i ,

$$\Pi(x_i) = (x_{\max(0, i - \lceil \frac{k}{2} \rceil)}, \dots, x_{i-1}, x_{i+1}, \dots, x_{\min(n, i + \lceil \frac{k}{2} \rceil)}). \quad (3)$$

We use $A(n, k)$ to represent the NK landscape model with adjacent neighborhood and $N(n, k)$ to represent the NK landscape model with the random neighborhood.

Definition 1. *The interaction graph of an NK landscape model is a graph $G(V, E)$ where the vertex set $V = \{x_1, \dots, x_n\}$ corresponds to the set of variables in the NK landscape and $(x_i, x_j) \in E$ if and only if x_i and x_j both appear in a local fitness function.*

The interaction graph of an NK landscape model captures all the interactions among the variables in the NK landscapes. A knowledge of these interactions is critical in understanding the complexity and designing appropriate algorithms to solve the problems. For example, if the interaction graph is a tree, then a linear time algorithm readily exists to solve the problem. As yet another example, if the underlying graph can be decomposed into several connected components, then

a viable approach to the problem is to first solve the subproblems represented by each connected component and then combine the partial solutions together.

The concept of treewidth and tree decomposition generalizes the above ideas further. Let us start with the definition of the l -tree.

Definition 2. ([7]) *l -Trees are defined recursively as follows:*

1. *A clique with $l+1$ vertices is an l -tree;*
2. *Given an l -tree T_n with n vertices, an l -tree with $n+1$ vertices is constructed by adding to T_n a new vertex which is made adjacent to an l -clique of T_n and non-adjacent to the rest of the vertices.*

Definition 3. ([7]) *A graph is called a partial l -tree if it is a subgraph of an l -tree. The treewidth of a graph G is the minimum value l for which G is a partial l -tree.*

The treewidth of a graph has an equivalent definition based on the concept of tree decomposition.

Definition 4. ([7]) *A tree decomposition of a graph $G = (V, E)$ is a pair $D = (S, T)$ where $S = \{X_i, i \in I\}$ is a collection of subsets of vertices of G and $T = (I, F)$ is a tree with one node for each subset of S , such that*

1. $\bigcup_{i \in I} X_i = V$,
2. *for all the edges $(v, w) \in E$ there exists a subset $X_i \in S$ such that both v and w are in X_i , and*
3. *for each vertex v , the set of nodes $\{i, v \in X_i\}$ forms a subtree of T .*

The width of the tree decomposition $D = (S, T)$ is $\max_{i \in I} (|X_i| - 1)$. And the treewidth of a graph is the minimum width over all tree decompositions of the graph.

3 The Treewidth of the NK Landscape Models

In this section the treewidth of the NK landscapes models is studied. We start with the treewidth of the NK landscape model with adjacent neighborhoods. In [2], it has been shown that the NK landscape model with adjacent neighborhoods can be solved by dynamic programming in linear time. The following theorem shows that the interaction graph of the NK landscape model with adjacent neighborhoods has a treewidth independent of n .

Theorem 1. *Let $A(n, k)$ be the NK landscape model with adjacent neighborhoods with the underlying graph G . Then, the treewidth of G is at most $2k$.*

Proof. By direct construction, we can get a tree decomposition with a width k if the cyclic interactions at the boundaries are ignored. When taking into account the cyclic interactions at the boundaries, we can get a tree decomposition with a width $2k$. $\square$

We now turn to the NK landscape model with random neighborhoods. Since the problem is in general NP hard, we do not expect its interaction graph to have a bounded treewidth because that will mean the problem is polynomially solvable. Instead, we are interested in how the treewidth changes as the problem size n and the interaction size k increase, and the probability with which the treewidth remains small enough for algorithms making use of treewidth-related information to work efficiently. Our result below, however, shows that the treewidth asymptotically grows linearly with n .

Definition 5. ([7]) Let $G(V, E)$ be a graph with $|V| = n$. A partition (S, A, B) of V is a balanced l -partition if the following conditions are satisfied:

1. $|S| = l + 1$;
2. $\frac{1}{3}(n - l - 1) \leq |A|, |B| \leq \frac{2}{3}(n - l - 1)$; and
3. S separates A and B , i.e., there are no edges between vertices of A and vertices of B .

Theorem 2. Let $w(n, k)$ be the treewidth of the interaction graph of the NK landscape model with random neighborhoods. Then, for $k \geq 2$, there is a fixed constant $\delta > 0$ such that

$$\lim_n Pr\{w(n, k) \leq \delta n\} = 0. \quad (4)$$

Proof. Let $l = w(n, k)$. It is well-known that if a graph has a treewidth l , then the graph must have a balanced l -partition [7]. Consider the interaction graph $G = G(V, E)$ of the NK landscape with random neighborhoods. Let $\mathcal{P}$ be the set of all the partitions of the vertex set V that satisfies the first two conditions in Definition 5. For a given $P = (S, A, B) \in \mathcal{P}$, define a random variable I_P as follows:

$$I_P = \begin{cases} 1, & \text{if } P \text{ is a balanced partition;} \\ 0, & \text{otherwise.} \end{cases} \quad (5)$$

and let O be the event that I_P is 1, i.e., that there are no edges between vertices of A and vertices of B . Recall that $\Pi(x_i)$ is the set of neighbors of the i -th local fitness function. For each $1 \leq i \leq n$ with $x_i \in A$ (or $x_i \in B$), let O_i be the event that $\Pi(x_i) \subset A \cup S$ ($\Pi(x_i) \subset B \cup S$ respectively). For $x_i \in S$, let O_i be the event that $\Pi(x_i) \subset A \cup S$ or $\Pi(x_i) \subset B \cup S$. Then, by the definition of the NK landscape model with random neighborhoods and its interaction graph, we have

$$O = \bigcap_{1 \leq i \leq n} O_i. \quad (6)$$

Since each local fitness function selects its neighbors independently, $O_1, \dots, O_n$ are mutually independent. We have

$$Pr\{O\} = \prod_{i=1}^n Pr\{O_i\}. \quad (7)$$

For $x_i \in A$ (or $x_i \in B$), we have

$$Pr\{O_i\} \leq \frac{\binom{\frac{2}{3}(n-l-1)+l}{k}}{\binom{n-1}{k}} = \left(\frac{1}{3}\right)^k \left(2 + \frac{l}{n-1}\right)^k. \quad (8)$$

Similarly, for $x_i \in S$, we have

$$Pr\{O_i\} \leq 2\left(\frac{1}{3}\right)^k \left(2 + \frac{l}{n-1}\right)^k. \quad (9)$$

Then,

$$Pr\{O\} \leq 2^{l+1} \left(\frac{1}{3}\right)^{kn} \left(2 + \frac{l}{n-1}\right)^{kn}. \quad (10)$$

Let $I = \sum_{P \in \mathcal{P}} I_P$. By its definition, we have

$$\begin{aligned} |\mathcal{P}| &= \binom{n}{l+1} \sum_{\frac{1}{3}(n-l-1) \leq a \leq \frac{2}{3}(n-l-1)} \binom{n-l-1}{a} \\ &\leq \binom{n}{l+1} 2^{n-l-1}. \end{aligned} \quad (11)$$

It follows that the expectation of I satisfies

$$\begin{aligned} \mathcal{E}\{I\} &= \sum_{P \in \mathcal{P}} \mathcal{E}\{I_P\} \\ &\leq \binom{n}{l+1} 2^{n-l-1} 2^{l+1} \left(\frac{1}{3}\right)^{kn} \left(2 + \frac{l}{n-1}\right)^{kn} \\ &\leq \binom{n}{l+1} 2^n \left(\frac{2}{3} + \frac{1}{3} \frac{l}{n-1}\right)^{kn}. \end{aligned} \quad (12)$$

Let $0 < y = \frac{l+1}{n} < 1$. We obtain from Stirling's formula that

$$\binom{n}{l+1} \sim \frac{1}{\sqrt{2\pi y(1-y)n}} \left(\frac{1}{y^y(1-y)^{1-y}} \right)^n. \quad (13)$$

And hence,

$$\mathcal{E}\{I\} \leq \frac{1}{\sqrt{2\pi y(1-y)n}} \left(\frac{2}{y^y(1-y)^{1-y}} \cdot \left(\frac{2}{3} + \frac{1}{3}y\right)^k \right)^n. \quad (14)$$

Since for $k \geq 2$,

$$\lim_{y \rightarrow 0} \left(\frac{2}{y^y(1-y)^{1-y}} \cdot \left(\frac{2}{3} + \frac{1}{3}y\right)^k \right) = 2\left(\frac{2}{3}\right)^k < 1, \quad (15)$$

there exists a $0 < \delta < 1$ such that

$$\lim_n \frac{1}{\sqrt{2\pi\delta(1-\delta)n}} \left(\frac{2}{\delta^\delta(1-\delta)^{1-\delta}} \cdot \left(\frac{2}{3} + \frac{1}{3}\delta\right)^k \right)^n = 0. \quad (16)$$

Therefore, we have

$$\begin{aligned} \lim_n \Pr\{w(n, k) \leq \delta n\} &\leq \lim_n \Pr\{I > 0\} \\ &\leq \lim_n \mathcal{E}[I] = 0. \end{aligned} \quad (17)$$

This concludes the proof. $\square$

4 Conclusions and Future Work

As we have shown in the previous sections, the treewidth of the NK landscape is bounded by the interaction index for the adjacent neighborhood model, but grows linearly with the problem size for the random neighborhood model. In addition to the NP complete study of the random neighborhood model, our result is the first one that depicts the difference between the two statistically similar NK landscape models.

It is well-known that optimization problems with bounded treewidth can be decomposed into independent sub-problems and solved polynomially using dynamic programming techniques. This is the case for the NK landscapes with adjacent neighborhoods [2]. Other examples include the constraint satisfaction problems and the inference problem for Bayesian networks [6,9] in which the popular tree-clustering method run polynomially if the problems under consideration have a bounded treewidth. For the random neighborhood model, our result shows that algorithms that make use of the information about the structures of the interactions in the same way as the tree-clustering approach cannot solve the problem efficiently.

An interesting question that deserves further investigation is “Do genetic algorithms exploit the treewidth-related structural information? And if so, to what extent do they rely on that information to work?” We suspect that the answer to the first question is affirmative. In fact, this is best illustrated by the recent work on sampling-based genetic algorithms. Instead of using genetic operators to generate new solutions, these sampling-based algorithms generate candidate solutions by sampling some probability distributions on the solution space and update the distributions based on the information gathered as new solutions are evaluated. The probability distributions may be modelled as the product of independent distributions [10], decomposable distributions naturally obtained from the knowledge about the interaction structures[11], or Bayesian networks that are constructed from the existing candidate solutions[12]. All of these models depend on the factorization of a multivariate probability distribution into the form

$$p(x_1, \dots, x_n) = \frac{1}{Z} \prod_{C \in \mathcal{C}} \psi_C(x_C),$$

where $p(\cdot)$ is the original distribution and $\mathcal{C}$ is a tree decomposition of a graph, the structure of which is defined (explicitly or implicitly) by the designers of the sampling-based algorithms and is believed to be able to capture the interaction

structure of the original optimization problems. The effectiveness and efficiency of these sampling-based algorithms thus depend critically on how well the factorization approximates the real tree-decomposition of the original problem, and on the width of the tree decomposition which is lower bounded by the treewidth of the original problem.

Another direction of future work is to study the treewidth of the NK landscape models by considering the number of local fitness functions as a parameter as well as the interaction index k . By relaxing the requirement that each variable is associated with a local fitness function in the current model, we can consider a more generalized model in which the number of variables that have an associated local fitness function is another tunable parameter. It would be interesting to study the treewidth of such a generalized NK landscape model.

References

1. Kauffman, S.: *The Origins of Order: Self-organization and Selection in Evolution*. Oxford University Press, Inc. (1993)
2. Weinberger, E.D.: NP completeness of Kauffman's NK model, a tunable rugged fitness landscape. Technical Report Working Papers 96-02-003, Santa Fe Institute, Santa Fe (1996)
3. Wright, A.H., Thompson, R.K., Zhang, J.: The computational complexity of NK fitness functions. Technical report, Department of Computer Science, University of Montana (1999)
4. Gao, Y., Culberson, J.: An analysis of phase transition in NK landscapes. *Journal of Artificial Intelligence Research* **17** (2002) 309–332
5. Evans, S.N., Steinsaltz, D.: Estimating some features of NK fitness landscapes. *Ann. Appl. Probab.* **12** (2002) 1299–1321
6. Dechter, R., Fattah, Y.: Topological parameters for time-space tradeoff. *Artificial Intelligence* **125** (2001) 93–118
7. Kloks, T.: *Treewidth: Computations and Approximations*. Springer-Verlag (1994)
8. Downey, R.G., Fellows, M.R.: *Parameterized Complexity*. Springer (1997)
9. Gottlob, G., Leone, N., Scarcello, F.: A comparison of structural CSP decomposition methods. *Artificial Intelligence* **124** (2000) 243–282
10. Leung, Y., Gao, Y., Zhang, W.: A genetic-based method for training fuzzy systems. In: *Proc. of the 10th IEEE International Conference on Fuzzy Systems*. Volume 1., IEEE (2001) 123–126
11. Mühlenbein, H., Mahnig, T.: Convergence theory and applications of the factorized distribution algorithm. *Journal of Computing and Information Technology* **7** (1999) 19–32
12. Pelikan, M., Goldberg, D.E., Lobo, F.: A survey of optimization by building and using probabilistic models. Technical Report 99018, IlliGAL, University of Illinois (1999)

Selection Intensity in Asynchronous Cellular Evolutionary Algorithms

Mario Giacobini¹, Enrique Alba², and Marco Tomassini¹

¹ Computer Science Institute, University of Lausanne, Lausanne, Switzerland

² Department of Computer Science, University of Málaga, Málaga, Spain

Abstract. This paper presents a theoretical study of the selection pressure in asynchronous cellular evolutionary algorithms (cEAs). This work is motivated by the search for a general model for asynchronous update of the individuals in a cellular EA, and by the necessity of better accuracy beyond what existing models of selection intensity can provide. Therefore, we investigate the differences between the expected and actual values of the selection pressure induced by several asynchronous update policies, and formally characterize the update dynamics of each variant of the algorithm. New models for these two issues are proposed, and are shown to be more accurate (lower fit error) than previous ones.

1 Introduction

Cellular evolutionary algorithms (cEAs), also called diffusion or fine-grained models, have been popularized, among others, by early work of Gorges-Schleuter [4] and Manderick and Spiessen [6]. These models are based on a spatially distributed population in which genetic operations may only take place in a small neighborhood of each individual. Usually, individuals are arranged on a regular grid of dimensions $d = 1, 2$ or 3 . Cellular EAs are a kind of decentralized EA model. They are not just a parallel implementation of an EA; in fact, although parallelism could be used to speed up the search, we do not address it in this work.

Although fundamental theory is still an open research line for cEAs, they have been empirically reported as being useful in maintaining diversity and promoting slow diffusion of solutions through the grid (exploration). Part of their behavior is due to a lower selection pressure compared to that of panmictic EAs (here *panmictic* means that any one chromosome may mate with any other in the population). The influence of the neighborhood, grid topology, and high efficiency in comparison to other EAs have all been investigated in detail in [2, 5, 7, 8], and tested on different applications such as combinatorial and numerical optimization.

Cellular EAs can be seen as stochastic cellular automata (CAs) [11, 13] where the cardinality of the set of states is equal to the number of points in the search space. CAs, as well as cEAs, usually assume a *synchronous* or “parallel” update policy (reproduction at a time), in which all the cells are formally updated

simultaneously. However, this is not the only option available. Indeed, several works on *asynchronous* CAs have shown that sequential update policies have a marked effect on their dynamics (see e.g. [9,10]). Thus, it would be interesting to investigate asynchronous cEAs and their problem solving capabilities. A first step in that direction was made in [1], where a set of standard problems were studied under several asynchronous update policies in a 2-d cGA environment. The main observation was that, although asynchronous update is not always the best choice in terms of solution quality, it is numerically faster, and the speed of convergence can be varied by changing the updating scheme. Thus, since convergence and diversity in EAs are related to selection, we would like to get a better understanding of the behavior of selection in asynchronous cEAs as compared to synchronous update and to the panmictic case. We present here an extension of the works [2,5,7,8] on selection pressure to asynchronous cEAs. For reasons of space, we limit ourselves to the two-dimensional grid case, the most common in practice.

The paper is organized as follows. The next section contains some background on asynchronous cEAs. Section 3 describes the results of our experiments on selection pressure in asynchronous cEAs. Section 4 analyzes the current logistic model of selection pressure and presents an improved characterization of asynchronous algorithms, leading to a new model proposal. Finally, section 5 offers our conclusions, as well as some comments on future work.

2 Asynchronous cEAs

Updating a cell (individual) in a cellular EA means selecting two parents in the individual's neighborhood (including the individual itself), applying genetic operators to them, and finally replacing the individual with the best offspring. In a conventional synchronous cEA, all the individuals in the grid are updated simultaneously. This step makes up a generation, and the process is repeated until a termination condition is reached.

There exist many ways for sequentially updating the cells of a 2-d cEA. Here we employ *step-driven* updates and ignore the so-called *time-driven* methods, in which (real) time is explicit. Time-driven methods are more realistic for physical simulation but are not needed in the EA case (an excellent discussion of asynchronous update in CAs is available in [9]). The most general update scheme is independent random ordering of updates in time, which consists of randomly choosing the cell to be updated next with replacement. This corresponds to a binomial distribution for the update probability. This update policy will be called *uniform choice* (UC) in the following and it is similar to the time-driven Poisson update in the limit of large n , n being the population size.

In our study we also consider three other update methods: *fixed line sweep*, *fixed random sweep*, and *new random sweep* (we employ the same terminology as in [9]).

- In *fixed line sweep* (LS), the simplest method, the n grid cells are updated sequentially $(1, 2 \dots n)$, line by line of the 2-d grid.

- In the *fixed random sweep* update (FRS), the next cell to be updated is chosen with uniform probability without replacement; this will produce a certain update sequence $(c_1^j, c_2^k, \dots, c_n^m)$, where c_q^p means that cell number p is updated at time q and $(j, k, \dots, m)$ is a permutation of the n cells. The same permutation is then used for all update cycles.
- The *new random sweep* method (NRS) works like FRS, except that a new random cell permutation is chosen anew for each sweep through the array.

A *time step* is defined as updating n times sequentially, which corresponds to updating *all* the n cells in the grid for LS, FRS and NRS, and possibly less than n different cells in the uniform choice method, since some cells might be updated more than once. It should be noted that, with the exception of fixed line sweep, the other asynchronous updating policies are stochastic, representing an additional source of non-determinism besides that of the genetic operators.

3 Takeover Times

In order to study the induced selection pressure by itself (without introducing the perturbing effect of recombination or mutation operators) a standard technique is to let selection be the only active operator, and then monitor the growth rate of the best individual in the initial population [3]. The takeover time is the time it takes for the single best individual to conquer the whole population. A shorter takeover time thus means a higher selection pressure. It has been shown that when we move from a panmictic population to a spatially structured one of the same size with synchronous updating of the cells, the global selection pressure induced on the entire population is qualitatively similar but weaker (Sarma and De Jong [7]).

Three standard selection algorithms were used in [7], namely fitness proportionate, linear ranking, and binary tournament. The cellular EA structure was a two-dimensional toroidal grid of size 32×32 with three different neighborhood shapes with 5, 9 and 13 neighbors respectively, which are the most common in practice. In the spatially distributed case it was observed that, for all three mentioned neighborhoods, the global selection pressure induced by fitness-proportionate selection was smaller than the pressures induced by linear ranking and binary tournament, with binary tournament being roughly equivalent to ranking as the neighborhood size increases, as it can be inferred from well-known existing theoretical considerations on selection pressure.

Sarma and De Jong [8] performed a more detailed empirical analysis of the effects of the neighborhood's size and shape on the local selection algorithms. They were able to show that propagation times are closely related to the neighborhood size, with larger neighborhoods giving rise to stronger selection pressures.

In the following, we report results (for three different neighborhoods) on the selection pressure for three selection methods in the case of asynchronous update. The neighborhoods used are Von Neumann (5 neighbors along the NWSE directions and the center cell, also called Linear 5), Moore (9 neighbors, including the central cell and its eight nearest neighbors, also called Compact 9), and

Compact 13 (like Moore, with four neighbors added along the N,W,S, and E directions -like a diamond-).

For the sake of comparison, we also include the curves corresponding to the panmictic case and to the synchronously updated grid. Since the results with the asynchronous Fixed Random Sweep are very similar to those using the New Random Sweep policy, only the latter curves are reported. Each of these results is the average of 100 independent runs.

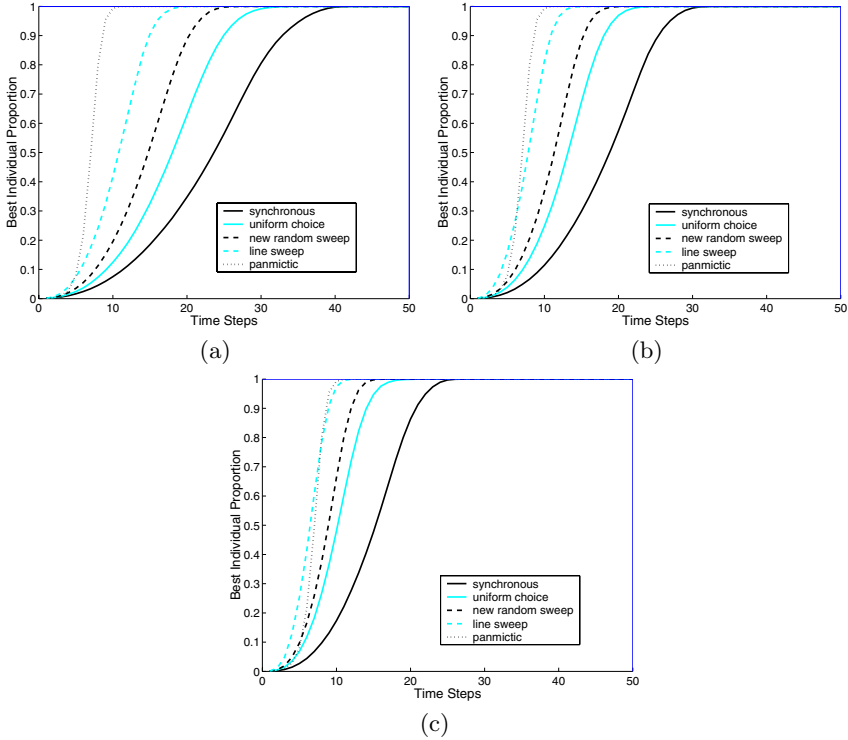


Fig. 1. Takeover times with rank selection. Linear 5 neighborhood (a); Compact 9 neighborhood (b); Compact 13 neighborhood (c). Mean values over 100 runs. The vertical axis represents the proportion of population consisting of best individual as a function of the time step.

Figure 1 shows the mean growth curves of a cEA using rank selection in a Linear 5, Compact 9 and Compact 13 neighborhood, respectively. Figure 2 depicts graphs of mean growth curves for a cEA using binary tournament selection respectively in the Linear 5, Compact 9 and Compact 13 neighborhood. The mean takeover times results for a cEA using a roulette wheel selection respectively in a Linear 5, Compact 9, and Compact 13 neighborhood are reported in Figure 3.

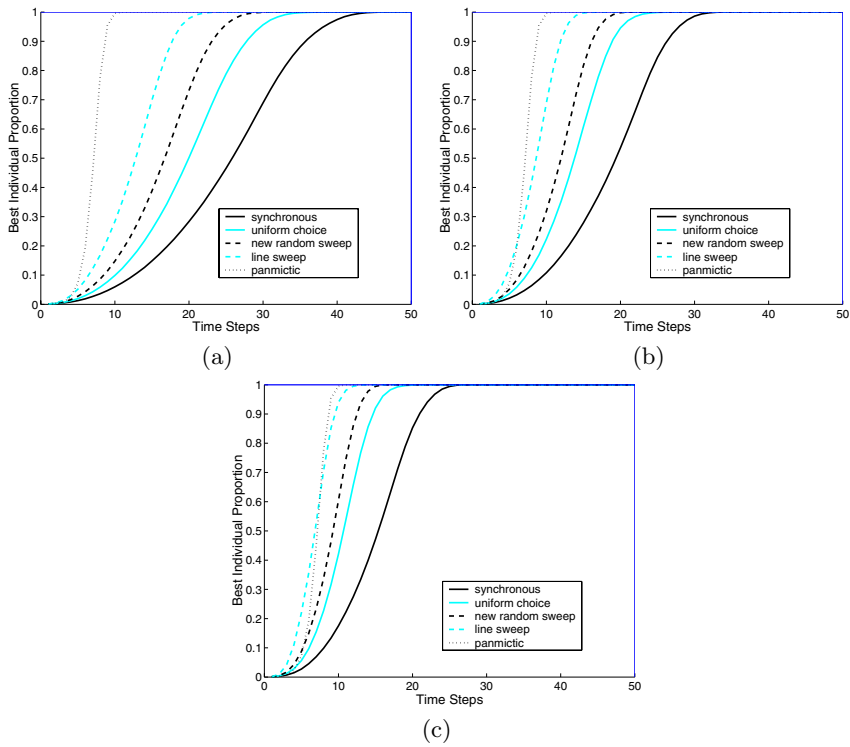


Fig. 2. Takeover times with binary tournament selection. Linear 5 neighborhood (a); Compact 9 neighborhood (b); Compact 13 neighborhood (c). Mean values over 100 runs. The vertical axis represents the proportion of population consisting of copies of the best individual as a function of the time step.

These results largely confirm the findings of Sarma and De Jong as far as synchronous and panmictic cEAs are concerned. Indeed, binary tournament and ranking induce very similar global selection pressure, while proportional selection exhibits less pressure. Moreover, for a given selection policy, larger neighborhoods induce a stronger selection intensity.

What is new in this paper (our contribution) is the behavior of the asynchronous models. Generally speaking, it can be observed that the asynchronous models give an emergent selection pressure that is between the panmictic upper bound and the synchronous lower bound. All graphs show that the global selection intensity grows going from uniform choice update to line sweep, with new random sweep and fixed random sweep in between, although an analysis of the variances should be conducted to quantitatively confirm the trend. This suggests that, by choosing the appropriate asynchronous update policy, one is able to control the selection pressure without using ad hoc numerical parameters. This opens new possibilities for dynamical EAs in which the selection pressure is under the control of the modeler even during the run (work in progress).

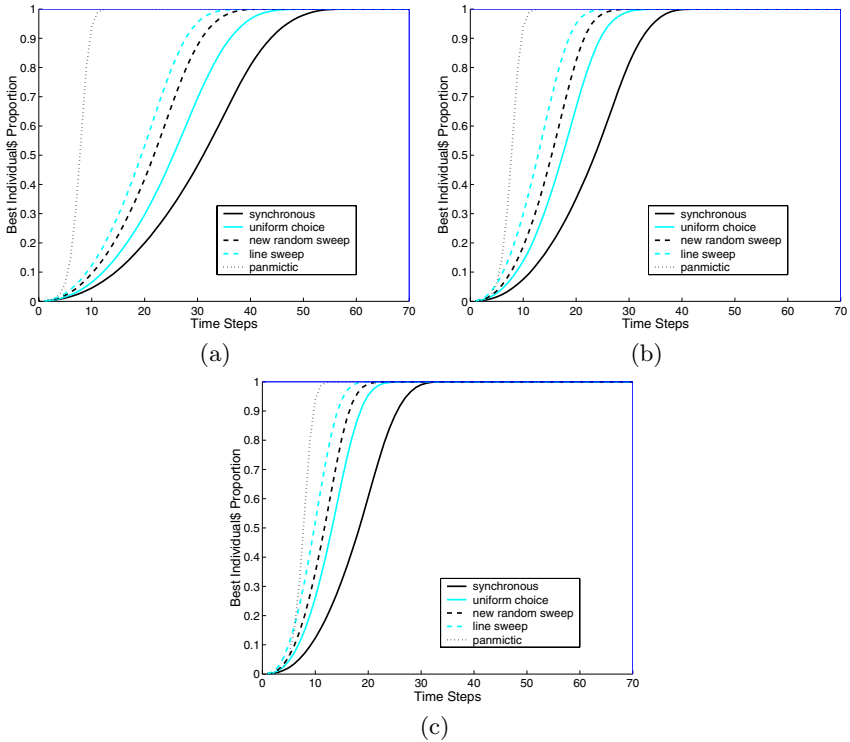


Fig. 3. Takeover times with fitness proportional selection. Linear 5 neighborhood (a); Compact 9 neighborhood (b); Compact 13 neighborhood (c). Mean values over 100 runs. The vertical axis represents the proportion of population consisting of copies of the best individual as a function of the time step. Note the change of scale on the horizontal axis to avoid cutting off the curves in figure (a).

The impression is confirmed by Table 1, where mean takeover times of all the update methods for each selection mechanism and each of the three considered neighborhoods are reported with their standard deviations.

4 Modelling Individual Growth

In this section, quantitative models for the individual growth (and thus for the different global selection pressures induced in the population) are presented using asynchronous update. We first give some statistical results valid for all finite 2-d cEAs discrete lattices. Next, we offer a quantitative analysis of the takeover time, and finally we hint at some possible improvements in the existing logistic model.

Table 1. Mean takeover time for the three selection mechanisms (vertically) and the five update methods (horizontally). Upper part: Linear 5 neighborhood. Middle: Compact 9 neighborhood. Lower part: Compact 13 neighborhood. The last column refers to the classical panmictic case. Standard deviations in parentheses.

LINEAR 5	Synchro	LS	FRS	NRS	UC	Panmictic
Roulette	52 (3.7)	34 (2.8)	37 (3.0)	37 (3.0)	44 (3.7)	12 (1.0)
Tournament	42 (2.7)	21 (1.9)	26 (2.0)	28 (1.9)	33 (3.7)	10 (0.7)
Ranking	39 (2.1)	18 (1.6)	24 (1.5)	24 (1.7)	30 (3.2)	10 (0.8)
COMPACT 9	Synchro	LS	FRS	NRS	UC	Panmictic
Roulette	38 (2.6)	23 (2.5)	26 (1.8)	26 (2.2)	29 (4.2)	12 (1.0)
Tournament	31 (1.8)	15 (1.4)	19 (1.4)	19 (1.5)	23 (2.9)	10 (0.7)
Ranking	30 (1.7)	13 (1.4)	18 (1.3)	18 (1.4)	22 (2.7)	10 (0.8)
COMPACT 13	Synchro	LS	FRS	NRS	UC	Panmictic
Roulette	31 (1.7)	18 (1.9)	20 (1.9)	20 (1.8)	23 (3.3)	12 (1.0)
Tournament	25 (1.4)	12 (1.2)	15 (1.0)	15 (1.2)	18 (2.9)	10 (0.7)
Ranking	25 (1.2)	11 (1.1)	14 (1.1)	15 (1.0)	18 (2.5)	10 (0.8)

4.1 Statistical Results on Information Propagation

Schönfish and de Roos [9] derived the expected value $E(Z)$ and the variance $V(Z)$ of the number of single steps between an update of a cell x and the next update of a cell $y \neq x$ in the neighborhood of x , $U(x)$, in order to compare the different asynchronous CA updating methods. These results can also be applied to asynchronous cEAs, where the local transition function f_0 does not deterministically determine the next state of the cell, but describes a probabilistic rule for such an update [11,13]. This rule is generally determined by the different selection, crossover and mutation mechanisms used in the EA. In our study of the takeover times induced by different update methods the local function f_0 only depends on the selection mechanism used.

Table 2. Values of the expected value $E(Z)$ and of the variance $V(Z)$ of the number of time steps between an update of a cell x and the next update of a cell $y \neq x$ in the neighborhood $U(x)$ for a cEA on a square grid. For Asynchronous Line Sweep, which depends from the chosen neighborhood, the Linear 5 neighborhood result is shown.

	Synchro	LS	FRS	NRS	UC
E(Z)	1	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{12} \left(7 + \frac{1}{n} \right)$	1
V(Z)	0	$\frac{1}{2} \left(\frac{1}{n} - \sqrt{n} + \frac{1}{2}n \right)$	$\frac{1}{12} (n - 2)$	$\frac{1}{12} \left(\frac{23}{12}n - \frac{13}{6} - \frac{13}{12n} \right)$	n-1

Schönfish and de Roos used the number of cell updates in their statistics. In order to be able to extend their results to the synchronous cEA, their statistics need to be translated using the number of time steps between an update of a cell x and the next update of a cell $y \neq x$ in the neighborhood $U(x)$ of x . In fact, for synchronous cEAs it is not possible to count the single cells updating, but a generational synchronous step can be compared to a time step of an asynchronous cEA. Table 2 contains the values of $E(Z)$ and $V(Z)$ for the four asynchronous and the synchronous updating policies. If in a cEA we keep the selection mechanism fixed and we vary the updating method, the results of Table 2 explain the ranking of the observed takeover times; notice that some experimental values are slightly different while theoretically they should be identical on the average. This is the case for Line Sweep and Fixed Random Sweep, as well as for Uniform Choice and Synchronous. However, these results do not explain the actual shapes of the selection pressure curves. We are currently working on the analytical study of the curves and some preliminary results are reported in the next two sections.

4.2 Fitting the Selection Pressure Curves

Sarma and De Jong [7] proposed a simple quantitative model for the study of the selection pressure curves for cEAs. They assumed that the diffusion of the best individual in the artificial evolution of a structured population would follow a logistic curve. Let us analyze their result, shown in Equation 1.

$$P_b(t) = \frac{1}{1 + \left(\frac{1}{P_b(0)} - 1 \right) e^{-\alpha t}} \quad (1)$$

This equation, where $P_b(t)$ represents the proportion of the best individual in the population at time t , was proposed for synchronous cEAs, and therefore we wondered whether it holds for asynchronous ones. Consequently, we proceeded to analyze the error (mean squared error) between an actual average selection pressure and the theoretically predicted values, for all the update methods considered in this work. The steps were (1) to compute the theoretical value of α , (2) to generate the predicted curve by using one point of the average observed performance curve, and (3) to compare it against the whole set of points of this observed curve.

To derive the α parameter we selected a mid point (with $P_b(t)$ around 0.5) from the experimental curves. Then, we generated the corresponding curve, and computed the squared error. Table 3 shows our measurements.

This table also shows that, although the fitting is satisfactory, it is not that good, since there exists a gap between the fitted curves and the experimental points. This claim is confirmed by Figure 4, in which the panmictic case is particularly good, while the other fittings could clearly be improved. This lead us to think that there could exist a better fitting for cellular EAs than the logistic one, whose main advantage is its similitude to the theoretical results existing for panmictic algorithms [3].

Table 3. Mean squared error between predicted and actual logistic fittings for Linear 5, Compact 9, and Compact 13 neighborhoods for all the update modes.

LINEAR 5	Synchro	LS	FRS	NRS	UC	Panmictic
Roulette	0.00406	0.00358	0.00372	0.00427	0.00358	0.00069
Tournament	0.00309	0.00270	0.00281	0.00274	0.00248	0.01053
Ranking	0.00366	0.00282	0.00257	0.00234	0.00290	0.00181

COMPACT 9	Synchro	LS	FRS	NRS	UC	Panmictic
Roulette	0.00349	0.00376	0.00284	0.00280	0.00332	0.00069
Tournament	0.00287	0.00194	0.00193	0.00178	0.00202	0.01053
Ranking	0.00311	0.00190	0.00197	0.00184	0.00209	0.00181

COMPACT 13	Synchro	LS	FRS	NRS	UC	Panmictic
Roulette	0.00339	0.00328	0.00199	0.00242	0.00281	0.00069
Tournament	0.00231	0.00148	0.00142	0.00149	0.00145	0.01053
Ranking	0.00225	0.00166	0.00153	0.00150	0.00122	0.00181

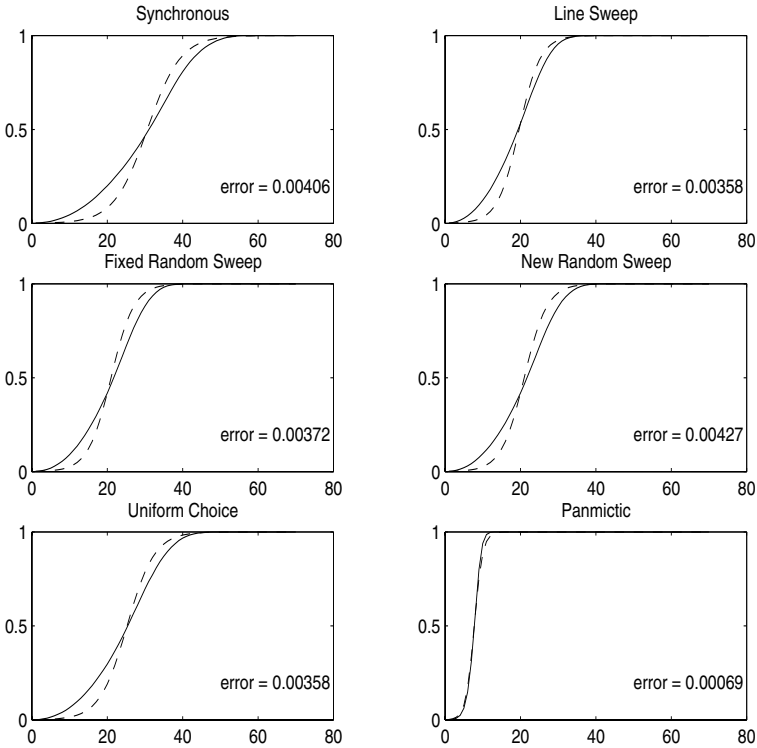


Fig. 4. Fitting of the experimental takeover time curves (full) with the logistic model (dashed) for the various update modes. Results refer to fitness proportional selection with Linear 5 neighborhood.

We have shown in this section that the logistic fitting should be improved for decentralized algorithms. In the next section some steps toward a more accurate model are described.

4.3 An Improved Model

It is well known since the work of Verhulst [12], that the assumption of logistic growth is true for biological populations within bounded resources. It is easy to see that this behavior also holds for the best individual growth in the artificial evolution of a finite panmictic population [3]. In fact, if we consider a population of size n , the number $N(t)$ of copies of the best individual in the population at time step t is given by the following recurrence:

$$\begin{cases} N(0) = 1 \\ N(t) = N(t-1) + p_s N(t-1)(n - N(t-1)) \end{cases} \quad (2)$$

where p_s is the probability that the best individual is chosen. This recurrence can be easily transformed into one that describes a logistic population growth in discrete time:

$$\begin{cases} N(0) = 1 \\ N(t) = N(t-1) + (p_s n) N(t-1) \left(1 - \frac{1}{n} N(t-1)\right) \end{cases} \quad (3)$$

Such a recurrence can be expressed in analytical form by the logistic equation:

$$N(t) = \frac{n}{1 + \left(\frac{n}{N(0)} - 1\right) e^{-\alpha t}} \quad (4)$$

where the growth coefficient α depends on the probability p_s and the population size n . This happens to be the approach taken in [8] for synchronous CEAs.

As suggested by Gorges-Schleuter in [5], in the artificial evolution of locally interacting, spatially structured populations, the assumption of a logistic growth doesn't hold anymore. In fact, in the case of a ring or a torus structure we have respectively a linear and a quadratic growth. We complete here her analysis which holds for unrestricted growth, extending it to bounded synchronously updated spatial populations.

For a structured population, let us consider the limiting case, which represents an upper bound on growth rate, in which the selection mechanism is deterministic, and a cell always chooses its best neighbor for updating. If we consider a population of size n with a ring structure (like that of 1-d cellular automata in which the two cells on the borders are linked) and a neighborhood radius of k (i.e. a neighborhood of a cell contains $2k + 1$ cells), the following recurrence describes the growth of the number of copies of the best individual:

$$\begin{cases} N(0) = 1 \\ N(t) = N(t-1) + 2k \end{cases} \quad (5)$$

This recurrence can be described by the closed equation $N(t) = N(0) + 2kt$, that clearly shows the linear character of the growth rate.

In the case of a population of size n disposed on a toroidal grid of size $\sqrt{n} \times \sqrt{n}$ (assuming $\sqrt{n}$ odd) and the Linear 5 neighborhood structure, the number of copies of the best individual can be described by the following recurrence:

$$\begin{cases} N(0) = 1 \\ N(t) = N(t-1) + 4t & , \text{ for } 0 \leq t \leq \frac{\sqrt{n}-1}{2} \\ N(t) = N(t-1) + 4(\sqrt{n}-t) & , \text{ for } t \geq \frac{\sqrt{n}-1}{2} \end{cases} \quad (6)$$

This growth is described by a convex quadratic equation followed by a concave one, as the two closed forms of the recurrence clearly show:

$$\begin{cases} N(t) = 2t^2 + 2t + 1 & , \text{ for } 0 \leq t \leq \frac{\sqrt{n}-1}{2} \\ N(t) = -2t^2 + 2(2\sqrt{n}-1)t + 2\sqrt{n}-n & , \text{ for } t \geq \frac{\sqrt{n}-1}{2} \end{cases} \quad (7)$$

Thus, a more accurate fitting should take into account the non-exponential growth followed by saturation (crowding effect).

5 Conclusions

In this work we have presented different update policies to deal with cellular EAs in a search for a more efficient algorithm with respect to its canonical form. We have investigated the induced selection pressure of such policies with respect to the widespread synchronous update and panmictic algorithms. Our results indicate that these two algorithms represent the smaller and higher (respectively) bounds to selection intensity, and that the asynchronous update methods represent intermediate values of pressure (which can even be ranked for such methods).

We have studied the existing proposals dealing with a logistic fitting that, although very similar to that existing for panmictic EAs and globally valid, are susceptible to further improvement for cellular EAs. We provide such an improved model of the best individual growth and additionally characterize the expected time between the update of two individuals residing in the same neighborhood for all the update methods considered in the paper.

Our aim in this paper has been to advance in the study of selection pressure in cellular EAs. Future work will consider more suitable functions for fitting the experimental curves (maybe a quadratic one). We will consider as well further extensions of the results to other aspects influencing the selection pressure of cellular EAs, such as the relationship existing between the neighborhood and the grid topology. Also, extensions to dimensions larger than two are being considered, as well as the application of cellular EAs to several hard problems. Our global aim is to obtain a body of knowledge of these algorithms, especially with regard to the numerical efficiency of the search.

Acknowledgments. The authors would like to acknowledge useful discussions on the models with A. Tettamanzi. The second author wishes to thank founding from the Spanish “Ministerio de Ciencia y Tecnología” and FEDER through contract TIC2002-04498-C05-02 (the TRACER project).

References

1. E. Alba, M. Giacobini, and M. Tomassini. Comparing synchronous and asynchronous cellular genetic algorithms. In J. J. Merelo et al., editor, *Parallel Problem Solving from Nature - PPSN VII*, volume 2439 of *Lecture Notes in Computer Science*, pages 601–610. Springer-Verlag, Heidelberg, 2002.
2. E. Alba and J. M. Troya. Cellular evolutionary algorithms: Evaluating the influence of ratio. In M. Schoenauer et al., editor, *Parallel Problem Solving from Nature, PPSN VI*, volume 1917 of *Lecture Notes in Computer Science*, pages 29–38. Springer-Verlag, 2000.
3. D. E. Goldberg and K. Deb. A comparative analysis of selection schemes used in genetic algorithms. In G. J. E. Rawlins, editor, *Foundations of Genetic Algorithms*, pages 69–93. Morgan Kaufmann, 1991.
4. M. Gorges-Schleuter. ASPARAGOS an asynchronous parallel genetic optimisation strategy. In J. D. Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms*, pages 422–427. Morgan Kaufmann, 1989.
5. M. Gorges-Schleuter. An analysis of local selection in evolution strategies. In *Genetic and evolutionary conference, GECCO99*, volume 1, pages 847–854. Morgan Kaufmann, San Francisco, CA, 1999.
6. B. Manderick and P. Spiessens. Fine-grained parallel genetic algorithms. In J. D. Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms*, pages 428–433. Morgan Kaufmann, 1989.
7. J. Sarma and K. A. De Jong. An analysis of the effect of the neighborhood size and shape on local selection algorithms. In H. M. Voigt, W. Ebeling, I. Rechenberg, and H. P. Schwefel, editors, *Parallel Problem Solving from Nature (PPSN IV)*, volume 1141 of *Lecture Notes in Computer Science*, pages 236–244. Springer-Verlag, Heidelberg, 1996.
8. J. Sarma and K. A. De Jong. An analysis of local selection algorithms in a spatially structured evolutionary algorithm. In T. Bäck, editor, *Proceedings of the Seventh International Conference on Genetic Algorithms*, pages 181–186. Morgan Kaufmann, 1997.
9. B. Schönfisch and A. de Roos. Synchronous and asynchronous updating in cellular automata. *BioSystems*, 51:123–143, 1999.
10. M. Sipper, M. Tomassini, and M. S. Capcarrere. Evolving asynchronous and scalable non-uniform cellular automata. In *Proceedings of International Conference on Artificial Neural Networks and Genetic Algorithms (ICANNGA97)*, pages 67–71. Springer-Verlag KG, Vienna, 1998.
11. M. Tomassini. The parallel genetic cellular automata: Application to global function optimization. In R. F. Albrecht, C. R. Reeves, and N. C. Steele, editors, *Proceedings of the International Conference on Artificial Neural Networks and Genetic Algorithms*, pages 385–391. Springer-Verlag, 1993.
12. P. F. Verhulst. *Mem. Acad. Roy. Bruxelles*, 28(1), 1844.
13. D. Whitley. Cellular genetic algorithms. In S. Forrest, editor, *Proceedings of the Fifth International Conference on Genetic Algorithms*, page 658. Morgan Kaufmann Publishers, San Mateo, California, 1993.

A Case for Codons in Evolutionary Algorithms

Joshua Gilbert and Maggie Eppstein

Dept. of Computer Science, University of Vermont,
Burlington VT, 05405 USA.

{jgilbert, eppstein}@emba.uvm.edu

Abstract. A new method is developed for representation and encoding in population-based evolutionary algorithms. The method is inspired by the biological genetic code and utilizes a many-to-one, codon-based, genotype-to-phenotype translation scheme. A genetic algorithm was implemented with this codon-based representation using three different codon translation tables, each with different characteristics. A standard genetic algorithm is compared to the codon-based genetic algorithms on two difficult search problems; a dynamic knapsack problem and a static problem involving many suboptima. Results on these two problems indicate that the codon-based representation may promote rapid adaptation to changing environments and the ability to find global minima in highly non-convex problems.

1 Introduction

The design and implementation of evolutionary algorithms as a means of computational optimization have been inspired by biological evolution through natural selection. It is generally recognized that there are several important aspects that contribute to the performance of a genetic algorithm, including the underlying data representation, method of selection, genetic operators (e.g., mutation and recombination), fitness evaluation, population structure, and the nature of the problem being solved [1]. While the choices of these aspects are not independent of each other, the purpose of this work is to explore a new method for data representation, motivated by the genetic code. Herein we use the term “genotype” to mean the data upon which the genetic operators are applied, and the term “phenotype” to mean the data that is evaluated in the fitness function.

1.1 Genotypic and Phenotypic Data Representation in Evolutionary Algorithms

Many optimization problems are of the form $F(\mathbf{p}) = \text{fitness}: \mathbb{P}^{np} \rightarrow \mathbb{R}$, where we attempt to find the optimal phenotype vector $\mathbf{p}$, of length np , that optimizes the fitness function F , possibly subject to some additional constraints. In discrete search problems, the phenotype alphabet $\mathbb{P}$ may be the set of integers $\mathbb{I}$, or some subset thereof. In continuous optimization problems, the phenotype alphabet $\mathbb{P}$ may be the set of real numbers $\mathbb{R}$, or some subset thereof. In evolutionary algorithms, the phenotype $\mathbf{p} \in \mathbb{P}^{np}$ may be represented by a genotype $\mathbf{g} \in \mathbb{G}^{ng}$, of length ng , such that

there is an encoding $\gamma(\mathbf{g})=\mathbf{p}: \mathbf{G}^{ng} \rightarrow \mathbf{P}^{np}$. For example, in “simple” genetic algorithms the genotype is represented as a binary string, where each ng/np successive bits are translated to the phenotypic alphabet using a standard binary encoding or Gray encoding function [1, 2], although recently there has been a trend towards use of strings of integers or real numbers directly for both genotype and phenotype (i.e., γ is trivially the identity function), as has been the case all along in evolution strategies [1]. Conceptually, there is nothing preventing the implementation of evolution strategies with a nontrivial encoding function γ between genotype and phenotype. Regardless of whether there is a trivial or non-trivial encoding function, the key point here is that the mapping from genotype to phenotype is almost always one-to-one.

There are some interesting exceptions to this generalization, however, and where redundant encodings have been used they have often been shown to offer advantages. In “messy GAs” [3] the adjacencies between alleles in the variable-length genotype are explicitly stored with index numbers; overdetermined strings are disambiguated using a conflict-resolution mechanism, and are thus an implementation of a dynamically changing many-to-one mapping. However, the emphasis in messy GAs, which have been shown to perform well on deceptive problems, has been on their ability to overcome linkage problems via the explicit adjacency representation rather than on the potential benefits offered by the redundancy. Diploidy and dominance [4] offer another way to effect a dynamic many-to-one mapping between genome and phenome, and have been shown to improve performance in some dynamic optimization problems. Similarly, gene “regulation” can be used to turn various genes on and off and thereby maintain hidden diversity, enabling rapid adaptation to dynamic changes in the fitness landscape [5].

Given that evolutionary computation algorithms were inspired by biological evolution, we find it somewhat surprising that we could find few references in the literature to implementations that adopted a more biologically motivated encoding between genotype and phenotype. Banzhaf and colleagues [6-8] are a rare exception to this, and have shown that they can improve results in genetic programming by adopting a partially redundant genotype-to-phenotype encoding, loosely modeled on the genetic code, to map binary strings to symbols in a genetic programming application. We briefly discuss biological encoding in the next section, and then address how we have adapted this for use in evolutionary algorithms.

1.2 Genotypic and Phenotypic Data Representation in Biological Systems.

In DNA, there is a 4-letter alphabet at the genotype level $\mathbf{G}=\{\mathbf{a}, \mathbf{g}, \mathbf{c}, \mathbf{t}\}$, representing the four nucleotide bases (adenine, guanine, cytosine, and thymine). A string of DNA bases is translated into a protein, or string of amino acids, using a codon triplet translation scheme (i.e. each sequence of 3 bases within the coding portions of genes is translated into 1 amino acid, or terminates the amino acid sequence). Mutation and recombination occur at the genotype level on the DNA strings, whereas the translated sequences of amino acids are the proteins that comprise the basis of the phenotype upon which selection acts. Since there are $4^3=64$ possible codon values, but only 20 amino acids (plus the stop codon), this is a redundant coding scheme (64 to 21) with some distinct codons mapping to the same amino acid, as shown in Table 1.

Table 1. The “universal” codon translation table from codon triplets to the 20 amino acids (shown by their standard 3-letter abbreviations) and the stop codon [9]. Here, the three dimensions of the table have been compressed to two

		2 nd codon position				
1 st codon position		T	C	A	G	
	T	Phe	Ser	Tyr	Cys	T
		Phe	Ser	Tyr	Cys	C
		Leu	Ser	Stop	Stop	A
		Leu	Ser	Stop	Trp	G
	C	Leu	Pro	His	Arg	T
		Leu	Pro	His	Arg	C
		Leu	Pro	Gln	Arg	A
		Leu	Pro	Gln	Arg	G
	A	Ile	Thr	Asn	Ser	T
		Ile	Thr	Asn	Ser	C
		Ile	Thr	Lys	Arg	A
		Met	Thr	Lys	Arg	G
	G	Val	Ala	Asp	Gly	T
		Val	Ala	Asp	Gly	C
		Val	Ala	Glu	Gly	A
		Val	Ala	Glu	Gly	G

A quick examination of Table 1 reveals that the organization of the genetic code is non-random, with most of the redundancy occurring in the 3rd codon position. Crick [10] has proposed, in his “wobble hypothesis”, that this may be due to biochemical limitations that cause decoding non-specificity by transfer RNA. However, statistical arguments have also been used to show that the code is “optimal” in that it minimizes errors, since erroneous codons either code for the same amino acid or to ones with similar hydrophobicity, therefore minimizing the phenotypic impact of single nucleotide polymorphisms [11]. More recently, Freeland [12] proposed that the redundant genetic code may actually act to maximize the rate of adaptive evolution to changing environments. Doubtless there are biochemical constraints that have strongly influenced why life as we know it has evolved with a base alphabet of 4, a codon size of 3, and this particular redundant codon mapping to the 21 possible phenotypic outcomes. Nonetheless there is, biochemically speaking, room for at least some variability. For example, there is a growing body of information regarding the existence of non-standard genetic codes in vertebrate mitochondria as well as in nuclear DNA in a wide range of organisms [13]. Even the size of the basic alphabet is potentially variable [14-15]. Given the highly conserved nature of the redundant codon-based genetic code, despite its demonstrated evolvability, we are motivated to explore the potential advantages such an encoding may have to offer in evolutionary search.

Certainly, the implementation of a codon-based representation affects the possible schemata [1] that can be manifested. Over the past decade, the potential information theoretic benefits of a codon-based representation have begun to receive some attention in the literature [6-8, 12, 16]. Using a Fourier analysis, Kargupta [16] demonstrates that some, but not all, codon-like translation schemes can enable

polynomial time approximations of some exponentially hard problems. However, his analysis is so far restricted to binary alphabets, and does not address the more general problem of how to optimally construct a codon translation scheme for arbitrary phenotypic alphabets and arbitrary problems. Freeland [12] argues that the structure of the redundant genetic code maximizes the probability that mutations will be favorable in dynamic fitness landscapes. The empirical studies by Banzhaf's group [7-8] found that a binary-to-symbolic codon mapping scheme in a genetic program improved search performance over a traditional genetic program in which genotype and phenotype were the same. These studies are intriguing and indicate the need for more comprehensive research in this area. In this manuscript we report on preliminary studies in codon-based representations in genetic algorithms.

2 Methods

We restricted our test problems to those with phenotype alphabet size of 21 distinct values, as in the biological phenotype. In what we will refer to as the "non-codon GA", the alleles in each chromosome consisted of 21 distinct integers. In the "codon GA" we continued with the biological metaphor and restricted the alleles in each genotypic chromosome to an alphabet size of 4 and a codon size of 3 (each three adjacent alleles were logically associated into a codon triplet). Thus, the codon-based chromosomes had three times as many entries as the non-codon based chromosomes, although the alphabet size was smaller (i.e., with packing, memory requirements could be made similar). In both cases, mutation and recombination occurred on these chromosomes of genotypic alleles. Mutation step size for each allele in the genotype, for the experiments reported here, followed a negative exponential distribution, with a range of mutation step size of 0..1 for the codon-GA genotype alphabet {1..4} and a range of 0..2 for the non-codon GA genotype alphabet {0..20} (we determined these values through experimentation with a variety of mutation step sizes and selected the values that generated the best performance for each type of GA on our test problems). Single-point crossover was employed for recombination, and elitism was implemented such that the best individual from each generation was included in the population for the next generation.

In the non-codon GA, the genotype doubled as the phenotype; i.e., the modified chromosomes were directly input into the objective function for fitness calculation (Figure 1a). In the codon GA, each codon (sequence of three alleles) was first mapped to one of the 21 possible phenotype values via a redundant codon translation table to yield the phenotype (sequence of values each in range 0..20); the phenotype was then input to the objective function for fitness calculation (Figure 1b).

Selection was performed using stochastic universal sampling [1]. For each problem, we ran 200 replicates (i.e., we used 200 distinct starting seeds for the random number generator) of the non-codon GA and the codon GA with three distinct codon translation tables (see Section 2.2), where each replicate had a population size of 300 individuals (chromosomes). For the results shown here, each replicate was allowed to run for 100 generations. Monte Carlo replicates of the non-codon GA and each codon GA were paired, in that the same set of 200 starting seeds for the random number generator was used for the non-codon and codon versions, and the paired individuals started with the same fitness. This latter was accomplished by first

generating random populations of individuals for the non-codon GA, then stochastically decoding them. I.e., using the codon table in reverse, we stochastically selected one of the matching codon triplet values for each corresponding phenotypic value, in order to ensure initial genotypes that yielded the same initial phenotypes as in the non-codon GA.

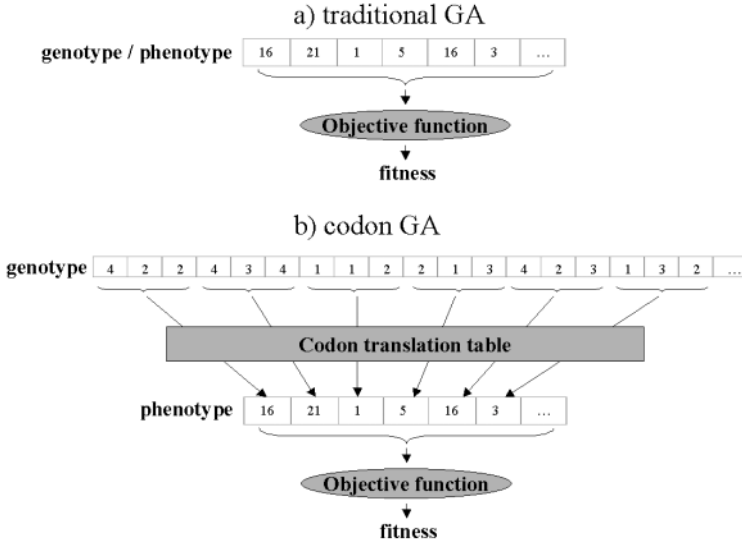


Fig. 1. Schematic of our implementation of data representation in a) a traditional non-codon genetic algorithm, and b) genetic algorithms using codon translation tables to separate phenotype from genotype

2.2 Creation of Codon Translation Tables

We defined two indices to characterize codon tables. The “similarity index” is a measure of how similar the codon triplets are within the various semantic “groups”, where a semantic group is defined as the set of codon values that translate into the same phenotypic value. In a codon triplet C , two distinct codon values (C' and C'') can share allele values at 0, 1, or 2 positions; we use the number of shared allelic values, at each of the 3 codon allele positions k , between codons in a group as a measure of similarity within the group. The sum (S_g) of the maximum possible number of shared alleles between any two pairs of codons in a group g with a given number of members n_g can be calculated based on the size of the genetic allele alphabet (e.g., 4), and the size of a codon (e.g., 3), and is used to normalize the index. The similarity index (Sim_t) of a codon mapping table t with m groups (e.g., 21) is then defined as follows:

shared alleles, dashed lines denote 1 shared allele, and dotted lines denote 0 shared alleles. The biological codon table has a high similarity index $Sim_{biological}=0.92$ (most of the codons in each group share the same first two allelic values, as reflected by all the vertical lines), but a relatively low adjacency index $Adj_{biological}=8.10$, suggesting that perhaps evolution has acted to maximize similarity but minimize adjacency.

In the biological codon table, semantic group sizes range from 1 to 6, reflecting the unequal frequencies of the various amino acids and the stop codon. However, if we are to adapt the codon translation idea to unbiased evolutionary algorithms designed to solve arbitrary black box optimization problems, then we desire equal *a priori* probabilities of all possible phenotypic values (i.e., all semantic groups should be as close to the same size as possible). For these initial studies, we selected problems with the same alphabet sizes as in the biological problem. Thus, within the constraints of mapping the 64 possible codon values onto 21 phenotypic values, we chose to create 20 codon group sizes of 3 and one codon group size of 4. We created 3 distinct mapping tables to try with our codon GA, each with different characteristics of similarity and adjacency. The first table, HL, has High (maximum) similarity and Low adjacency, similar to the characteristics of the biological codon table (Figure 3a). The second table, HM, has High (maximum) similarity and Medium high adjacency (Figure 3b). The third table, LH, has Low similarity (the minimum possible, given that there is one codon group size of 4) and High adjacency (Figure 3c). The similarity and adjacency indices of these three tables, and of the biological table, are given in Table 2. In this study, we have not yet accounted for the phenotypic nearness of adjacent phenotypes in the codon translation table; phenotypes were arbitrarily assigned to codon groups.

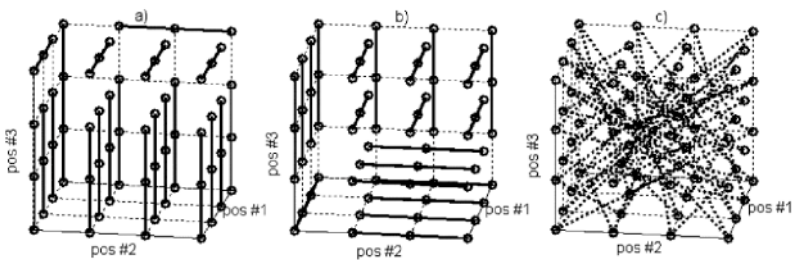


Fig. 3. Graphical depictions of the 3 codon translation tables used in this study. a) HL, b) HM, and c) LH. Meanings of linetypes are as described in Figure 2

2.3 Test Problems

In these preliminary studies, we tested the non-codon GA and the codon GA (using each of the three codon translation tables) on two arbitrarily selected difficult test problems: 1) Dynamic Knapsack, and 2) Indecisive.

2.3.1 Dynamic Knapsack

The objective in the “Dynamic Knapsack” problem is to fill a knapsack with items, so as to maximize the sum of the values of the items in the knapsack, without violating a

weight constraint. (This was converted to a minimization problem by inverting the sum of values; a penalty term was used to enforce the weight constraint.) We arbitrarily specified that there were 5 distinct object types with randomly assigned real values and weights. We allowed between 0 and 20 of each item type placed in the knapsack. The phenotype for each individual was thus a sequence of 5 integers, each in the range 0..20. The problem was made dynamic by alternating the weight constraint between 100 and 400 each 5 generations. The true optimal solution was determined for each random set of values and weights, and for each weight constraint, by exhaustive search, for comparison to evolved solutions.

Table 2. Similarity and adjacency indices of the biological codon table and the three artificial codon tables used in these experiments

Codon Table	Sim	Adj
Biological	0.92	8.10
HL	1.00	7.05
HM	1.00	11.05
LH	0.01	15.52

2.3.2 Indecisive

The phenotype for the “Indecisive” problem is again a sequence of integer values, each in the range 0..20. The results shown here are for sequences of length 10. The objective function is computed as follows: 1) count the frequency of occurrence of each value, 0 through 20. 2) Subtract 1 from all frequency counts *except for* the count for one arbitrarily chosen (but fixed) number. 3) Take the maximum of the resulting modified counts. 4) Invert this value to turn this into a minimization problem. Note that Indecisive is a difficult search problem in that it has a multi-modal search space with many attractive local suboptima, but only one global optimum, all located at maximum Hamming distances from one another.

3 Results

3.1 Dynamic Knapsack

The results for Dynamic Knapsack, on one set of representative random object weights and values, are plotted in Figures 4 and 5. In all cases the GAs exhibited rapid adaptation to each of the individual weight constraints over each series of 5 generations for which the same weight constraint applied. Since all solutions found that satisfied the weight constraint of 100 were also feasible solutions at the weight constraint of 400, the transition from weight constraint of 100 to 400 was relatively easily accommodated. However, since the converse is not true, the transition from weight constraint of 400 to 100 caused a large increase in objective function value and this “overshoot” was notably worse for the non-codon GA than for the codon GA, and continued to get worse with each cycle (Figure 4). On the other hand, overshoot for all three codon GAs was much lower than for the non-codon GA (note the log scale in Figure 4), and tended to stay constant or even reduce (e.g., when using the

codon translation table LH) over successive cycles. Because the scale differences in these results make comparisons difficult to see in Figure 4, we present a different view of the data in Figure 5. In Figure 5a, we show the objective function values at the end of each period with weight constraint of 100; here, it is apparent that the non-codon GA is not able to recover quickly enough from the increasing overshoot in order to adapt well to the periodic weight constraint. A close-up view of this same data is plotted in Figure 5b, to enable discrimination between the codon GA using the three tables. All three codon tables gave good results and in contrast to the non-codon GA they all continued to improve their adaptation to the weight constraint of 100 over the periodic cycles. It is of interest that codon table HL had the highest overshoot (i.e., greatest variability in solution quality) (Figure 4) but evolved the lowest objective function values at the ends of each 5 successive generations at weight constraint 100. In Figure 5c we show the objective function values at the end of each period with weight constraint of 400; in this case, all the GAs did well and the performance over successive periodic cycles was fairly constant, with the non-codon GA showing slightly better adaptation to the higher weight constraint than the codon GAs.

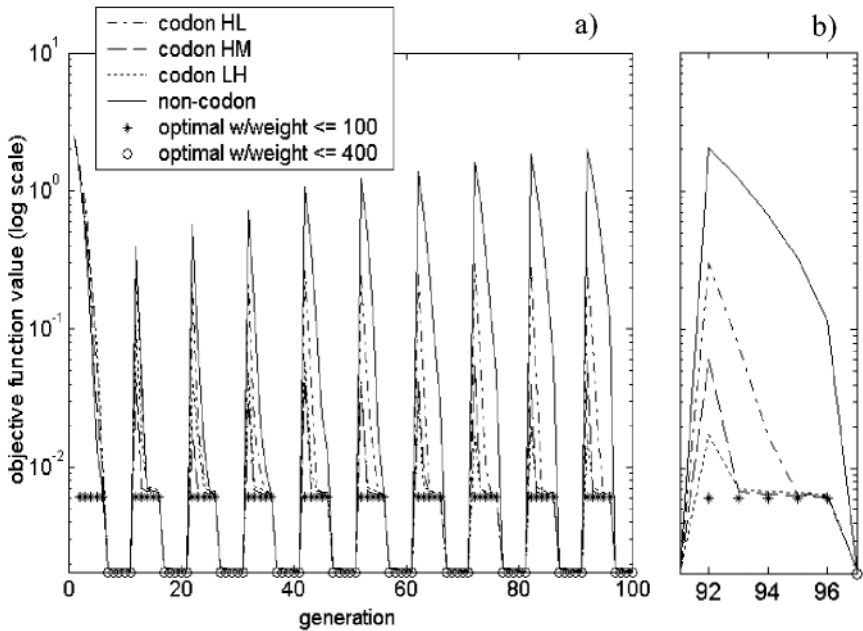


Fig. 4. a) Average over 200 replicates of the objective function values for the best individual in a population of 300 individuals for both codon and non-codon GAs on the Dynamic Knapsack problem. The symbols indicated the step function representing the true global optima under the two different weight constraints. Note the log scale. b) Close up of the last cycle with weight constraint 100.

3.2 Indecisive

The results on Indecisive are shown in Figure 6, where there is a dramatic difference in performance between the non-codon GA, which never found the global optimum within 100 generations, and the codon GAs, which always found the global optimum

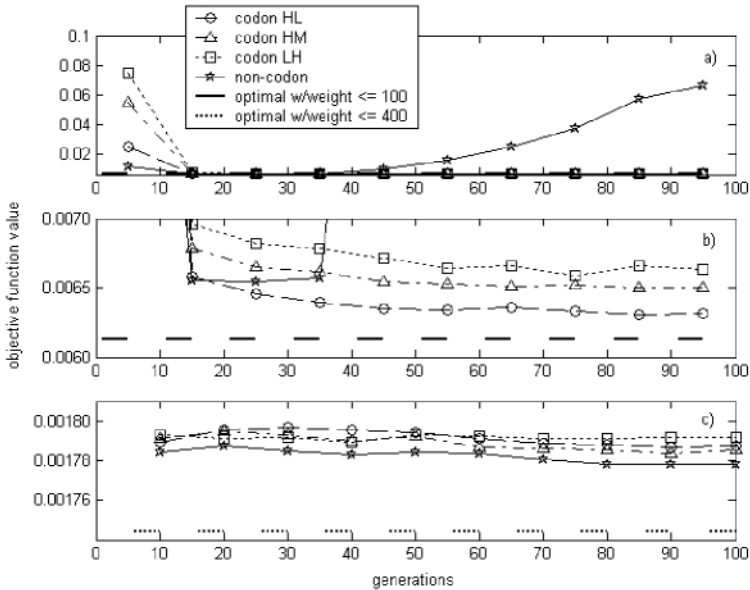


Fig. 5. Three views of the data plotted in Figure 4. a) The objective function value at the end of each interval with a weight constraint of 100. b) Close-up of data plotted in a. c) The objective function value at the end of each interval with a weight constraint of 400

rapidly. In this problem, the codon GA using codon table HL consistently found the global optimum the most rapidly.

4 Conclusions

Our preliminary results imply that, at least for some problems, a redundant codon-based representation positively affects the ability of a genetic algorithm to a) rapidly adapt to dynamic changes in the fitness function and b) find a global optimum in a difficult multi-modal search space. The optimal choice of how to organize the codon translation table for a cyclical fitness landscape appears to depend on whether it is more important to minimize the variability of solution quality (low similarity, high adjacency) or the evolution of the best solution quality over time (high similarity, low adjacency). In the static problem we tested, the codon translation table with high

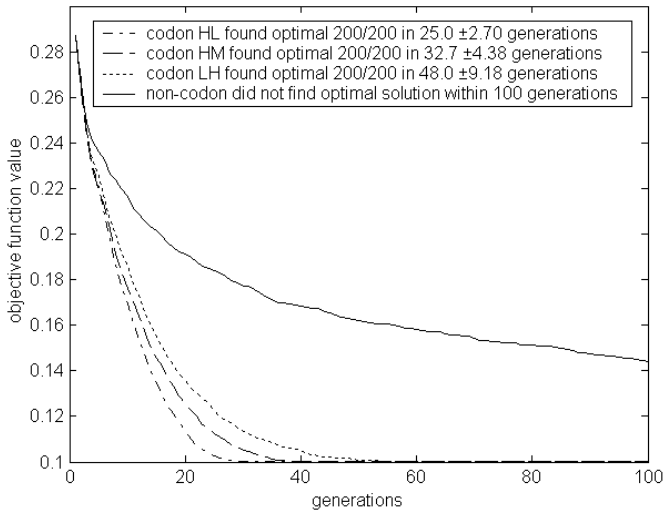


Fig. 6. Average, over 200 replicates, of the objective function values for the best individual in a population of 300 individuals, for both codon and non-codon GAs on the Indecisive problem. Values in the legend indicate the number of times the global optimum was found out of the 200 replicates, and the number of generations required to find it is shown as mean $\pm$ standard deviation

similarity and low adjacency yielded both lowest average values and variances of number of generations to find the single global minimum. Interestingly, high similarity and low adjacency are the characteristics that have evolved in the biological codon table. Much more work in this area is warranted to further understand the mechanisms by which redundant codon-based representations enhance search, which evolutionary algorithms and application problems can benefit from this approach, and how to best select the fundamental parameters such as the size of the codon, the size of the genotypic alphabet at each codon position, and the internal organization of the codon translation table. In addition, it will be important to explore whether the effects of redundant codon translation schemes are complementary, synergistic, or alternatives to the benefits offered by other types of redundant data representations in evolutionary algorithms.

Acknowledgement. This work was supported in part by DE-FG02-00ER45828 awarded by the US Department of Energy through its EPSCoR Program.

References

1. Bäck T. Fogel, D.B., and Michalewicz, T. (eds): Evolutionary Computation 1: Basic Algorithms and Operators, Inst. of Physics Publ., Bristol, UK (2001)

2. Rothlauf, F.: Binary representations of integers and the performance of selectorcombinative genetic algorithms. In: Proceedings of the Genetic and Evolutionary Computation Conference. Morgan Kaufmann, San Francisco, California (2002) 695
3. Goldberg, D.E., Deb, K., and Korb, B.: Don't worry, be messy. In: R.K. Belew and L.B. Booker, (eds.): Proc. 4th Intl. Conf. On Genetic Algorithms. Morgan Kaufman, San Mateo, CA (1991) 24–30
4. Goldberg, D.E., and Smith, R.E.: Nonstationary function optimization using genetic algorithms with dominance and diploidy. In J.J. Grefenstette, (ed.): 2nd Intl. Conf. On Genetic Algorithms. Lawrence Erlbaum Assoc. (1987) 59–68
5. Dasgupta, D.: Incorporating redundancy and gene activation mechanisms in genetic search for adapting to non-stationary environments. In: L. Chambers, (ed.), Practical Handbook of Genetic Algorithms: New Frontiers (Vol II). CRC Press, (1995) 303–316
6. Banzhaf, W.: Genotype-Phenotype-Mapping and Neutral Variation – A case study in Genetic Programming. In: Y. Davidor, H.-P. Schwefel and R. Männer, (eds.) Proc. Parallel Problem Solving from Nature III, Lecture Notes in Computer Science (1994) 322–332
7. Keller, R., and Banzhaf, W.: Genetic Programming using Genotype-Phenotype Mapping from Linear Genomes into Linear Phenotypes. In: J. Koza, D. Goldberg, D. Fogel, R. Riolo, (eds.) Proc. 1st annual Conf. on Genetic Programming (GP-96). MIT Press, Cambridge, MA (1996) 116–122
8. Banzhaf, W.: The Evolution of Genetic Code in Genetic Programming. In: W. Banzhaf, J. Daida, A. Eiben, M. Garzon, V. Honavar, M. Jakiela and R. Smith (eds.) Proc. of first GECCO conference, Morgan Kaufmann, San Francisco. (1999) 1077–1082
9. Creighton, T.E.: Proteins: Structures and Molecular Properties, 2nd Ed., W.H. Freeman and Co., New York (1993)
10. Crick, F.H.C.: Codon-anticodon pairing: the wobble hypothesis. *J. Mol. Biol.*, **19** (1966) 548–555
11. Freeland, S.J. and Hurst, L.D.: The genetic code is one in a million. *J. Mol. Evol.*, **47** (1998) 238–248
12. Freeland, S.J.: The Darwinian genetic code: an adaptation for adapting? *Genetic Programming and Evolvable Machines*, **3** (2002) 113–127
13. Knight, R.D., Freeland, S.J., and Landweber, L.F.: Rewiring the keyboard: evolvability of the genetic code. *Nature Reviews: Genetics*, **2** (2001) 49–58
14. Szathmáry, E.: What is the optimum size for the genetic alphabet? *Proc. Natl. Acad. Sci., USA*, **89** (1992) 2614–2618
15. Piccirilli, J.A., Karuch, T., Moroney, S.E., and Benner, S.A.: Enzymatic incorporation of a new base pair into DNA and RNA extends the genetic alphabet. *Nature* **343** (1990) 33–37
16. Kargupta, H.: A striking property of genetic code-like transformations. *Complex Systems*, **13** (2001) 1–32

Natural Coding: A More Efficient Representation for Evolutionary Learning^{*}

Raúl Giráldez, Jesús S. Aguilar-Ruiz, and José C. Riquelme

Department of Computer Science, University of Seville
Avenida Reina Mercedes s/n, 41012 Sevilla, Spain
{giraldez,aguilar,riquelme}@lsi.us.es

Abstract. To select an adequate coding is one of the main problems in applications based on Evolutionary Algorithms. Many codings have been proposed to represent the search space for obtaining decision rules. A suitable representation of the individuals of the genetic population can reduce the search space, so that the learning process is accelerated by decreasing the number of necessary generations to complete the task. In this sense, natural coding achieves such reduction and improves the results obtained by other codings. This paper justifies the use of natural coding by comparing it with hybrid coding that joins well-known binary and real representations. We have tested both codings on a heterogeneous subset of databases from the UCI Machine Learning Repository. The experiments' results show that natural coding improves the quality of the obtained knowledge-model using only one third of the generations that hybrid coding needs as well as a smaller population.

Keywords: Evolutionary Algorithms, Coding, Supervised Learning

1 Introduction

Machine Learning is used when we want to build a knowledge model from a training dataset and predict the outcome of a new unseen instance. Normally, the instances of the dataset are named examples and each example is formed by attributes. These attributes can be classified by different criteria according to the kind of values that they take. The most common classification is in continuous and discrete attributes. If an attribute takes values in a real and infinite domain, we say that it is continuous. On the contrary, if the domain is a finite set of values, we name it as a discrete attribute. Usually, one of these attributes is selected as a decision attribute or class. Thus, the knowledge model is built to predict the value that the class takes in new examples depending on the other attributes values. When the class of the training data is known (labeled dataset), we work in the Supervised Learning field.

^{*} This research was supported by the Spanish Research Agency CICYT under grant TIC2001-1143-C03-02.

The knowledge model can be represented by different structures: decision rules, association rules, decision trees, etc. In Supervised Learning, the structures generated are usually decision rules or trees. The databases have a set of attributes or characteristics that define each example. A decision rule establishes conditions which the examples must fulfil in order to be classified by this rule. These conditions affect the values that the attributes can take in that if the attributes of an example meet all the conditions that a rule establishes then we say that the rule covers the example, independent of whether it classifies such example correctly or not. The usual representation of rules generated by learning systems is shown in Figure 1.

If $Cond_1$ and $Cond_2$ and... and $Cond_M$ then $Class = C$

Fig. 1. Decision Rule.

where $Cond_i$ is the condition that the i^{th} attribute of an example has to satisfy in order to be classified with class C , and M is the number of attributes in the database. Logically, the case can arise where an attribute does not appear in the rule, from which we assume that the condition concerning the attribute is always evaluated as *true*. When an attribute (a_i) is continuous, the condition ($Cond_i$) takes the form $a_i \in [l_i, u_i]$, restricting the range of values of the attribute to the interval defined by the lower (l_i) and the upper (u_i) bound. On the other hand, when the attribute is discrete, the condition takes the form $a_i \in \{v_1, v_2, \dots, v_k\}$, where the values $\{v_1, v_2, \dots, v_k\}$ are not necessarily all those the attribute can take.

In literature, there are many methods that acquire the inherent knowledge from a labeled dataset and generate a structure that represents it: CN2 [6,7], AQ-based systems [13], OC1 [15], C4.5 [16], SEE5.0 [17], SIA [20], among others. This work is focused on those methods that apply Evolutionary Algorithms (henceforth EA) in the training phase, and specifically in the tool HIDER (Hierarchical Decision Rules) [2]. This tool generates a set of hierarchical decision rules by applying an EA whose population is a set of encoded rules. Thus, each individual is a decision rule where each attribute is coded by one or several genes. Different versions of HIDER have used different codings for the individuals of population: binary, real and hybrid. This work is focused on a new version of this tool, called HIDER*, which applies the natural coding [1] to represent the individuals. Thus, we compare the two last versions, HIDER and HIDER*, that use the hybrid and natural coding respectively. Both codings can treat continuous and discrete attributes. Hybrid coding applies binary coding for discrete attributes and real coding for continuous. Natural is an original coding that represents both kinds of attributes by means of a simple natural number. As it is explained later, the main advantage of natural coding is the reduction of the search space.

Our goal in this paper is to present the advantages of natural coding against hybrid coding. HIDER and HIDER* were run and their performances measured. The results show that HIDER* finds solutions before HIDER, since natural coding

reduces the number of candidate solutions. Thus, HIDER* can be run with fewer generations and with a smaller population size than HIDER.

The rest of this paper is organized into the following: Section 2 describes the EA used by HIDER and HIDER*. The characteristics of hybrid coding and natural coding are presented in Sections 3 and 4 respectively. The experimental results of our research are presented in Section 5. Finally, in Section 6, the conclusions of this work are summarized.

2 Algorithm

HIDER is a tool that produces a hierarchical set of rules. When a new example needs to be classified, the set of rules is sequentially evaluated according to its hierarchy, so if the example does not fulfil a rule, the next one in the hierarchy order is evaluated. This process is repeated until the example matches every condition of a rule and is classified with the class that such rule establishes.

HIDER uses an EA to search for the best solutions. Since the aim is to obtain a set of decision rules, the population of the EA is formed by some possible solutions. Each genetic individual is a rule that evolves by applying the mutation and crossover operators. In each generation, some individuals are selected according to their goodness of fit and they are included in the next population along with their offspring. Thus, the performance of the tool is influenced by two factors: the fitness function and the coding. The fitness function used in this research is

$$f(r) = N - CE(r) + G(r) + coverage(r) \quad (1)$$

where r is an individual; N is the number of examples being processed; $CE(r)$ is the class error, i.e. the number of examples belonging to the region defined by the rule r but they do not have the same class; $G(r)$ is the number of examples correctly classified by r ; and $coverage(r)$ gives the proportion of the search space covered by the rule. This fitness function is described in detail in [2]. The goal of our work is focused on the coding, so the another factor is not analyzed in detail. In fact, HIDER and HIDER* use the same EA and the same fitness function. The difference between both is in the coding. HIDER applies hybrid coding, whereas HIDER* uses natural coding.

The pseudocode of HIDER is shown in Figure 2. The main algorithm is a typical sequential covering method [14], where the algorithmic function that produces the rules is an EA. Each call for this function (line 8) generates only one rule that is inserted into the final set of rules (line 9) and is used to eliminate examples from the training data (line 10). The evolutionary function is started again with the reduced training data. This loop is repeated until the set of training data is empty.

The function *EvoAlg* has a set of examples as an input parameter. It returns a rule which is the best individual of the last generation. The initial population is built randomly by the function *InitializePopulation*. Some examples are randomly selected and individuals that cover such examples are generated. After

```

1 Procedure HIDER
2   Input: T: File of examples (Training file)
3   Output: R: Set of rules (Sorted set)
4   begin
5      $R := \emptyset$ 
6      $initialSize := |T|$ 
7     while  $|T| > initialSize$ 
8        $r := \text{EvoAlg}(T)$ 
9        $R := R \oplus r$ 
10      DeleteCoveredExamples(T,r)
11    end while
12 end HIDER

13 Function EvoAlg (T: File of encoded-examples) ret(r: Rule)
14 begin
15   InitializePopulation(P)
16   For  $i := 1$  to  $num\_generations$ 
17     Evaluate(P)
18      $next\_P := \text{SelectTheBestOf}(P)$ 
19      $next\_P := next\_P + \text{Replicate}(P)$ 
20      $next\_P := next\_P + \text{Recombine}(P)$ 
21      $P := next\_P$ 
22   end for
23   Evaluate(P)
24   return SelectTheBestOf(P)
25 end EvoAlg

```

Fig. 2. Pseudocode of HIDER.

initializing the population, the for-loop repeats the evolutionary process a number of times which is determined by the parameter *num.generations*. In each iteration, the individuals of the population are evaluated according to a defined fitness function, thus each individual acquires a “goodness” (function *Evaluate*). The best individual of every generation is replicated to the next one (elitism). Later, a set of individuals are selected through the roulette wheel method and replicated to the next generation. Finally, another set of individuals are recombined and the offspring is included in the next generation. The selection of these individuals is also carried out by means of the roulette wheel. Once the loop finishes, the best individual of the last generation is returned.

3 Hybrid Coding

After studying other EA-based approaches proposed in the bibliography, we deduced that a combination of binary coding and real coding would be suitable for our problem. First, we analyzed two well-known systems: GABIL [8] and GIL [11]. These concept learners use binary coding, assigning a bit to each value of an attribute. A value of one, in a bit, implicates that the value is present, so that several bits could be active for the same attribute. This implies that binary coding is a good election for encoding symbolic attributes. Nevertheless, it is not suitable for continuous attributes, because the size of their alphabet is very large (theoretically infinite) and this aspect does not allow a complete search. Non-binary codings have been widely researched in literature [3,4,12,

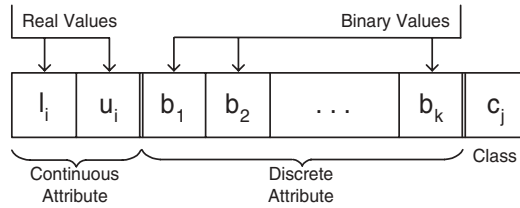


Fig. 3. An hybrid individual with a continuous attribute and a discrete one.

21,22]. These jobs show that real coding is more appropriate than binary in continuous domains.

Hybrid coding tries to profit from the advantages of binary and real coding. Thus, it mixes both codings by using the real to represent the boundaries of the intervals in continuous domains, and the binary to encode the discrete attributes. Figure 3 explains the representation for continuous and discrete attributes in hybrid coding. The continuous attribute is encoded by two real values being the lower (l_i) and the upper bound (u_i) of the interval that the rule establishes for such attribute. The discrete attribute is represented by a number k of bits, where each of them denotes the presence or absence of a discrete value in the condition of the rule. The last value is the class, which is not usually encoded. Each class is represented as a integer, so that for a number n of classes, the value c_j will belong to the set $\{0, 1, 2, \dots, n-1\}$.

With regard to the genetic operators, crossover and mutation for hybrid coding are described in detail in [2]. The crossover for continuous attributes is an extension of Radcliffe's [18] to parents coded as intervals. If two parents, a and b , establish the intervals $[l_i^a, u_i^a]$ and $[l_i^b, u_i^b]$ respectively for the attribute a_i , then the interval of the offspring ($[l_i^o, u_i^o]$) fulfils expressions in Equation 2. In case of discrete attributes, the uniform crossover is carried out [19].

$$\begin{aligned} l_i^o &\in [\min(l_i^a, l_i^b), \max(l_i^a, l_i^b)] \\ u_i^o &\in [\min(u_i^a, u_i^b), \max(u_i^a, u_i^b)] \end{aligned} \quad (2)$$

In order to apply the mutation to a continuous attribute, a gene (g_i) is randomly selected. Such gene can be the lower bound (l_i) or the upper (u_i) bound of the interval. The mutation consist in replacing the gene g_i with $g_i \pm \delta$, where δ is the smaller HOEM (Heterogeneous Overlap-Euclidean Metric [23]) between two examples of the training dataset. Thus, an interval is expanded ($l_i - \delta$ or $u_i + \delta$) or contracted ($l_i + \delta$ or $u_i - \delta$). When the attribute is discrete, the mutation changes the gene (a bit) depending on the discrete-value mutation probability, so that some values are included or excluded from the condition.

4 Natural Coding

The ideal coding must fulfil the following properties: completeness, coherence, uniformity (uniqueness), simplicity, locality, consistency and minimality. To find

Table 1. Coding for a continuous attribute.

Cutpoints	2.5	3.9	4.7	5.0	6.2
1.4	1 $\equiv [1.4, 2.5]$	2 $\equiv [1.4, 3.9]$	3 $\equiv [1.4, 4.7]$	4 $\equiv [1.4, 5.0]$	5 $\equiv [1.4, 6.2]$
2.5	—	7 $\equiv [2.5, 3.9]$	8 $\equiv [2.5, 4.7]$	9 $\equiv [2.5, 5.0]$	10 $\equiv [2.5, 6.2]$
3.9	—	—	13 $\equiv [3.9, 4.7]$	14 $\equiv [3.9, 5.0]$	15 $\equiv [3.9, 6.2]$
4.7	—	—	—	19 $\equiv [4.7, 5.0]$	20 $\equiv [4.7, 6.2]$
5.0	—	—	—	—	25 $\equiv [5.0, 6.2]$

a suitable representation is very difficult but to find one that fulfils every aforementioned property is practically impossible. Thus, we try to design a coding that keeps at least two well-known principles proposed by Goldberg in [10]: the principle of meaningful building blocks and the principle of minimal alphabets.

After analyzing the problem, we developed a coding, named “natural” [1], which only uses natural numbers to represent the set of values that can take part of the decision rules, independently of the kind of attributes (continuous or discrete). In particular, natural coding encode each condition of a rule with only one gene that is a natural number. Thus, this coding fulfils two aforementioned properties: uniqueness (every individual has an unique representation) and minimality (the length of coding must be as short as possible). Although continuous and discrete values are encoded as natural numbers, the meaning of this number is different in both cases.

As in regards to continuous attributes, a method that calculates the cutpoints in range of values for each continuous attribute is applied. The cutpoints are the possible bounds of intervals that a condition can set. In principle, these cutpoints can all be of values the attribute takes in the training dataset. However, this choice generates too many cutpoints. In order to reduce the number of cutpoints, we apply the method named USD [9]. This method produces a set of bounds that maximize the goodness of the possible intervals. Such an aspect has a favorable influence on the rules that HIDER* is going to obtain later. Once the cutpoints are calculated, a natural number is assigned to each possible combination of bounds, so every possible interval is represented by a natural number. Table 1 shows an example of coding for a continuous attribute where the set of cutpoints generated by USD is {1.4, 2.5, 3.9, 4.7, 5.0, 6.2}.

In general, if k is the number of cutpoints, the rows are labeled from the first one to $(k - 1)^{th}$ and the column are labeled from the second one to k^{th} cutpoint. Thus, each element of the table is a natural number that encodes the interval defined by its row and column.

In case the attributes are discrete, the natural coding is obtained from a binary coding similar to that used in GABIL and GIL. In decision rules, a condition can establish a set of discrete values that the attribute must take to classify an example. Each gene that represents a condition for a discrete attribute is also a natural number. The binary representation of this gene is a number where there is a bit for each different discrete value that an attribute can take. Thus, if Ω is the set of possible values for a discrete attribute, then the natural number that encodes the corresponding condition of the rule is in interval $[0, 2^{|\Omega|} - 1]$. If a value is included in the condition, its corresponding bit

Table 2. Coding for a discrete attribute.

Discrete values					Natural
white	red	green	blue	black	Coding
0	0	0	0	0	—
0	0	0	0	1	1
0	0	0	1	0	2
0	0	0	1	1	3
⋮	⋮	⋮	⋮	⋮	⋮
0	1	0	1	1	11
⋮	⋮	⋮	⋮	⋮	⋮
1	1	1	1	0	30
1	1	1	1	1	31

is equal to 1, otherwise it is 0. The natural coding for this gene is the conversion of the binary number into a natural one. Table 2 shows an example for a discrete attribute with five different values: *white*, *red*, *green*, *blue* and *black*.

The mutation and crossover operators are described in detail in [1]. For continuous attributes, mutation is a shift (up, down, left or right) in the coding table (see Table 1). Such shift mutates the gene changing the bounds of the interval. The crossover consists in calculating the intersection of the rows and columns of both parents, so that the obtained set of natural numbers is the offspring. For discrete attributes, the mutation is a transformation of the value that encodes the gene, by changing some bits of the binary code associated with the natural number. The crossover is based on the mutation. Each gene that take part in the crossover provides a set of candidates obtained from its possible mutations joined to itself. The offspring of two genes is a random selection of numbers from the intersection of both sets of candidates. Although the natural operators seem to be very complex processes, in fact, they are very simple algebraic operations. They do not imply any conversion between binary and natural numbers and do not need any operation with arrays. As shown in [1], the natural operators are algebraic operations with a low computational cost.

Figure 4 illustrates the differences between natural and hybrid coding. It shows two individuals that encode the same rule. The attribute AT_1 is continuous whereas AT_2 is discrete. Both attributes are encoded according to Tables 1 and 2.

We can see that natural coding is simpler, since the hybrid needs eight genes to code the rule whereas the natural encodes it with only three genes. Natural coding minimizes the size of individuals, assigning only one gene for each attribute. Hybrid uses two genes for continuous attributes and $-\Omega-$ for discrete, where $-\Omega-$ is the number of different values that an attribute can take.

5 Results

In order to show the quality of natural coding, we applied HIDER and HIDER* to a set of databases from UCI Repository [5]. As we mentioned in Section 2, both tools use the same EA, although HIDER uses hybrid coding, to the contrary of

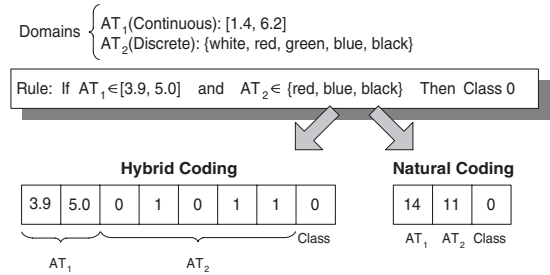


Fig. 4. Hybrid individual vs. Natural individual.

Table 3. Parameter of HIDER.

Parameter	HIDER (hybrid)	HIDER* (natural)
Population Size	100	70
Number of Generations	300	100
Replication	20%	20%
Recombination	80%	80%
Individual Mutation Probability	0.5	0.5
Gen Mutation Probability	$\frac{1}{\ attributes\ }$	$\frac{1}{\ attributes\ }$
Discrete-value Mutation Probability	$\frac{1}{\ values\ }$	$\frac{1}{\ values\ }$

HIDER* which uses natural coding. Both were run with the same crossover and mutation parameters, but with a different number of individuals and generations. Table 3 shows the parameters used in each case. The lower values that HIDER* needs for the population size and number of generations are due to the reduction of the search space by natural coding. Thus, HIDER* used only 100 generations and 70 individuals to obtain similar numbers to those obtained by HIDER, which needed 300 and 100 respectively (see Table 4).

The databases used in the experiments were 16, some of which contain only continuous attributes, others contain only discrete attributes and the remainder include both types of attributes. Thus, we can compare the behaviour of natural and hybrid coding with both types of attributes. To measure the performance of each method, a 10-fold cross-validation was achieved with each database. The values that represent the performance are the error rate (ER) and the number of rules (NR) obtained. The ER is the average number of misclassified examples expressed as a percentage and the NR is the average number of rules for the 10-fold cross-validation. The algorithms were run on the same training sets and the knowledge structures tested using the same test sets, so the results were comparable. With these conditions and the aforementioned parameters for the EAs, the results obtained by applying both codings are given in Table 4.

In Table 4, the first column shows the databases used in the experiments; the next two columns give the ER and NR obtained respectively by HIDER with hybrid coding for each database. Likewise, the fourth and fifth columns give the ER and NR for HIDER* with natural coding. The last two columns show a

Table 4. Comparing HIDER and HIDER*.

Database	HIDER (hybrid)		HIDER* (natural)		Improvement	
	ER	NR	ER	NR	ϵ_{er}	ϵ_{nr}
breast-c	4.3	2.6	4.06	2	1.06	1.3
bupa	35.7	11.3	37.35	4.2	0.96	2.69
cleve	20.5	7.9	25.33	5.9	0.81	1.34
german	29.1	13.3	27.4	8	1.06	1.66
glass	29.4	19	35.24	11.7	0.83	1.62
heart	22.3	9.2	21.85	4.3	1.02	2.14
hepatiti	19.4	4.5	16.67	3.7	1.16	1.22
horse-co	17.6	6	20	11.1	0.88	0.54
Iris	3.3	4.8	3.33	3.2	0.99	1.5
Lenses	25	6.5	25	4.5	1	1.44
mushroom	0.8	3.1	1.18	3.5	0.68	0.89
pima	25.9	16.6	25.66	5.1	1.01	3.25
vehicle	30.6	36.2	33.81	19.7	0.91	1.84
vote	6.4	4	4.42	2.2	1.45	1.82
wine	3.9	3.3	8.82	5.6	0.44	0.59
zoo	8	7.2	4	7.9	2	0.91
Average	17,64	9,72	18,38	6,41	1,02	1,55

measure of improvement for the error rate (ϵ_{er}) and the number of rules (ϵ_{nr}). The ϵ_{er} coefficient was calculated by dividing the error rate for HIDER by the corresponding error rate for HIDER*. The same operation was carried out to obtain ϵ_{nr} , but by using the number of rules for both tools. Finally, the last row shows the average results for each column. As we can observe, HIDER* does not attain a reduction in ER for 8 out of 16 datasets. Nevertheless, on average, it improves on HIDER, although this improvement is very small (2%). As regards to number of rules, the results are more significant, since HIDER* obtains a smaller number of rules in 12 out of 16 cases, with an average improvement of 55%.

Although the results show that HIDER* has a better performance, we must not forget that those numbers were obtained using a smaller number of generations and individuals of a genetic population. In particular, HIDER* needed one third of the generations and fewer than three quarters of the population size invested by HIDER. The reduction of the search space that natural coding gets, allows HIDER* to obtain these results with such parameters.

In Table 5, the length of individuals of the genetic population for the hybrid and natural coding are shown. The first column contains the databases. The second shows the number of continuous attributes (NC), if any, for each database. Likewise, the next column gives the number of discrete attributes (ND) along with the total number of different values in brackets (NV). The column labeled as “Hybrid” gives the length of individuals (number of genes) with hybrid coding. Finally, the last one shows the length of individuals encoded with natural coding. These lengths were calculated easily from the second and third column. Hybrid

Table 5. Comparing length of Hybrid and Natural Individuals.

Database	NC	ND (NV)	Hybrid	Natural
breast-c	9	-	18	9
bupa	6	-	12	6
cleve	6	7 (19)	31	13
german	7	13 (54)	68	20
glass	9	-	18	9
heart	13	-	26	13
hepatiti	6	13 (26)	38	19
horse-co	7	15 (55)	69	22
Iris	4	-	8	4
Lenses	-	4 (9)	9	4
Mushroom	-	22 (117)	117	22
Pma	8	-	16	8
tic-tac-	9	-	18	9
vehicle	18	-	36	18
vote	-	16 (48)	48	16
wine	13	-	26	13
zoo	-	16 (36)	36	16
Average			34,94	13

coding uses two genes to represent continuous attributes and a number k of genes for discrete ones, k being the number of different values of the attribute. On the other hand, natural coding uses only one gene for each attribute, regardless of its type (continuous or discrete). Thus, the length for hybrid individuals is $2 \times NC + NV$, whereas for natural individuals is $NC + NV$. As we can see, natural coding noticeably decreases the length of individuals. On average, it obtains a reduction greater than 63% regarding the hybrid individuals.

The most important contribution of natural coding is the decrease in size of the search space. This aspect is considered only when the dataset includes continuous attributes. For discrete attributes, natural coding does not reduce the space, since it is based on binary coding. Thus, if Ω_d is the set of values for a discrete attribute, then the size of the search space is $2^{|\Omega_d|}$ for both codings. However, hybrid coding represents continuous attributes with real coding, i.e the genes can take values in an infinite domain. Thus, when there are continuous attributes, the search space with hybrid individuals is also infinite. On the contrary, genes that encode continuous attributes with natural coding can only take values belonging to a finite set of natural numbers. Let N be the number of cutpoints that USD algorithm obtains for an attribute a_i . Let Ω_c be the set of natural numbers that the gene g_i can take according to the cutpoints. Then, the size of the search space for the attribute a_i is the number of elements of Ω_c , that it is given by Equation 3.

$$|\Omega_c| = \binom{N}{2} = \frac{N(N-1)}{2} \tag{3}$$

Thus, the total size of the search space for a dataset with a number M of attributes using hybrid or natural coding is

$$S = \prod_{i=1}^M |\Omega_i| \quad (4)$$

where $|\Omega_i|$ is the number of possible values of the attribute a_i . If there are any continuous a_i , such size is infinite for hybrid coding but finite for natural coding.

6 Conclusions

In this paper, natural coding for EA-based decision rules generation is described and tested. This coding transforms the attributes domain (continuous and discrete) in a finite set of natural numbers. Such conversion allows a decrease in size of search space in such a way that the algorithm converges more quickly. Furthermore, natural coding encodes each attribute with only one gene, reducing considerably the length of individuals. The quality of this coding has been tested by applying the same evolutionary learning tool (HIDER) with natural and hybrid coding and by comparing the obtained sets of rules for a set of databases from the UCI Repository. The algorithms with hybrid coding needed 300 generations and a population with 100 individuals in order to obtain a suitable model. Nevertheless, the use of natural coding obtained models with the same accuracy, but a lesser number of rules. Although this is a noteworthy aspect, the most important fact is that such results were obtained with only 70 individuals per population and a number of generations as little as 100. The faster convergence of the algorithm is due to the decreasing in size of search space, since natural coding converts the infinite domain of the continuous attributes to a finite set of natural numbers.

References

1. Jesus S. Aguilar-Ruiz and J.C. Riquelme. Improving the Evolutionary Coding for Machine Learning Tasks. *European Conference on Artificial Intelligence, ECAI'02*, pp. 173–177, IOS Press, Lyon, France, 2002.
2. Jesus S. Aguilar-Ruiz, J.C. Riquelme and M. Toro. Evolutionary Learning of Hierarchical Decision Rules. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, ISSN: 1083-4419, vol. 33, no. 2, pp. 324–331, 2003.
3. J. Antonisse. A new interpretation of schema notation that overturns the binary encoding constraint. In: *Third International Conference on Genetic Algorithms*, pp. 86–97, 1989.
4. S. Bhattacharyya and G. Koehler. An analysis of non-binary genetic algorithms with cardinality 2^v . *Complex Systems* 8, pp. 227–256, 1994.
5. C. Blake and E. K. Merz. UCI repository of machine learning databases, 1998.
6. P. Clark and R. Boswell. The cn2 induction algorithm. *Machine Learning*, vol. 3, no. 4, pp. 261–283, 1989.

7. P. Clark and R. Boswell. Rule induction with cn2: Some recent improvements. *Machine Learning: Proceedings of EWSL-91*, pp. 51–163, 1991.
8. K. A. DeJong, W. M. Spears, and D. F. Gordon. Using genetic algorithms for concept learning. *Machine Learning*, 1(13):161–188, 1993.
9. R. Giráldez, Jesus S. Aguilar-Ruiz and J.C. Riquelme. Discretization Oriented to Decision Rule Generation *International Conference on Knowledge-Based Intelligent Information & Engineering Systems*, KES'02 IOS Press, pp. 275–279 September 16–18, Crema, Italy, 2002.
10. D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, 1989.
11. C. Z. Janikow. A knowledge-intensive genetic algorithm for supervised learning. *Machine Learning*, 1(13):169–228, 1993.
12. G. Koehler, S. Bhattacharyya and M. Vose. General cardinality genetic algorithms. *Evolutionary Computation* 5(4), 439–459, 1998.
13. R. S. Michalski, I. Mozetic, J. Hong, and N. Lavrac. The aq15 inductive learning system: An overview and experiments. In *Proceedings of the American Association for Artificial Intelligence Conference (AAAI)*, 1986.
14. T. Mitchell. *Machine Learning*. McGraw Hill, 1997.
15. S. K. Murthy, S. Kasif, and S. Salzberg. A system for induction of oblique decision trees. *Journal of Artificial Intelligence Research*, 1994.
16. J. R. Quinlan. *C4.5: Programs for Machine Learning*, Morgan Kaufmann, San Mateo, California, 1993.
17. J. R. Quinlan. See5.0 (<http://www.rulequest.com>), 1998–2001.
18. N. J. Radcliffe. Genetic Neural Networks on MIMD Computers. Ph. d., University of Edinburgh, 1990.
19. G. Syswerda. Uniform crossover in genetic algorithms. In: *Proceedings of the Third International Conference on Genetic Algorithms*. pp. 2–9, 1989.
20. G. Venturini. SIA: a supervised inductive algorithm with genetic search for learning attributes based concepts. In *Proceedings of European Conference on Machine Learning*, pp. 281–296, 1993.
21. M. Vose and A. Wright. The simple genetic algorithm and the Walsh transform: Part I, theory. *Evolutionary Computation* 6(3), pp. 253–273, 1998.
22. M. Vose and A. Wright. The simple genetic algorithm and the Walsh transform: Part II, the inverse. *Evolutionary Computation* 6(3), pp. 275–289, 1998.
23. D. R. Wilson and T. R. Martinez. Improved Heterogeneous Distance Functions. *Journal of Artificial Intelligence Research* 6(1), pp. 1–34, 1997.

Hybridization of Estimation of Distribution Algorithms with a Repair Method for Solving Constraint Satisfaction Problems

Hisashi Handa

Okayama University, Tsushima-Naka 3-1-1,
Okayama 700-8530, JAPAN,

handa@sdc.it.okayama-u.ac.jp,

<http://www.sdc.it.okayama-u.ac.jp/~handa/index-e.html>

Abstract. Estimation of Distribution Algorithms (EDAs) are new promising methods in the field of genetic and evolutionary algorithms. In the case of conventional Genetic and Evolutionary Algorithm studies to apply Constraint Satisfaction Problems (CSPs), it is well-known that the incorporation of the domain knowledge in the CSPs is quite effective. In this paper, we propose a hybridization method (memetic algorithm) of Estimation of Distribution Algorithms with a repair method. Experimental results on general CSPs tell us the effectiveness of the proposed method.

1 Introduction

As the scale and the complexity of engineering problems increases, the distributed processing approaches inspired by the constraints-oriented problem solving methods are now receiving attentions. The notion of Constraint Satisfaction Problems (CSPs) provides us general framework adaptable to a wide variety of problems in various fields [1,2]. The CSPs are a problem class which consists of variables and constraints on the variables. Solving problems by using Constraint-oriented approach is that, first, we have to describe constraints which should satisfy between elements in the target environments. Then, we employ “constraint satisfaction problem solver (CSP solver),” to find satisfied solutions of the described problem instance such that all constraints in the problem are satisfied. Recently, CSP solvers from genetic and evolutionary algorithms have been broadly studied by many researchers [2]-[9]. These studies showed that the uses of the domain knowledge in the CSPs, that is, the hybridization with local search method (repair method) based upon the notion of Min-Conflict Hill Climbing (MCHC) [10], and the utilization of constraint networks [6], are quite effective.

Estimation of Distribution Algorithms (EDAs) are new promising methods in the field of genetic and evolutionary algorithms [11,12]. The EDAs employ the probabilistic model, which is constituted by a database containing the genetic information of the selected individuals in the previous generation, to yield a

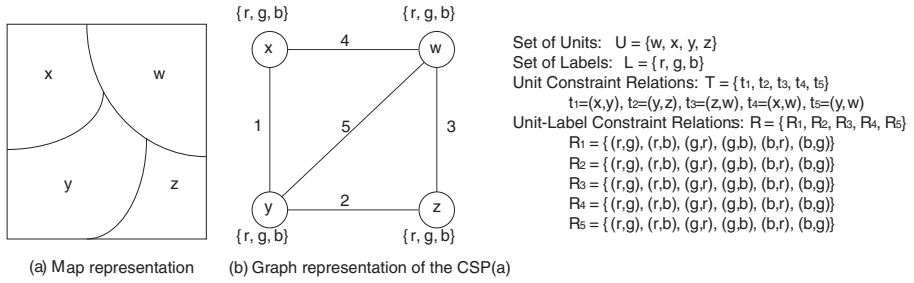


Fig. 1. An example of CSP: graph coloring problem

new population. Hence, in the EDAs, Genetic operations such like crossover and mutation are not adopted.

In this paper, we propose a new evolutionary constraints satisfaction problem solver incorporating a repair method into the EDA. Moreover, we also introduce a manner which incorporates the knowledge of the constraint network to the Bayesian Network structure search.

Related works are described as follows: Tsang wrote comprehensive text book about CSPs which is also written about genetic approach for solving CSPs [1]. Eiben summarized how to solve several classes of CSPs by using GAs in [3,4]. Riff proposed a fitness function and genetic operators for solving CSPs effectively, which are utilizing the knowledge with regard to the constraint network [5,6]. Coevolutionary Evolutionary Computations were often adopted to solve CSPs [2,9]. With respect to the EDAs, Larrañaga and Lozano edited a comprehensive book of the EDAs[12]. EDAs in section 3 are introduced by referring to this textbook. Genetic algorithms with local search methods are often called “memetic algorithms,” and have been studied by many researcher for the last decade [13].

In the next section, the basics of the CSPs will be described. Then we will briefly introduce three kinds of EDAs in section 3, which are employed for our experiments. Section 4 will introduce the proposed method. Then, experiments on general Constraint Satisfaction Problems will be carried out for conventional GEA approach, EDAs introduced in section 3 and the proposed method. Section 6 will conclude this paper.

2 Constraint Satisfaction Problems

2.1 Formulation

Constraint Satisfaction Problems (CSPs) are a class of problems which consists of variables and constraints on the variables [1]. In addition, the class of CSPs in which each constraint in the problems is related only to two variables is called binary CSPs. In this paper, we treat a class of discrete binary CSPs, where the

```

Procedure Min-Conflict Hill Climbing
begin
   $e(i) \leftarrow$  evaluate each variable  $i$  in the current solution
  until Stopping criteria is hold
     $i^* \leftarrow$  Select a variable with the worst evaluation
    Re-evaluate  $e(i^*)$  for all labels in the variable  $i^*$ 
    Select a label in the variable  $i^*$  with the best evaluation
    Modify the current solution to the selected label
    Re-evaluate the current solution with respect to the modification
  end
end

```

Fig. 2. Pseudo code of Min-Conflict Hill Climbing

word ‘discrete’ means that each variable is associated with a finite set of discrete values (labels) that are candidate values of the variable. An example of the graph coloring problem [10], a binary CSP which is one of the benchmark CSP is delineated in Fig. 1. As depicted in the figure, CSPs are defined by (U, L, T, R) . U , L , T and R denote a set of units (variables), a set of labels (values), unit constraint relations and unit-label constraint relations, respectively.

In this paper, we use two indices, tightness and density to analyse the difficulty of CSPs [1]. The tightness of an edge ij is given as the ratio of the number of satisfying 2-compound labels (in unit-label constraint relations) on the edge ij over the number of all 2-compound labels on the edge ij . Furthermore, the tightness of a problem is given by the averaged value of tightness of the edges in the problem. The density of a problem indicates the proportion of constraint relations that actually exist between any pair of nodes. Furthermore, the number of 2-compound labels on the edge XY is the same as the product of the number of labels on each nodes, that is, 3×3 .

2.2 Min-Conflict Hill Climbing

This local search method, often called heuristic repair method, is adopted not only from genetic and evolutionary algorithms but also from approximation algorithms for solving CSPs. The procedure of the MCHC is described in Fig. 2: The MCHC begins with a given solution. First, the solution is evaluated. In that time, the number of constraint violations for each variable is memorized. Then, the variable which have the most constraint violations, i.e., the least evaluation, is chosen. If some variables tie with each other, the variable is randomly chosen among them. For the selected variable, all the labels are examined and evaluated. In similar to the variable selection, one of labels with the least constraint violations is chosen (randomly in the case of tie). Re-evaluation for above modification is carried out, and the process returns to the variable selection phase.

```

Procedure Estimation of Distribution Algorithm
begin
  initialize  $D_0$ 
  evaluate  $D_0$ 
  until Stopping criteria is hold
     $D_{l-1}^{Se} \leftarrow$  Select  $N$  individuals from  $D_{l-1}$ 
     $p_l(\mathbf{x}) \leftarrow$  Estimate the probabilistic model from  $D_{l-1}^{Se}$ 
     $D_l^{Se} \leftarrow$  Sampling  $M$  individuals from  $p_l(\mathbf{x})$ 
  end
end

```

Fig. 3. Pseudo code of Estimation of Distribution Algorithms

3 Brief Introduction of the Estimation of Distribution Algorithms

3.1 General Framework of EDAs

The Estimation of Distribution Algorithms are a class of evolutionary algorithms which adopt probabilistic models to reproduce the genetic information of the next generation, instead of conventional crossover and mutation operations. The probabilistic model is represented by conditional probability distributions for each variable (locus). This probabilistic model is estimated from the genetic information of selected individuals in the current generation. Hence, the pseudo-code of EDAs can be written as Fig. 3, where D_l , D_l^{Se} , and $p_l(\mathbf{x})$ indicate the set of individuals at l th generation, the set of selected individuals at $l - 1$ th generation, and estimated probabilistic model at l th generation, respectively [12]. The representation and estimation methods of the probabilistic model are devised by each algorithm. The following subsections will overview some EDAs. For a more thorough overview, see [12,14].

3.2 UMDA

UMDA (Univariate Marginal Distribution Algorithm) was introduced by Mühlenbein [12]. As indicated by its name, the variables of the probabilistic model in this algorithm is assumed to be independent from other variables. That is, the probability distribution $p_l(\mathbf{x})$ is denoted by a product of univariate marginal distributions, i.e.,

$$p_l(\mathbf{x}) = \prod_{i=1}^n p_l(x_i),$$

where $p_l(x_i)$ denotes the univariate marginal distribution $X_i = x_i$ at a variable X_i at generation l . This univariate marginal distribution is estimated from marginal frequencies:

$$p_l(x_i) = \frac{\text{the number of solutions, where } X_i = x_i \text{ in the selected individuals}}{N},$$

where N denotes the number of selected individuals, which is fixed in advance.

3.3 MIMIC

De Bonet *et al.* proposed MIMIC [12,17], a kind of EDAs whose probabilistic model is constructed with bivariate dependency such like COMIT [18]. While the COMIT generates a tree as dependency graph, the probabilistic model of the MIMIC is based upon a permutation π .

$$p_l(\mathbf{x}) = \prod_{j=1}^{n-1} p_l(x_{i_{n-j}} | x_{i_{n-j+1}}) \cdot p_l(x_{i_n}),$$

where the permutation π is represented by $(i_1, i_2, \dots, i_n)$. The permutation π is decided in each generation such that the following Kullback-Leibler divergence $H_l^\pi(\mathbf{x})$ is minimized:

$$H_l^\pi(\mathbf{x}) = h_l(X_{i_n}) + \sum_{j=1}^{n-1} h_l(X_{i_j} | Y_{i_{j+1}}),$$

where $h_l(X) = -\sum_x p(X=x) \log p(X=x)$, and $h_l(X|Y) = -\sum_x \sum_y p(Y=y) p(X=x|Y=y) \log p(X=x|Y=y)$. However, such minimization is NP so that this minimization is carried out by greedy search.

3.4 EBNA

Like BOA and LFDA [15,16], The EBNA (Estimation of Bayesian Networks Algorithms) adopts Bayesian Network (BN) as the probabilistic model, which is proposed by Larrañaga *et al.* [12,19]. They proposed several kinds of EBNA, such as EBNA_{PC}, EBNA_{K2+pen}, EBNA_{BIC}, and so on. Here, we introduce only EBNA_{BIC} used in our experiments. EBNA_{BIC} searches for the better structure of BN by using search+score method. In the case of the EBNA_{BIC}, scoring is achieved by penalized maximum likelihood $BIC(S, D)$ for a given structure S and a dataset D , called Bayesian Information Criteria, denoted by the following equation:

$$BIC(S, D) = \sum_{i=1}^n \sum_{j=1}^{q_i} \sum_{k=1}^{r_i} N_{ijk} \log \frac{N_{ijk}}{N_{ij}} - \frac{1}{2} \log N \sum_{i=1}^n q_i(r_i - 1),$$

where the structure S is represented by Direct Acyclic Graphs, n is the number of variables of the Bayesian Network, r_i is the number of different values that variable X_i can take, q_i is the number of different values that the parent variables of X_i in the structure S can take, N_{ij} is the number of individuals in D in which the parent variables of variable X_i take their j^{th} value, and N_{ijk} is the number of individuals in D in which variable X_i takes its k^{th} value and the parent variables of the variable i take their j^{th} value [12]. As the search method in the EBNA_{BIC}, an arc-based local search is adopted due to the NP property of searching the best structure for BNs.

```

Procedure EDA for CSP
begin
  initialize  $D_0$ 
  evaluate  $D_0$ 
  until Stopping criteria is hold
     $D_{l-1}^{Se} \leftarrow$  Select  $N$  individuals from  $D_{l-1}$ 
     $p_l(\mathbf{x}) \leftarrow$  Estimate the probabilistic model from  $D_{l-1}^{Se}$ 
     $D_l^{Se} \leftarrow$  Sample  $M$  individuals from  $p_l(\mathbf{x})$ 
     $D_l^{Se} \leftarrow$  carry out a Repair Algorithm to  $D_l^{Se}$ 
  end
end

```

Fig. 4. Pseudo code of EDA for CSP

4 The Proposed Method

As mentioned in the introduction of this paper, applying conventional GEA to solve CSPs have been studied by many researchers. According to the conclusions of their studies, the utilization of the CSP-specific knowledge, such as the topology of constraint networks, local constraint evaluation (sub-evaluation of individuals), and so on, yields the great improvement of the search ability of GEAs [3]-[9]. Hence, we propose a hybrid method of EDAs with repair method. Moreover, we also introduce a manner which incorporates the knowledge of the constraint network to the BN structure search.

As a repair method for the proposed method, we employ asexual heuristic operator in H-GA proposed by Eiben *et al.* [3]. The asexual heuristic operator uses the notion of the MCHC described in Fig. 2 to operate the genetic information of individuals. The number of applying the MCHC is pre-defined, that is, $1/4$ of the string length. The timing to apply this operation to the EDA population is after sampling new individuals from the estimated (joint) probability distribution as denoted in the Fig. 4.

Furthermore, in order to reduce the computational effort, the variable candidate selection guided by constraint networks is introduced to bivariate or multivariate dependent algorithms. That mechanism is quite simple: arcs which do not have constraint-relations, which can be referred by the constraint network, do not calculate its indices to construct the probabilistic model, such like $H_l^\pi(\mathbf{x})$ and $BIC(S, D)$. However, it saves a large amount of computational time since the calculation of conditional probabilities takes much time and appears for corresponding candidate structures.

5 Experimental Results

In this paper, we carry out several experiments based on various general CSPs that are generated randomly for a wide variety of “density” and “tightness” of constraint conditions in the CSPs that are the basic measures of characterizing CSPs and are described in section 2. The general CSPs are randomly generated

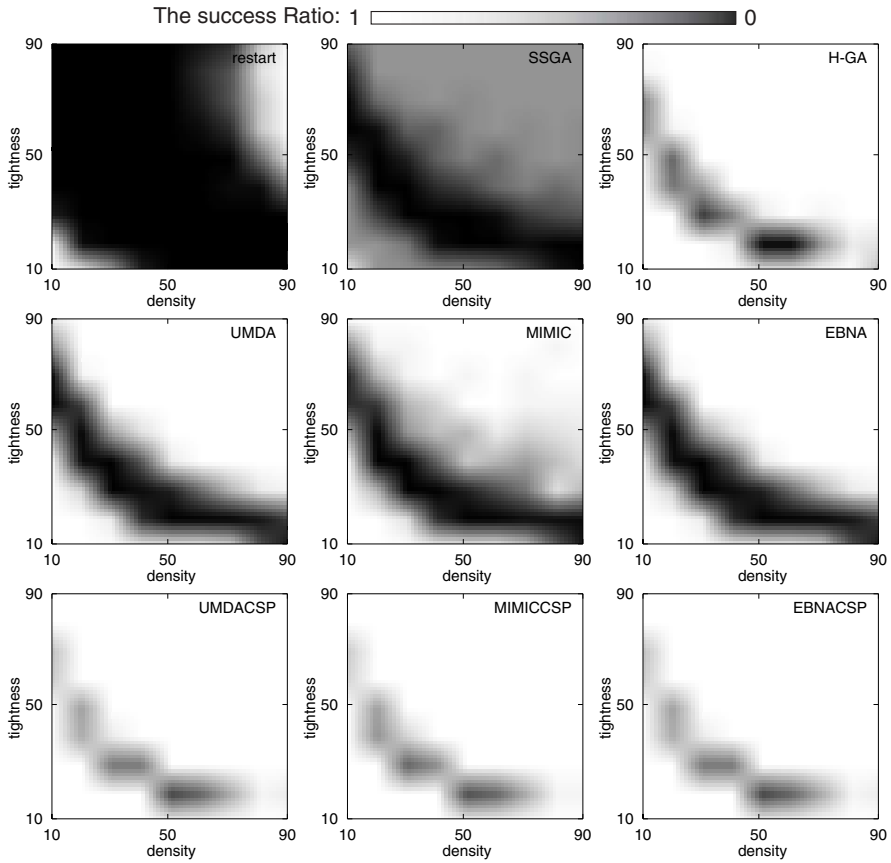


Fig. 5. Experimental results on general CSP, the ratio of success examination: results with higher success ratio tend to be brighter, and results with lower success ratio tend to be darker; The upper row is for conventional methods, restart method with MCHC, Steady-State Genetic Algorithms, and H-GA; The middle row is for EDAs, UMDA, MIMIC, and EBNA; The lower row is for the proposed methods, UMDACSP, MIMICCSP, and EBNACSP.

as follows: First, specify the tightness and density in the sense in section 2. Next, for all combination of two indices, decide whether unit constraint relation is set to each of the pairs of variables by taking account of the value of density. Finally, for all unit constraint relations, the number of the unit-label constraint relationships is set to be directly proportional to the tightness.

In order to solve the general CSPs, i.e., to find a solution such that it has no constraint violation, we introduce following fitness function which is generally adopted:

$$1 - N_{CV}/N_{MC},$$

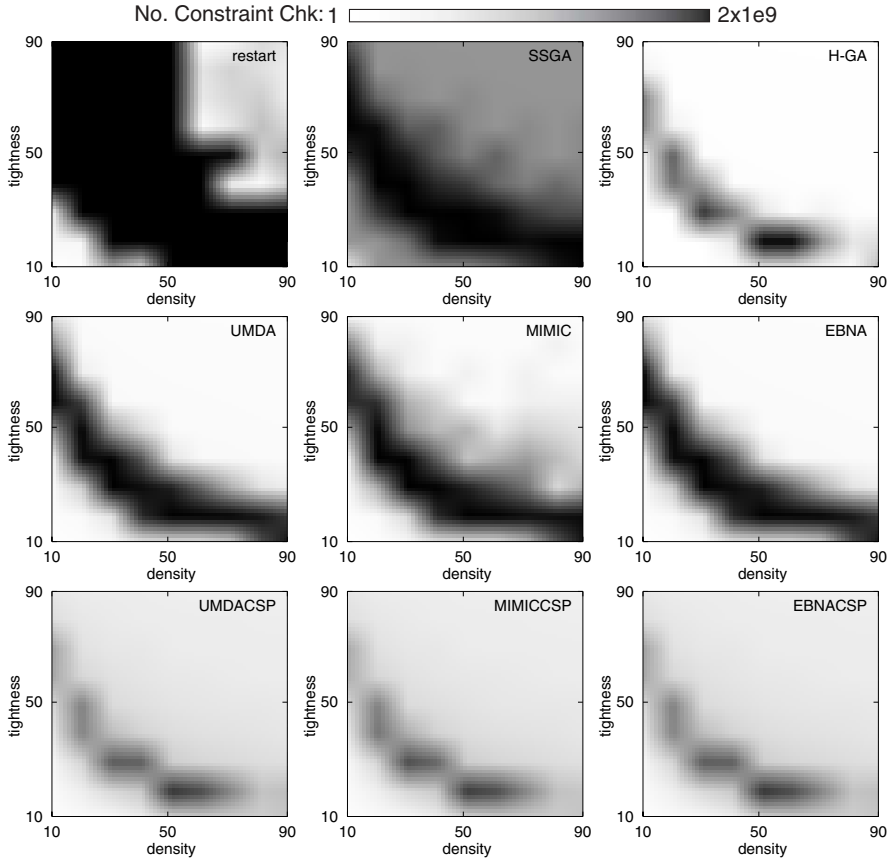


Fig. 6. Experimental results on general CSP, the number of constraint checks until finding satisfiable solutions when satisfiable solutions are found within 2 billion constraint checks: results with faster discovers tend to be brighter, and results with later discovers tend to be darker; The allocation of each graph is the same as the previous figure.

where N_{CV} and N_{MC} indicate the number of constraint violations in a certain individual, and the number of possible constraint relations in a problem with n variables, i.e., $n(n-1)/2$ in the case of binary CSPs, respectively. The coding method adopted in this paper is a naive one such that each variable in a problem instance is corresponding to each gene. That is, each label associated with each variable is directly represented as each allele at corresponding locus.

First, we compare the proposed methods for UMDA, MIMIC, and EBNA, i.e., UMDACSP, MIMICCSP, and EBNACSP, respectively, with various kinds of conventional method: restart method, steady-state GA, H-GA proposed by Eiben et al., UMDA, MIMIC, and EBNA [21]. Parameters for each algorithm are described as follows: The restart method is a (non-evolutionary) conventional

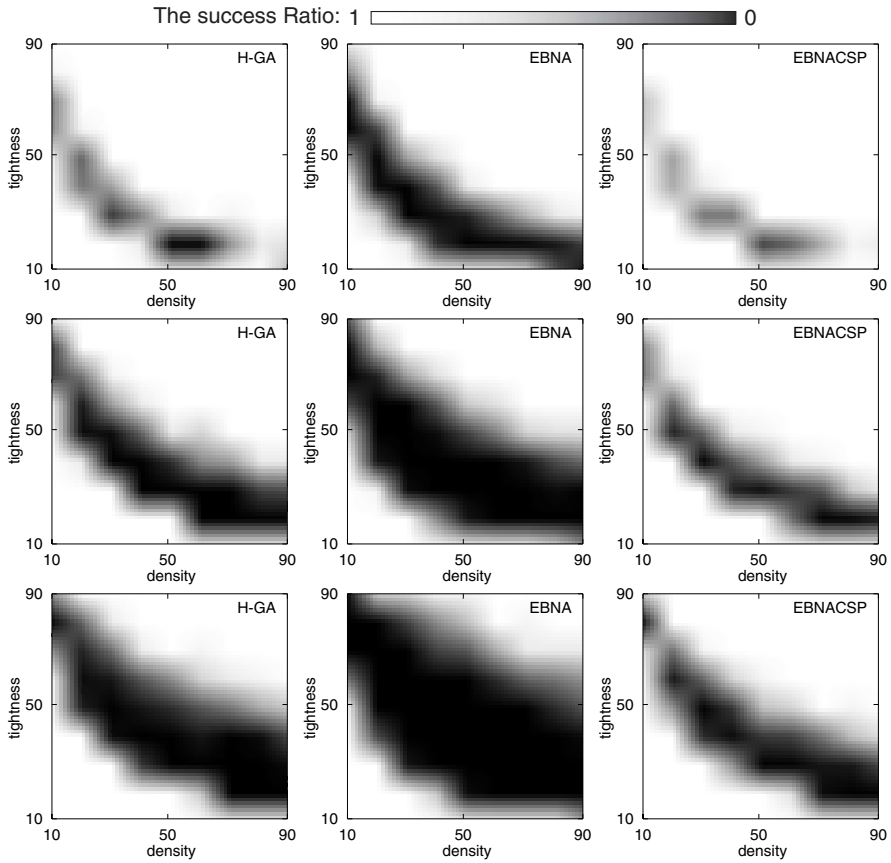


Fig. 7. The scalability of each algorithm, the ratio of success examination: the depiction manner of each graph is the same as Fig. 5; 40 variables with 10(UPPER ROW), 20(MIDDLE ROW), and 30(LOWER ROW) labels in each variable; H-GA(LEFT COL.), EBNA(MIDDLE COL.), and EBNACSP(RIGHT COL.)

CSP solver and employs the MCHC, described in Fig. 2, as Hill-Climber. If the MCHC fails the evaluation improvement of the solution 100 times succeedingly, a new initial solution is randomly generated again. The population size of SSGA and H-GA are set to be 200, which is a resultant of parameter tuning. The mutation probability of SSGA is set to be 0.025 which is identical to the $1 / (\text{string length})$. We use the H-GA version 1 which shows the best performance in [3] to our experiment. The population size M of EDAs including the proposed methods is set to be 3000. The size N of selected individual used to estimate the probabilistic model is set to be 1000.

The experimental results for solving general CSPs are plotted in Fig. 5 and 6. The x and y axes of all graphs in these figures denote the values of density and tightness, respectively. All graphs in this paper are averaged results over

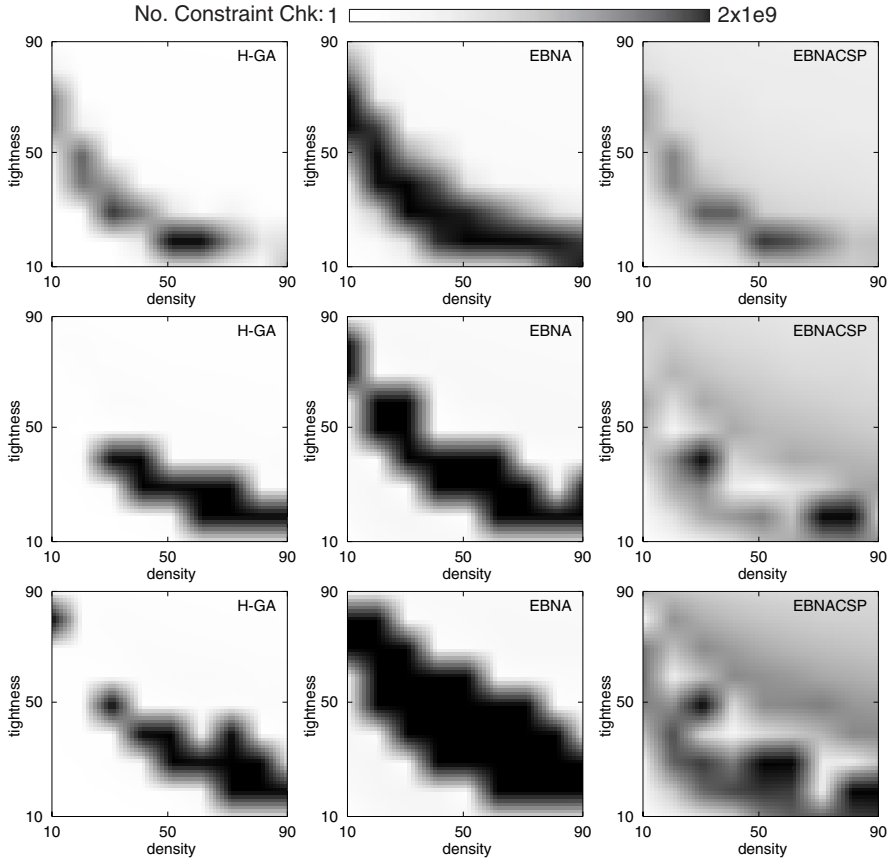


Fig. 8. The scalability of each algorithm, the number of constraint checks until finding satisfiable solutions when satisfiable solutions are found within 2 billion constraint checks: the depiction manner of each graph is the same as Fig. 6; 40 variables with 10(UPPER ROW), 20(MIDDLE ROW), and 30(LOWER ROW) labels in each variable; H-GA(LEFT COL.), EBNA(MIDDLE COL.), and EBNACSP(RIGHT COL.)

25 experiments for each couple of (density, and tightness), where density and tightness separately range from 10 to 90 with step size 10. We define “a success experiment” whether algorithms can find satisfiable solution until 2 billion constraint checks. That is, the ratio of success (Fig. 5) indicates how many success experiments exist over 25 experiments. If the success ratio is equal to 0, the number of constraint checks until finding satisfiable solution in the case of success experiments (Fig. 6) is set to be 2 billion. The colors in Fig. 5 and 6, such that favourable results tend to be brighter, and unfortunate results tend to be darker, indicate the success ratio and the number of constraint checks until finding satisfiable solutions. As shaded in all graphs, darker areas lie across the

couples of density and tightness. Such an area is called the “phase transition”, where it is difficult to solve the problem for any algorithms.

As depicted in Fig. 5 and Fig. 6, the restart method and the SSGA were not found satisfiable solution effectively. Almost all experiments of them were failed. We can confirm the role of GEAs and EDAs in these hybrid approaches since naive iteration cannot help us to find satisfiable solutions. The conventional EDAs outperform conventional GA (SSGA). By comparing H-GA with SSGA, we can confirm the effectiveness of the repair algorithm for CSPs since H-GA also adopts the steady-state selection. The proposed method outperform any other algorithms in the sense of the success ratio. However they have a drawback that it takes a large number of constraint checks even if it is easier problem instances for H-GA and the proposed methods, for problem instances in the phase transition area, the proposed method outperform the H-GA.

Next, we investigate the scalability of the H-GA, EBNA and EBNACSP in Fig. 7 and Fig. 8. In these figures, the domain size (the number of labels in each variables) of general CSPs with 40 variables change from 10 to 30. EBNACSP exhibit the best scalability in the sense of the success ratio (Fig. 7) among three algorithms. H-GA can quickly solve problem instances, but its success ratio is low.

6 Conclusions

In this paper, we proposed the hybrid method of EDAs with repair method for solving CSPs. As the same as the conventional GEAs, the repair method improve the search methods of EDAs dramatically. Moreover, the proposed method outperforms the restart method, GEAs with or w/o the repair method and conventional EDAs. Future work is (1) to reduce the number of constraint checks in the case of easier problems. That is, actually, the proposed method apply the repair method to all individuals. It yields great number of constraint checks. Furthermore, (2) another one is further incorporation of the domain knowledge in CSPs to estimation of the probabilistic model. The proposed method employed BIC as the score metric. However, the BIC does not take account into the local consistency of labels. We consider the local consistency of labels makes the score metric more effective.

References

1. E. Tsang: *Foundations of Constraint Satisfaction*, Academic Press, 1993
2. J. Paredis: Genetic State-Space Search for Constraint Optimization Problems, *Proceedings of the 13th International Joint Conference on Artificial Intelligence II* (1993) 967–972
3. A. E. Eiben *et al.*: Solving Constraint Satisfaction Problems Using Genetic Algorithms, *Proceedings of the 1st International Conference on Evolutionary Computation II* (1994) 542–547

4. A. E. Eiben *et al.*: How to Apply Genetic Algorithms to Constrained Problems, *Practical Handbook of Genetic Algorithms*, Lance Chambers Editors, CRC Press, Chapter 10 (1995) 307–365
5. M.-C. Riff: Using the knowledge of the Constraint Network to design an evolutionary algorithm that solves CSP, *Proceedings of the 3rd International Conference on Evolutionary Computation* (1996) 279–284
6. M.-C. Riff: Evolutionary Search guided by the Constraint Network to solve CSP, *Proceedings of the 4th International Conference on Evolutionary Computation* (1997) 337–342
7. E. Marchiori: Combining Constraint Processing and Genetic Algorithms for Constraint Satisfaction Problems, *Proceedings of the Seventh International Conference of Genetic Algorithm* (1997) 330–337
8. G. Dozier *et al.*: Solving constraint satisfaction problems using hybrid evolutionary search, *IEEE Transactions on Evolutionary Computation* **2**(1) (1998) 23–33
9. H. Handa *et al.*: Coevolutionary Genetic Algorithm for Constraint Satisfaction with a Genetic Repair Operator for Effective Schemata Formation, *Proceedings of the 1999 IEEE System Man and Cybernetics Conference III* (1999) 617–621
10. S. Minton *et al.*: Minimizing conflicts: a heuristic repair method for constraint satisfaction and scheduling problems, *Constraint-Based Reasoning*, E. C. Freuder and A. K. Mackworth Editors, The MIT Press (1994) 161–206,
11. H. Mühlenbein and G. Paaß: From Recombination of genes to the estimation of distributions I. Binary parameters. *Parallel Problem Solving from Nature - PPSN IV* (1996) 178–187
12. P. Larrañaga and J. A. Lozano Editors: *Estimation of Distribution Algorithms*, Kluwer Academic Publishers (2002)
13. Memetic Algorithms' Home Page,
http://www.ing.unlp.edu.ar/cetad/mos/memetic_home.html
14. M. Pelikan: Bayesian optimization algorithm: From single level to hierarchy, Ph.D. thesis, University of Illinois at Urbana-Champaign, Urbana, IL. Also ILLIGAL Report No. 2002023 (2002)
15. M. Pelikan *et al.*: BOA: The Bayesian optimization algorithm, *Proceedings of the Genetic and Evolutionary Computation Conference* **1** (1999) 525–532
16. H. Mühlenbein and T. Mahnig: FDA - a scalable evolutionary algorithms for the optimization of additively decomposed functions, *Evolutionary Computation* **7**(4) (1999) 353–376
17. J. S. De Bonet *et al.*: MIMIC: Finding optima by estimating probability densities, *Advances in Neural Information Processing Systems* **9** (1996)
18. S. Baluja: Using a priori knowledge to create probabilistic models for optimization *International J. of Approximate Reasoning*, **31**(3) (2002) 193–220
19. P. Larrañaga *et al.*: Combinatorial Optimization by Learning and Simulation of Bayesian, *Uncertainty in Artificial Intelligence, Proceedings of the Sixteenth Conference* (2000) 343–352
20. E. Bengoetxea *et al.*: Learning and simulation of Bayesian networks applied to inexact graph matching, *Pattern Recognition*, **35**(12) (2002) 2867–2880
21. G. Syswerda: A Study of Reproduction in Generational and Steady-State Genetic Algorithms, *Foundations of Genetic Algorithms*, G. J. E. Rawlins Editors, Morgan Kaufman (1990) 94–101

Efficient Linkage Discovery by Limited Probing

Robert B. Heckendorn¹ and Alden H. Wright²

¹ University of Idaho, Moscow, ID 83844-1010 USA, heckendo@cs.uidaho.edu,

² Computer Science, University of Montana, Missoula, MT USA, wright@cs.umont.edu

Abstract. This paper addresses the problem of determining the epistatic linkage of a function from binary strings to the reals. There is a close relationship between the Walsh coefficients of the function and “probes” (or perturbations) of the function. This relationship leads to two linkage detection algorithms that generalize earlier algorithms of the same type. A rigorous complexity analysis is given of the first algorithm. The second algorithm not only detects the epistatic linkage, but also computes all of the Walsh coefficients. This algorithm is much more efficient than previous algorithms for the same purpose.

1 Introduction

In very simple fitness functions each bit in the domain independently contributes to the total value of the function. In optimizing these simple fitness functions, each bit can be tested independently against a fixed background of other bits to determine the contribution of that bit. Proceeding through all the bits the optimum can be found in linear time with respect to the number of bits.

Most practical functions, however, are not nearly as simple. For many, the contribution of a bit in the domain to the function value is non-linear in that it is dependent on the state of one or more other bits in the domain. This **linkage** effect is called **epistasis** and can be succinctly defined:

“...if the effect of one unit is not predictable unless the value of another unit is known, then the effects are epistatic...in other words, *the effect of a unit is context dependent*” [Bro00].

Applied to the case of evolutionary computation the “units” in the quote above refer to the positions in the problem representation whose values are selected from an **alphabet**. The more units, or positions, that simultaneously interact (the higher the epistasis) the greater the degree of freedom to “hide” the optimum anywhere in the subdomain formed by the interacting units [HW99]. High epistasis, however, is no guarantee of a difficult problem. Nor is low epistasis a guarantee of an easy problem. In fact, MAX3SAT problems are equivalent to problems of low epistasis in which all epistatic interactions are known and they are provably NP-complete [Hec99]. Still, knowing the location of epistatically interacting blocks of bits may be used to guide a search for the optimum or the formulation of a representation [MG99b, MG99a, KP01]. If the function is separable, each component can be solved separately. If the function is close to separable, this can guide the choice of crossover operators. In this case, Mühlenbein and Mahnig [MM99] also suggest

applying the UMDA algorithm where each component makes up a string position with a higher-order alphabet. Mühlenbein, Mahnig, and Rodriguez [MMR99] give a factorized distribution algorithm (FDA) that applies to additively decomposed functions (that we call embedded landscapes in this paper).

The general problem of discovering epistatic linkage has been addressed directly and indirectly by many papers. Munetomo and Goldberg showed a simple direct perturbational approach to generalized linkage discovery over a binary alphabet in [MG99b] [MG99a]. These papers also summarize some other approaches to the problem. Kargupta et al. [KP01] have shown that for epistatically bounded functions, f , where the epistasis is known to be bounded by k bits, all the Walsh coefficients, a direct measure of the magnitude of epistasis, can be computed in time $O(L^k)$, where L is the length of the representation.

In this paper we present a theoretical framework for the detection of epistatic linkage and the computation of Walsh coefficients for epistatically bounded functions. The Walsh coefficients completely describe the function and so completely characterize the epistatic linkage. The algorithms in this paper are **blackbox algorithms** in that they assume minimal prior knowledge of the function being analyzed. This paper deals with perturbation methods, or what we call **probes**. We give a randomized algorithm for linkage detection which is based on our theoretical framework, and we give rigorous complexity bounds for this algorithm. We extend this to another randomized algorithm that both detects linkage and computes the Walsh coefficients. This algorithm makes much more efficient use of function evaluations than previous algorithms.

2 Notation

The space of all bit strings of length L is denoted by $\mathcal{B}$. The binary operators on $\mathcal{B}$ include $\wedge$ which denotes bitwise AND, and $\oplus$ which denotes bitwise EXCLUSIVE-OR. An overbar (e. g., $\overline{m}$) denotes 1's complement. Since the L -bit binary representations of the integers in the interval $[0, 2^L)$ coincide with the elements of $\mathcal{B}$, a bit strings may be denoted by the corresponding integer. For example, the integer 2^k , $0 \leq k < L$ corresponds to the bit string with a single one in position k , where bit positions are labeled from the right starting at 0. Thus, $2^2 \equiv 0000100$ for $L = 7$. It is convenient to think of a bit string i as corresponding to the set of bit positions indicated by the 1 bits in i . Thus, we write $i \subseteq j$ (i is **contained in** j) when the set corresponding to i is contained in the set corresponding to j , i. e., when $i \wedge j = i$. If $i \subseteq j$ and $i \neq j$ we write $i \subset j$. The **unitation** or **bit count** function $bc(i)$ of string i is the number of ones in i . Given a mask $m \in \mathcal{B}$, let the set $\mathcal{B}_m = \{i \in \mathcal{B} : i \subseteq m\}$. Note $|\mathcal{B}_m| = 2^{bc(m)}$. Square brackets are used to denote an **indicator function**: if $expr$ is an expression that may be true or false, then

$$[expr] = \begin{cases} 1 & \text{if } expr \text{ is true} \\ 0 & \text{otherwise} \end{cases}$$

3 Walsh Analysis and Embedded Landscapes

Any function $f : \mathcal{B} \rightarrow \mathbb{R}$ can be written as a linear combination of Walsh functions:

$$f(x) = \sum_{i \in \mathcal{B}} w_i \psi_i(x)$$

where i^{th} Walsh function is defined:

$$\psi_i(x) = (-1)^{bc(i \wedge x)}$$

and the w_i are referred to as **Walsh coefficients**. The **Walsh transform** is a linear transform of the Walsh coefficients represented as a vector w in $\mathbb{R}^{2^L}$ to the function space f in $\mathbb{R}^{2^L}$. This is a change of basis transformation corresponding to the matrix Ψ with $\Psi_{i,j} = \psi_i(j)$.

$$f = \Psi w \quad \text{and} \quad w = \frac{1}{2^L} \Psi f \quad (1)$$

It is not hard to show that Ψ is symmetric and $\Psi\Psi = 2^L I$ where I is the identity matrix.

f **depends on** a bit position k , $0 \leq k < L$, if there exists a $j \in \mathcal{B}$ such that $f(j) \neq f(j \oplus 2^K)$. In other words, f depends on bit position k if flipping bit k changes the value assigned to some string j . The **support** of f is the set of loci that f depends on. The **support mask** of f is a bitstring in $\mathcal{B}$ with 1 bits in exactly and only those positions that support f . By the definition the support mask of ψ_i is i .

An **embedded landscape** is a function $f : \mathcal{B} \rightarrow \mathbb{R}$ which can be written in the form $f = \sum g_j$ where each subfunction g_j has a support mask m_j . Normally, there will be some restriction on the support set masks m_j . The function $f : \mathcal{B} \rightarrow \mathbb{R}$ has **k -bounded epistasis** if it can be written as the sum of subfunctions each of whose support is a set of at most k bits. It has been shown, perhaps most recently in [Hec02]:

Theorem 1. (*Embedded Landscape Theorem*) *A function $f : \mathcal{B} \rightarrow \mathbb{R}$ has k -bounded epistasis if and only if $w_j = 0 \forall bc(j) > k$*

Thus, f has k -bounded epistasis if and only if all of its Walsh coefficients of order greater than k are zero. The function f is **linear** if it has 1-bounded epistasis. The function f is **additively separable** if it can be written as a sum of at least two subfunctions where the supports of all subfunctions are pairwise disjoint.

4 Probes

A probe is a way of determining epistatic properties of a function $f : \mathcal{B} \rightarrow \mathbb{R}$ by performing a series of specific function evaluations. More specifically, a **probe** is:

$$P(f, m, c) = \frac{1}{2^{bc(m)}} \sum_{i \in \mathcal{B}_m} (-1)^{bc(i)} f(i \oplus c)$$

where $m \in \mathcal{B}$ and $c \in \mathcal{B}_{\overline{m}}$. c is called the **background** of the probe. The **order of the probe** is number of ones in the mask, or $bc(m)$. The direct computation of the value of a probe requires $2^{bc(m)}$ function evaluations.

Theorem 2. (Walsh Function Probing) For any $j, m \in \mathcal{B}$ and $c \in \mathcal{B}_{\overline{m}}$,

$$P(\psi_j, m, c) = \begin{cases} \psi_j(c) & \text{if } m \subseteq j \\ 0 & \text{otherwise} \end{cases}$$

Proof.

$$\begin{aligned} P(\psi_j, m, c) &= \frac{1}{2^{bc(m)}} \sum_{i \in \mathcal{B}_m} (-1)^{bc(i)} \psi_j(i \oplus c) \\ &= \frac{1}{2^{bc(m)}} \sum_{i \in \mathcal{B}_m} \psi_1(i) \psi_j(i \oplus c) \\ &= \frac{1}{2^{bc(m)}} \sum_{i \in \mathcal{B}_m} \psi_1(i) \psi_j(i) \psi_j(c) \\ &= \frac{1}{2^{bc(m)}} \psi_j(c) \sum_{i \in \mathcal{B}_m} \psi_{\bar{j}}(i) \end{aligned}$$

By the Balanced Sum Theorem for Hyperplanes [HW99] the sum is $2^{bc(m)}$ if $\bar{j} \subseteq \overline{m}$ which is the same as $m \subseteq j$ and is 0 otherwise. $\square$

A probe is really probing for nonzero Walsh coefficients by adding and subtracting over a set of Walsh coefficients. If the result is nonzero then one of the component Walsh coefficients is nonzero. If it is zero then without further information we can say very little. The following theorem identifies the set of Walsh coefficients.

Theorem 3. (Probe Subset) For any $m \in \mathcal{B}$ and $c \in \mathcal{B}_{\overline{m}}$,

$$P(f, m, c) = \sum_{j \in \mathcal{B}} [m \subseteq j] w_j \psi_j(c)$$

Proof. Recall that $f = \sum_{j \in \mathcal{B}} w_j \psi_j$. Thus,

$$\begin{aligned} P(f, m, c) &= \sum_{j \in \mathcal{B}} w_j P(\psi_j, m, c) \\ &= \sum_{j \in \mathcal{B}} [m \subseteq j] w_j \psi_j(c) \quad \text{by the Walsh Function Probing theorem} \end{aligned}$$

$\square$

A **maximal nonzero Walsh coefficient** is a Walsh coefficient w_m such that $w_m \neq 0$ and $w_j = 0 \forall j \supset m$.

Corollary 1. (Maximal Probe) If w_m is a maximal nonzero Walsh coefficient, then for any $c \in \mathcal{B}_{\overline{m}}$,

$$P(f, m, c) = w_m$$

Proof. It follows from theorem 3 that

$$P(f, m, c) = w_m \psi_m(c)$$

And from the definition of a Walsh function: $\psi_m(c) = (-1)^{bc(m \wedge c)} = (-1)^0 = 1$. $\square$

A probe can be written as a sum of lower-order probes.

Theorem 4. (*Probe Recursion*) For any function $f : \mathcal{B} \rightarrow \mathbb{R}$, any masks $m, n \in \mathcal{B}$ with $n \subseteq m$, and any $c \in \mathcal{B}_{\overline{m}}$:

$$P(f, m, c) = \frac{1}{2^{bc(n)}} \sum_{i \in \mathcal{B}_n} (-1)^{bc(i)} P(f, m \oplus n, i \oplus c)$$

Proof. Any $j \in \mathcal{B}_m$ can be written uniquely as $j = i \oplus u$ where $i \in \mathcal{B}_n$ and $u \in \mathcal{B}_{m \oplus n}$. Thus:

$$\begin{aligned} P(f, m, c) &= \frac{1}{2^{bc(m)}} \sum_{j \in \mathcal{B}_m} (-1)^{bc(j)} f(j \oplus c) \\ &= \frac{1}{2^{bc(n)}} \sum_{i \in \mathcal{B}_n} (-1)^{bc(i)} \frac{1}{2^{bc(m \oplus n)}} \sum_{u \in \mathcal{B}_{m \oplus n}} (-1)^{bc(u)} f(u \oplus i \oplus c) \\ &= \frac{1}{2^{bc(n)}} \sum_{i \in \mathcal{B}_n} (-1)^{bc(i)} P(f, m \oplus n, i \oplus c) \end{aligned}$$

Theorem 5. (*Nonzero Probe Existence*) Given a maximal nonzero Walsh coefficient $w_m \neq 0$ for all $a : a \subseteq m$, there exists an $i \in \mathcal{B}_{m \oplus a}$ such that

$$P(f, a, i \oplus c) \neq 0 \quad \forall c \in \mathcal{B}_{\overline{m}}$$

Proof. By the Maximal Probe Corollary, $P(f, m, c) = w_m \neq 0$ for any $c \in \mathcal{B}_{\overline{m}}$. By the Probe Recursion Theorem applied with $n = m \oplus a$,

$$P(f, m, c) = \frac{1}{2^{bc(n)}} \sum_{i \in \mathcal{B}_n} (-1)^{bc(i)} P(f, m \oplus n, i \oplus c)$$

Thus, there must exist an $i \in \mathcal{B}_n$ such that $P(f, m \oplus n, i \oplus c) = P(f, a, i \oplus c) \neq 0$. $\square$

5 The Linkage Graph and Hypergraph

A **hypergraph** is a collection of vertices V together with a family of subsets E of V called hyperedges where each hyperedge is nonempty. The set of vertices of the **linkage hypergraph** is the set of string positions j . A set of vertices corresponding to mask $m \neq 0$ is a hyperedge if there is a $c \in \mathcal{B}_{\overline{m}}$ such that $P(f, m, c) \neq 0$. A hyperedge will be identified with the corresponding mask. The **order of a hyperedge** is the number of ones in the mask. In view of Theorem 5, the mask m is a hyperedge if and only if there is a $j \supseteq m$ such that $w_j \neq 0$. Thus, we have the following corollary.

Corollary 2. *If m is a hyperedge of the hypergraph, and if $a \subseteq m$, then a is also a hyperedge.*

```

DETECT-LINKAGE( $j, N$ )
begin
   $V \leftarrow \{0, 1, \dots, L - 1\}$ 
   $E \leftarrow \emptyset$ 
  for  $i \leftarrow 1$  to  $N$  do
    for each mask  $m$  with  $bc(m) = j$  do
      if  $m \notin E$  then
         $c \leftarrow$  a random string in  $\mathcal{B}_{\overline{m}}$ 
        if  $P(f, m, c) \neq 0$  then
           $E \leftarrow E \cup \{m\}$ 
        end if
      end if
    end for
  end for
  return  $E$ 
end DETECT-LINKAGE

```

The order- j linkage detection algorithm constructs the set of order- j hyperedges of the linkage hypergraph. The order-2 version of this algorithm is similar to the LINC algorithm of [MG99b]. However, they start with a population of strings. Then each probe is done using one of the strings of the population to provide the background for the probe.

For an arbitrary function f it is impossible to conclude anything conclusively from evaluating f at a subset of points. For example, if f would be k -epistatically bounded except for its value at one point, then the above algorithm for $j > k$ will return 0 for any probe unless the probe happens to sample the one exceptional point. For a large string length, the probability that this one exceptional point is sampled can be very small.

Thus, assumptions on f are needed in order to use the order- j linkage detection algorithm to make conclusions. The natural assumption is that f is k -epistatically bounded. The following theorems give a worst-case complexity analysis of the order- j linkage detection algorithm in this case.

Theorem 6. (Nonzero Probe Probability) *Let f be k -epistatically bounded and let m be a mask corresponding to an order- j hyperedge of the linkage hypergraph of f . If c is a randomly chosen string in $\mathcal{B}_{\overline{m}}$, then the probability that $P(f, m, c) \neq 0$ is at least 2^{j-k} .*

Proof. Since m is a hyperedge by the Nonzero Probe Existence Theorem there is a u such that $m \subseteq u$ and $w_u \neq 0$. Without loss of generality we can assume that u has the property that $u \subset v \Rightarrow w_v = 0$. By assumption, $bc(u) \leq k$. Theorem 5 shows that there is at least one $i \in \mathcal{B}_{u \oplus m}$ such that $P(f, m, i \oplus b) \neq 0$ for any $b \in \mathcal{B}_{\overline{u}}$. The probability that the randomly selected background c matches some such i on the positions masked by $u \oplus m$ is at least $2^{-bc(u \oplus m)} = 2^{bc(m) - bc(u)} \geq 2^{j-k}$. $\square$

The lower bound of Theorem 6 cannot be improved under these assumptions. Start with a $(j-1)$ -epistatically bounded function whose support is m with $bc(m) = k$, and then perturb the value of one point. Any probe that does not include the perturbed point will return a value of zero. Since an order- j probe includes 2^j points, and since there are 2^k probes, the probability of including the perturbed point is 2^{j-k} .

Theorem 7. *Let f be k -epistatically bounded and let J be the number of order- j hyperedges in the linkage hypergraph of f . If the number of iterations N in the order- j linkage detection algorithm is chosen so that either*

$$N \geq \frac{\ln(1 - \delta^{1/J})}{\ln(1 - 2^{j-k})}$$

or

$$N \geq -2^{k-j} \ln(1 - \delta^{1/J})$$

then the probability that all order- j hyperedges are detected is at least δ .

Proof. In the following, a “success” is the detection of a nonzero probe. Theorem 6 shows that the probability of failure for one probe on one trial is at most $1 - 2^{j-k}$. Thus, the probability of failure on N trials is at most $(1 - 2^{j-k})^N$, and the probability of success on N trials is at least $1 - (1 - 2^{j-k})^N$. The probability of success on all J hyperedges is at least

$$(1 - (1 - 2^{j-k})^N)^J$$

Thus, we want to choose N so that

$$\begin{aligned} ((1 - (1 - 2^{j-k})^N))^J &\geq \delta \\ 1 - \delta^{1/J} &\geq (1 - 2^{j-k})^N \\ \ln(1 - \delta^{1/J}) &\geq N \ln(1 - 2^{j-k}) \\ \frac{\ln(1 - \delta^{1/J})}{\ln(1 - 2^{j-k})} &\leq N \end{aligned}$$

To prove the second formula, note that $-\ln(1 - x) \geq x$ for any $x > 0$ from the Taylor series of $-\ln(1 - x)$. Thus, $2^{k-j} \geq 1/\ln(1 - 2^{j-k})$. $\square$

Strictly speaking, these results do not apply to the LINC algorithm of [MG99b] since the above analysis assumes that the backgrounds of probes are chosen independently, and this is not the case for the LINC algorithm. However, our empirical results show that these formulas are quite accurate when the backgrounds are chosen from a population. (See Section 8.) In fact, it is much more accurate than the population sizing formula given by [MG99b].

Thus, the overall worst-case complexity of the order-2 linkage detection algorithm is $O(2^k L^2 \ln(1 - \delta^{1/J}))$ if one wants to maintain the same probability of overall success as the string length increases. Munetomo and Goldberg [MG99b] give the overall complexity of their LINC algorithm as $O(2^k L^2)$, but this assumes that the probability of success per subfunction of the embedded landscape stays constant as the string length increases, which would seem to be a less desirable assumption.

6 Computing the Walsh Coefficients Using the Kargupta-Park Top-Down Algorithm

Kargupta and Park [KP01] give a “deterministic” algorithm to find the Walsh coefficients of a function f with k -bounded epistasis. It is “top-down” since it does high-order probes before low-order probes. In this section we show how this algorithm can be expressed in terms of probes.

Let w_m be a maximal nonzero Walsh coefficient. The Maximal Probe Corollary shows that $P(f, m, c) = w_m$ for any $c \in \mathcal{B}_m$. Thus, if f has k -bounded epistasis, and if we do the probe $P(f, m, 0)$ where $bc(m) = k$, the result will be w_m . Thus, all of the order- k Walsh coefficients can be computed by doing $\binom{L}{k}$ probes, each of which uses 2^k function evaluations.

Let j be a mask with $bc(j) = k - 1$. Then Theorem 3 gives the equation

$$P(f, j, 0) = w_j + \sum_{j \subset u} w_u \psi_u(0) = w_j + \sum_{j \subset u} w_u \quad (2)$$

(Note that $\psi_u(0) = 1$.) The potentially nonzero Walsh coefficients in the summation are all of order k and have been computed. Thus, w_j can be computed from $P(f, j, 0)$ plus these order- k Walsh coefficients. Let m be such that $bc(m) = k$ and $j \subseteq m$. If the Probe Recursion Theorem is applied to $P(f, m, 0)$ with $n = m \oplus j$, then the first term in the summation is $P(f, j, 0)$. This shows that all function evaluations necessary to compute $P(f, j, 0)$ have already been done in the computation of $P(f, m, 0)$. (This observation is ours and is not included in [KP01].)

The same idea can be used to compute the lower order Walsh coefficients. Thus, the Walsh coefficients are computed in of decreasing bit count, starting with bit count k .

7 Detecting Linkage and Computing the Walsh Coefficients

Kargupta and Park [KP01] give a “bottom up” randomized algorithm that finds the nonzero Walsh coefficients. They suggest that they can find the values of these nonzero Walsh coefficients, but the method to do this is not included in their algorithm, and so presumably one applies the algorithm referred to in Section 6.

In this section, we give a well-specified algorithm that efficiently finds the nonzero Walsh coefficients and computes their values. The algorithm first proceeds in a bottom-up fashion to find which Walsh coefficients are nonzero, and then it proceeds top-down to determine their values without doing any additional function evaluations. (We assume that function evaluations are disproportionately expensive to compute.)

A key observation is that if probe backgrounds are determined using a population, as in the Munetomo/Goldberg LINC algorithm, then higher order probes can be computed relatively cheaply by using the function evaluations of previously computed lower order probes. This is justified by Theorem 8 below. In other words, computing $P(f, m, c)$ can be done with only one additional function evaluation as long as the probes for all a , $a \subset m$, have been computed using the same background c .

Theorem 8. For any $m \in \mathcal{B}$, $c \in \mathcal{B}_{\overline{m}}$,

$$f(m \oplus c) = \sum_{a \in \mathcal{B}_m} (-2)^{bc(a)} P(f, a, c)$$

This can be restated as:

$$P(f, m, c) = f(m \oplus c) - \sum_{a \in \mathcal{B}_m \setminus \{m\}} (-2)^{bc(a)} P(f, a, c)$$

Proof. Using the definition of a probe, the conclusion can be rewritten as:

$$f(m \oplus c) = \sum_{a \in \mathcal{B}_m} (-1)^{bc(a)} \sum_{i \in \mathcal{B}_a} (-1)^{bc(i)} f(i \oplus c)$$

We prove this by induction on $bc(m)$. The base cases of $bc(m) = 0$ and $bc(m) = 1$ are easy. Let $m = u \oplus v$ where $u \wedge v = 0$, $u \neq 0$, $v \neq 0$. Then

$$\begin{aligned} & \sum_{a \in \mathcal{B}_m} (-1)^{bc(a)} \sum_{i \in \mathcal{B}_a} (-1)^{bc(i)} f(i \oplus c) \\ &= \sum_{j \in \mathcal{B}_u} (-1)^{bc(j)} \sum_{k \in \mathcal{B}_v} (-1)^{bc(k)} \sum_{i \in \mathcal{B}_{j \oplus k}} (-1)^{bc(i)} f(i \oplus c) \\ &= \sum_{j \in \mathcal{B}_u} (-1)^{bc(j)} \sum_{r \in \mathcal{B}_j} (-1)^{bc(r)} \sum_{k \in \mathcal{B}_v} (-1)^{bc(k)} \sum_{s \in \mathcal{B}_k} (-1)^{bc(s)} f(s \oplus r \oplus c) \\ &= \sum_{j \in \mathcal{B}_u} (-1)^{bc(j)} \sum_{r \in \mathcal{B}_j} (-1)^{bc(r)} f(v \oplus r \oplus c) \\ &= f(u \oplus v \oplus c) = f(m \oplus c) \end{aligned}$$

□

The algorithm takes advantage of previously computed function evaluations by caching all function evaluations in a hash table. When the function f is applied to a bit string, this hash table is checked before doing the actual function evaluation.

The basic idea of the bottom-up part of the algorithm (TRAVERSE-HYPERGRAPH) is to do a breadth-first traversal of the lattice of masks, starting with the empty mask, then looking at the order-1 masks, etc. When a new mask m is considered for inclusion in the linkage hypergraph, all submasks of order $bc(m) - 1$ are checked for membership in the hypergraph. If any of these submasks is not in the hypergraph, then m cannot be in the hypergraph. If these tests succeed, then a sequence of probes is done to determine if the mask is in the hypergraph.

The backgrounds of the probes can be determined either by using a population or by randomly choosing background strings. The first element of the population or the first background is the all-zeros string since this simplifies the computation of the Walsh coefficients in the top-down part of the algorithm. If a population is used, the remainder of the population is chosen randomly. The probe value using the all-zeros background is saved in the hash-table *hypergraph* which is also used to determine whether a mask has been added to the hypergraph.

In addition to the queue used for the breadth-first traversal, the masks added to the hypergraph are stored in a linked list *hypergraphList* which is traversed in the top-down part of the algorithm.

The TESTBYPROBES function does up to N probes using the mask a . If one of these probes is nonzero (or greater than a tolerance in practice), then it returns the probe value corresponding to the all-zeros string. If all probes are zero, then it returns *null*.

```

TRAVERSE-HYPERGRAPH()
  population.initialize()
  hypergraphList.initialize()
  queue.initialize()
   $m \leftarrow \{ \}$  // Empty mask
  ProbeValue  $\leftarrow$  TESTBYPROBES( $a$ )
  if ProbeValue  $\neq$  null then
    queue.add( $m$ )
    hypergraph[ $m$ ]  $\leftarrow$  ProbeValue
  end if
  while queue.notEmpty() do
     $m \leftarrow$  queue.remove()
    probeValue  $\leftarrow$  hypergraph[ $m$ ]
    for all supersets  $a$  of  $m$  of cardinality  $bc(m) + 1$  do
      if all subsets of  $a$  of cardinality  $bc(m)$  are in the hypergraphList then
        ProbeValue  $\leftarrow$  TESTBYPROBES( $a$ )
        if ProbeValue  $\neq$  null then
          queue.add( $a$ )
          hypergraph[ $a$ ]  $\leftarrow$  ProbeValue
          hypergraphList.addFirst( $a$ )
        end if
      end if
    end for
  end while

```

The top-down part of the algorithm (COMPUTE-WALSH-COEFS) traverses the hyperedges of hypergraph using the list *hypergraphList* from higher order masks to lower order, that is in the reverse order from which they were added to the hypergraph. The Walsh coefficients are computed using only the function evaluations done in the bottom-up part of the algorithm.

The algorithm is based on Equation 2. This equation would suggest that to compute w_a , one would want to traverse those supersets of a that correspond to hyperedges. However, in the top-down algorithm we are already traversing these superset hyperedges, and it is more efficient to add the Walsh coefficient of each these superset hyperedges to its subsets, and this is what the algorithm does. In other words, as the supersets of a are traversed in the algorithm, their Walsh coefficients are added to $wCoe.f[a]$.

```

COMPUTE-WALSH-COEFS(hypergraphList)
  for  $m \in$  hypergraphList do //traverse in the reverse order from the order added

```

```

probeValue  $\leftarrow$  hypergraph[m]
if wCoeef[m]  $\neq$  null then wCoeef[m]  $\leftarrow$  wCoeef[m] + probeValue
else wCoeef[m]  $\leftarrow$  probeValue end if
for each  $a \subset m$  do
  if wCoeef[a]  $\neq$  null then wCoeef[m]  $\leftarrow$  wCoeef[a] - wCoeef[m]
  else wCoeef[a]  $\leftarrow$  -wCoeef[m] end if
end for
end for

```

8 Empirical Results

The test function used in this section is an embedded landscape with 50 5-bit subfunctions and a string length of 50. Each subfunction is linear with a randomly placed “needle”. The coefficients of the linear function are chosen randomly from the interval $[0, 1]$. The needle is a single point with a value of 0.1 greater than the maximum value given by the linear function. The support of each subfunction is randomly chosen. A value is considered to be zero if it is less than 10^{-7} . This class of test function represents the worst case for the algorithms given in this paper.

The algorithm used is that given in Section 7. The algorithm is considered to be successful only if it correctly finds all hyperedges of the hypergraph. On smaller examples, when the algorithm finds all hyperedges, it correctly computes all Walsh coefficients. When the algorithm fails (on this class of functions), it is most likely to fail when doing the order-2 probes. Thus, the formula of Theorem 7 should be applied with $j = 2$ and $k = 5$. The number of order-2 hyperedges is at most $50 \binom{5}{2} = 500$ since there are $\binom{5}{2}$ order-2 hyperedges per subfunction. However, some of these overlap, and the actual number is about 420.

The algorithm of Section 7 was run for 1000 trials for each of $N = 40, 50, 60, 70, 80, 90$, where N is the number of probes per potential hyperedge. (i. e., N is the population size.) The algorithm was also run with the same parameters using randomly chosen backgrounds instead of backgrounds from a population. In addition, the first equation of Theorem 7 was solved for the success rate for the same values of N and with $j = 2$, $k = 5$, and $J = 420$. These are shown in the table on the left below. The table on the right shows the average number of function evaluations for these experiments.

These results suggest that when f is an embedded landscape with the number of subfunctions being $O(L)$, then the complexity is given by the formula of Theorem 7. Further theory and/or experiments are needed to confirm this hypothesis.

Accuracy			
N	Theory	Population	Random
40	0.1331	0.222	0.168
50	0.5889	0.658	0.648
60	0.8700	0.891	0.888
70	0.9640	0.963	0.967
80	0.9904	0.992	0.992
90	0.9975	1.00	0.995

Function Evaluations		
N	Population	Random
40	69008	279526
50	85889	345577
60	102547	411381
70	119241	490876
80	135907	540427
90	152501	604383

9 Conclusions

There are two contributions of this paper. First, the paper gives a rigorous mathematical foundation for perturbational methods for determining the epistatic structure of a function from binary strings to the real numbers. These methods are closely related to the Walsh basis representation of the function.

Second, the paper gives two new randomized algorithms. The first generalizes the LINC algorithm of [MG99b] and [MG99a] to finding epistasis of arbitrary order. The primary parameter in the algorithm is the number of probes. If the function has k -bounded epistasis (is k -delineable in the terminology of [MG99a]), then rigorous bounds can be given for the number of probes that are needed, and this leads to a complexity analysis of the algorithm. The second algorithm generalizes the algorithms of [KP01]. This algorithm both determines the epistatic structure and finds the Walsh coefficients of a k -epistatic function. It is more practical when most of the Walsh coefficients of order less than k are zero. This algorithm is more efficient than the methods of [KP01].

More research is needed in applying this class of algorithms to functions where the assumptions of k -bounded epistasis and sparseness of the Walsh basis representation are only approximately satisfied. Further research is also needed in understanding how genetic algorithms and estimation of distribution algorithms can work in tandem to take advantage of the epistatic structure of functions that can be discovered by algorithms such as the ones given in this paper.

References

- [Bro00] E. D. Brodie. *Why Evolutionary Genetics Doesn't Always Add Up*, pages 3–19. Oxford University Press, Oxford, England, 2000.
- [Hec99] Robert B. Heckendorn. *Walsh Analysis, Epistasis, and Optimization Problem Difficulty for Evolutionary Algorithms*. PhD thesis, Colorado State University, Department of Computer Science, Fort Collins, Colorado, 1999.
- [Hec02] Robert B. Heckendorn. Embedded landscapes. *Evolutionary Computation*, 10(4): 345–376, 2002.
- [HW99] R. B. Heckendorn and Darrell Whitley. Predicting epistasis from mathematical models. *Evolutionary Computation*, 7(1):69–101, 1999.
- [KP01] H. Kargupta and B. Park. Gene expression and fast construction of distributed evolutionary representation. *Evolutionary Computation*, 9(1):43–69, 2001.
- [MG99a] Masaharu Munetomo and David E. Goldberg. Identifying linkage groups by nonlinearity/non-monotonicity detection. In W. Banzhaf et. al., editor, *Proc. of the Genetic and Evolutionary Computation Conference*, volume 1, pages 433–440, Palo Alto, CA, 1999. Morgan Kaufmann Publishers, Inc.
- [MG99b] Masaharu Munetomo and David E. Goldberg. Linkage identification by non-monotonicity detection for overlapping functions. *Evolutionary Computation*, 7(4): 377–398, 1999.
- [MM99] Heinz Mühlenbein and Thilo Mahnig. Convergence theory and application of the factorized distribution algorithm. *Journal of Computing and Information Technology*, 7(1):19–32, 1999.
- [MMR99] Heinz Mühlenbein, Thilo Mahnig, and Aberto O. Rodriguez. Schemata, distributions and graphical models in evolutionary optimization. *J. of Heuristics*, 5:215–247, 1999.

Distributed Probabilistic Model-Building Genetic Algorithm

Tomoyuki Hiroyasu¹, Mitsunori Miki¹, Masaki Sano¹, Hisashi Shimosaka¹,
Shigeyoshi Tsutsui², and Jack Dongarra³

¹ Doshisha University, Kyoto, Japan,
`tomo@is.doshisha.ac.jp`,

² Hannan University, Osaka, Japan,

³ University of Tennessee, TN, USA

Abstract. In this paper, a new model of Probabilistic Model-Building Genetic Algorithms (PMBGAs), Distributed PMBGA (DPMBGA), is proposed. In the DPMBGA, the correlation among the design variables is considered by Principal Component Analysis (PCA) when the offsprings are generated. The island model is also applied in the DPMBGA for maintaining the population diversity. Through the standard test functions, some models of DPMBGA are examined. The DPMBGA where PCA is executed in the half of the islands can find the good solutions in the problems whether or not the problems have the correlation among the design variables. At the same time, the search capability and some characteristics of the DPMBGA are also discussed.

1 Introduction

Genetic Algorithms (GAs) are stochastic search algorithms based on the mechanics of natural selection and natural genetics[1]. The GAs can be applied to several types of optimization problems by encoding design variables to individuals. Recently, a new type of GA called the Probabilistic Model-Building GA (PMBGA)[2] or Estimation of Distribution Algorithm (EDA)[3] have been the focus. In the canonical GA, children are generated from parents that are selected randomly. However, in the PMBGA and EDA, the good characteristics of parents are inherited by children using the statistical information. Since children must have the characteristics of parents, effective searching is expected. It is reported that the PMBGA and EDA have a better search ability than that of the canonical GA.

To make an effective search in continuous problems, the correlation among the design variables should be handled. Therefore, new searching points should be generated so that the correlation exists in the new points. Many real coded GAs where the real vectors are used as a genotype treat this correlation problem. One of the typical real coded GAs is Unimodal Normal Distribution Crossover (UNDX)[4]. The UNDX is good at finding the optimum in the functions where there is strong correlation between the design variables. Takahashi et al. introduced a new method[5]; they used the Principle Component Analysis (PCA) and

the Individual Component Analysis (ICA). In the Takahashi et al. algorithm, by using the PCA and the ICA, the individuals are transferred into the space. Then, Blend Crossover (BLX- α) [6] is performed to the transferred individuals. The correlation between the design variables is considered by using both ICA and PCA. At the same time, the diversity of the solutions is maintained by using BLX- α . Besides GAs, some of the evolutionary strategies[7] have the operation for the correlation relationship among the design variables. One of them is the correlated mutation method that is proposed by Schwefel[8]. In this method, there is a parameter that indicates the direction of the distribution of the individuals. By using this parameter, the mutation operation is performed by considering the correlation relationship among the design variables.

In this paper, a new PMBGA for continuous problems is proposed, which is called Distributed PMBGA (DPMBGA). This is one of the real coded GAs and the real vectors are treated as a genotype. In this algorithm, PCA is used for transforming the set of the solutions. This operation handles the correlation relationship among the design variables. The PMBGA sometimes lacks the diversity of the solutions during the search. To overcome this problem, a model of the distributed GA is performed.

In this paper, the basic algorithm of the DPMBGA is explained. In this paper, the DPMBGA is applied to solve test functions. Through these experiments, the following four topics of the DPMBGA are discussed. Firstly, the DPMBGA that is based on the distributed scheme is examined. Secondly, the search capability of the DPMBGA is compared with UNDX and Minimum Generation Gap. From the results, it is found that the PCA prevents the effective search in some functions. The discussion of the results is the third topic. Finally, the search capability of the DPMBGA for functions whose optimum is located near the boundary is discussed.

2 Distributed Probabilistic Model-Building Genetic Algorithm

2.1 Flow of DPMBGA

In this paper, a new PMBGA is proposed: that is Distributed Probabilistic Model-Building Genetic Algorithm (DPMBGA). The DPMBGA uses the distributed GA scheme[9,10]. Therefore, there are several subpopulations and the migration operation is performed. In the DPMBGA, the following procedure is performed for the migration operation. The topology of the migration is a ring. This ring topology is formed randomly when the migration is performed. In the ring, migrated individuals are moved from one subpopulation to the other in one direction. The migrated individuals are chosen randomly and they are substituted for the individuals whose fitness values are the worst in the subpopulation.

In the DPMBGA, the following procedures are performed at the generation t .(Fig. 1).

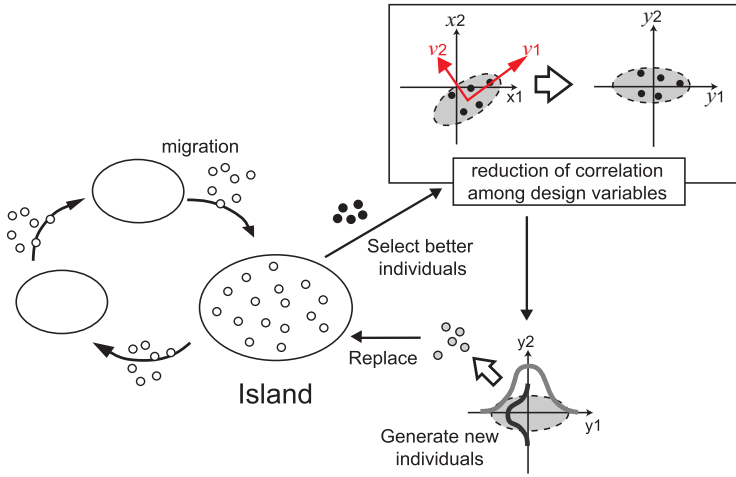


Fig. 1. DPMBGA

1. The elite individual is reserved.
2. The individuals who have the good values of the fitness are sampled.
3. The above individuals are transferred by the PCA into the new space.
4. The new individuals are generated in the new space.
5. The new individuals are transferred into the original space.
6. The new individuals are substituted for the old individuals.
7. The mutation is operated.
8. When the reserved elite individuals are eliminated, they are recovered.
9. The new individuals are evaluated.

In the following sections, each operation is explained precisely.

2.2 Sampling Individuals for Probabilistic Model

The following operation is performed in each island. The individuals who have the good evaluation values are chosen from each island $P_{sub}(t)$. The number of these individuals is determined with the sampling rate R_s . These individuals become sample individuals $S(t)$. The new individuals are generated from the information of these sampling individuals. These sampled individuals are chosen according to the higher fitness values. However, the same individual is not chosen repeatedly, hence the total number of $S(t)$ is fixed. When the number of individuals is low, the individuals are generated randomly and added to $S(t)$ and $S(t)$ exists in each subpopulation.

2.3 Sampling Individuals for PCA

This operation is also performed in each island. $S(t)$ is transferred by the PCA operation. The PCA is determined using the information of individual set $T(t)$.

$T(t)$ is different from $S(t)$ and $T(t)$ is formed in the following way. $T(t)$ consists of the individuals who are the best in each generation. Even when the number of $T(t)$ is less than the particular size, the new individuals are not added. When the size of $T(t)$ is exceeded, the worst individual is eliminated one by one. By this operation, the arbitrary number of the individuals can be used for the information of PCA. This is independent from the number of the subpopulation. $T(t)$ also exists in each island.

2.4 PCA Transformation

The average of $T(t)$ is subtracted from $T(t)$ and $T(t)$ becomes matrix $\mathbf{T}$ ($nT(t)column \times Dline$). The average of $T(t)$ is also subtracted from $S(t)$ and $S(t)$ becomes $\mathbf{X}$ ($nS(t)column \times Dline$). Then, the covariance matrix $\mathbf{S}$ of $\mathbf{T}$ is derived and the eigen values and vectors are obtained. $\mathbf{S}$ is a real symmetric matrix and derived as follows,

$$\mathbf{S} = \frac{1}{nS(t) - 1} \mathbf{T}^T \mathbf{T}. \quad (1)$$

The eigen vector indicates the axis of the new space.

Using the derived eigen vectors, the design variables $\mathbf{X}$ of the solution set $S(t)$ are transferred. After the transfer into the new space, there is no correlation among the design variables. The coordinate transfer matrix consists of the vectors $\mathbf{V} = [v_1, v_2, \dots, v_D]$. After multiplying $\mathbf{V}$, the vector $\mathbf{X}$ becomes $\mathbf{Y}$. The coordinate of $\mathbf{Y}$ corresponds to the eigen vectors.

2.5 Generation of New Individuals

The new individuals are generated using the normal distribution of the information of $\mathbf{Y}$. Each value of the design variable in a new individual is also determined one by one independently. Therefore, when there are n design variables in an individual, there should be n different normal distributions. The normal distribution is formed as follows; the average is the same as the average value of the target design value of $\mathbf{Y}$. The distribution is derived by multiplying the distribution of $\mathbf{Y}$ by the parameter *Amp*. The values of design variables are determined randomly, but the total distribution of the new individuals should be the same as the formed normal distribution. The number of the created new individuals is the same as the number of individuals in an island ($nP(t)$) and the generated individuals are stored in $\mathbf{Y}_{offs}$.

2.6 Restoring Correlation and Substitution of Old Individuals with New Individuals

$\mathbf{Y}$ is the transferred set of $\mathbf{X}$ into the new space. In this step, the derived $\mathbf{Y}_{offs}$ is then substituted into the original space. $\mathbf{Y}_{offs}$ is multiplied by the inverse of $\mathbf{V}$. After this operation, the set of $\mathbf{Y}_{offs}$ is in the original space.

$$\mathbf{X}_{offs} = \mathbf{Y}_{offs} \cdot \mathbf{V}^{-1} \quad (2)$$

The average of $\mathbf{X}_{offs}$ is added to the new individuals. These new individuals are substitute for the old ones $P(t)$ and those become $P(t+1)$.

2.7 Mutation

The values of the design variables are changed randomly within the constraints using the mutation ratio R_{mu} .

2.8 Preservation and Recovering of Elite Individuals

The elite individuals are preserved as $E(t)$. The number of the preserved individuals is $nE(t)$. After the substitution of the new individuals generated from the probabilistic model, the elite individuals are recovered in the total population. In this case, the elite individuals $E(t)$ are substituted with the individuals $P(t+1)$ whose evaluation values are not good.

3 Test Functions and Used Parameters for Numerical Experiments

In the following sections, the search capability and the characteristics of the DPMBGA are discussed. These discussions are illustrated through numerical experiments. In this section, the test functions and parameters for these experiments are explained.

The DPMBGA is used to find optimum solutions of the following five test functions: the Rastrigin function, Schwefel function, Rosenbrock function, Ridge function, and Griewank function. All of test functions are minimization problems. The global optimums are located at 0. There are 10 dimensions of the design variables in the Schwefel function and 20 dimensions of the design variables in the rest of the functions.

There is no correlation between the design variables in the Rastrigin function and the Schwefel function. There are many sub-peaks in the landscape of these functions. On the other hand, there is a correlation between the design variables in the Rosenbrock function and the Ridge function. In these test functions, there is only a peak in a landscape. In the Griewank function, there is a correlation between the design variables and many peaks in the landscape.

$$F_{Rastrigin} = 10n + \sum_{i=1}^n (x_i^2 - 10 \cos(2\pi x_i)) \quad (3)$$

$$(-5.12 \leq x_i < 5.12)$$

$$\begin{aligned}
 F_{Schwefel} &= \sum_{i=1}^n -x_i \sin\left(\sqrt{|x_i|}\right) - C \\
 &\quad (C : \textit{optimum}.) \\
 &\quad (-512 \leq x_i < 512)
 \end{aligned}
 \tag{4}$$

$$\begin{aligned}
 F_{Rosenbrock} &= \sum_{i=2}^n (100(x_1 - x_i^2)^2 + (1 - x_i)^2) \\
 &\quad (-2.048 \leq x_i < 2.048)
 \end{aligned}
 \tag{5}$$

$$\begin{aligned}
 F_{Ridge} &= \sum_{i=1}^n \left(\sum_{j=1}^i x_j \right)^2 \\
 &\quad (-64 \leq x_i < 64)
 \end{aligned}
 \tag{6}$$

$$\begin{aligned}
 F_{Griewank} &= 1 + \sum_{i=1}^n \frac{x_i^2}{4000} - \prod_{i=1}^n \left(\cos\left(\frac{x_i}{\sqrt{i}}\right) \right) \\
 &\quad (-512 \leq x_i < 512)
 \end{aligned}
 \tag{7}$$

The parameters used in these experiments are summarized in Table 1.

Table 1. Parameters

Population size	512
Number of elites	1
Number of islands	32
Migration rate	0.0625
Migration interval	5
Archive size for PCA	100
Sampling rate	0.25
Amp. of Variance	2
Mutation rate	0.1/ (Dim. of function)

4 Discussion on Effectiveness of PCA and Distributed Environment Scheme

In the DPMBGA, the new individuals are generated using the PCA. By this operation, the information from the correlation among the design variables that is found during the search is reflected to the new individuals. In the problems where there is a correlation among the design variables, the distribution of the value of each design variable is not independent from each other. Therefore,

the distribution of each design variable is affected by the distribution of the other design variables. In the new model, at first, the set of the individuals is transferred into the space where there is no correlation among the design variables. After this transformation, it is easy to generate new individuals using their information of the distribution. Therefore, it is expected that the DPMBGA can perform an effective search in the problems where there is a strong correlation among the design variables. In the numerical experiments, the following three models are discussed.

Model 1 : In every island, the PCA is performed.

Model 2 : In every island, the PCA is not performed.

Model 3 : In half of the islands, the PCA is performed.

In these models, the number of the islands where the PCA is performed is different. The model 1 is the same model explained in section 2. In model 2, the PCA is not performed at all. In model 3, the distributed environment scheme is applied for using the PCA.

The Distributed Environment GA (DEGA) is one of the distributed GA schemes that Miki et al. proposed[11]. In the DEGA, the different parameters or the different operations are performed in each island. It is well known that the search capability of GA depends on the value of the parameters. The optimum values of these parameters also depend on the targeted problems. Therefore, preliminary experiments are necessary in order to derive the optimum values of the parameters. In the DEGA, the values of the parameters and the operations are different in each island. These parameter values are not the best but can derive adequate solutions. In this paper, the DEGA scheme is applied; the CPA is performed in some islands and not in other islands.

In Table 2, the number of the trials where the optimum value is derived is summarized in using 20 trials. The higher number means that the robust search has been performed. In Figure 2, the average number of the evaluations is illustrated when the simulation derives the optimum values. The model that has the smaller number of this value may be the better model.

Table 2. Number of times that the threshold is reached

	model 1	model 2	model 3
Rastrigin	0	20	20
Schwefel	20	20	20
Rosenbrock	20	0	20
Ridge	20	20	20
Griewank	19	17	20

In the Schwefel function, all the models derive the optimum solutions in every trial with the small number of the evaluations. In the Rastrigin function, the search ability of model 1 is worse than the other models. That means the

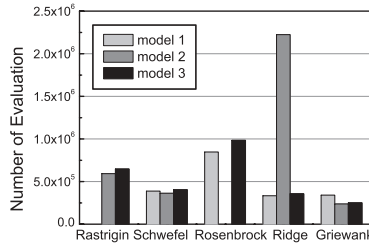


Fig. 2. Comparison of search capability between models

PCA does not help for achieving a good solution. In the Rosenbrock function that has the correlation relationship among the design variables, model 2 (where the PCA is not performed) cannot derive the optimum solutions. In the Ridge function that also has the correlation, model 2 can derive the optimum solutions but needs many evaluations compared to the other models. These results suggest that the PCA should have a positive effect on the search in these problems.

From these results, the following point is made clearer; the PCA is useful for the problems that have the correlation among the design variables but not useful for the problems that do not have the correlation. Therefore, the effect of the PCA depends on the type of the problems. On the other hand, model 3 where the PCA operations are performed in some islands but not in others is good at finding optimum solutions in every function. The Griewank function has the correlation among the design variables and has also many peaks in the landscape. Therefore, it is a difficult problem to find the optimum. Model 1 and 2 did not find the optimum solutions in some trials. On the other hand, model 3 derived the optimum in all the trials. From these results, there is a possibility that model 3 can find the optimum solutions not only in the problems whether the correlation of the design variables exists or not but also in the problems where there are many peaks in the landscape. Because of the result in this section, in the following discussions, model 3 is used.

5 Comparison of DPMBGA with UNDX + MGG

In this section, the search capability of the DPMBGA is compared with the conventional real-coded GA. The comparison real-coded GA is UNDX[4] with Minimal Generation Gap (MGG)[12,13]. In the UNDX, two new individuals are generated from three individuals. Two of the parent individuals form the main axis with the normal distribution generated on this main axis. The third parent determines the variance of the normal distribution. The child individuals are generated in accordance with this normal distribution. Using the UNDX, an effective search can be performed with the consideration of the correlation between the design variables.

MGG is one of generation alternation models. When the generation alternation occurs in the MGG, the following procedure is performed. Two of the parent individuals are chosen from the population randomly from which child individuals are generated by the crossover with the parent one time or many times. From the set of the child and parent individuals, the individuals that remain as the next generation are selected. These individuals are then substituted for the parent individuals and are backed to the total population. The MGG has the characteristics to maintain the diversity of the solutions during the search since the selection is limited to the small number of solutions.

In Figure 3, the transitions of the DPMBGA search and the UNDX+MGG model are shown. The horizontal axis shows the number of evaluations and the vertical axis shows the average fitness values in 20 trials. In the UNDX+MGG model, some of 20 trials did not get the optimum solution. The average values of the trials where the optimum solutions are derived are illustrated. These are minimization problems with smaller fitness values indicating the better solutions.

For the parameters of UNDX+MGG, there are 300 individuals for the functions with many peaks and there are 50 individuals for the functions with only one peak. The number of crossover is 100 and $\alpha = 0.5$. $\beta = 0.35$ is used.

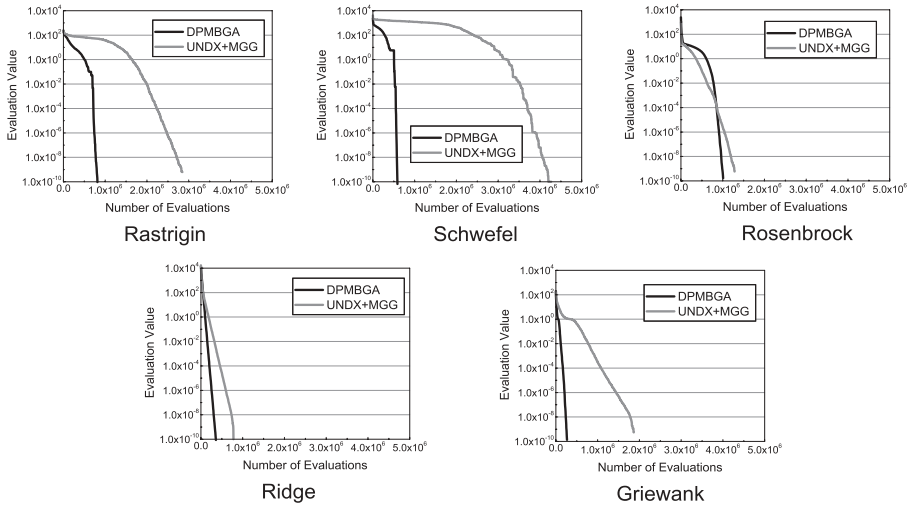


Fig. 3. History of average of evaluation values

Figure 3 indicates that the results of the DPMBGA are better than the other models in the problems whether a correlation relationship between the design variables exists or not. Therefore, it can be concluded that the DPMBGA is a useful GA for these continuous functions.

6 Discussion on Case Where PCA Does Not Work Effectively

In the former section, it is found that the PCA operation sometimes prevents the effective finding of the optimum. In the Rastrigin function, the model without the PCA operation derived better solutions than the model with the PCA. In this section, the reasoning for why the PCA does not work effectively is discussed.

One reason may be the early convergence of the solutions in the archive. The PCA uses the information from the solutions in the archive and it is important for the effective search that this archived information should reflect the information of the real landscape of the problem. If these solutions in the archive are not renewed, the proper transfer by the PCA cannot be expected to occur. Since the Rastrigin function has many sub-peaks on the landscape, it may be that many solutions are stuck in the sub-peaks. Then, the solutions of the archive are not renewed possibly preventing an effective search.

This assumption is illustrated with numerical experiments. In these experiments, the model where the PCA operation is performed in all the sub-populations is used. The target test functions are the Rastrigin function and the Rosenbrock function.

In Figure 4, the history for the renewal of the archive is shown. The horizontal axis shows the number of the evaluations and the vertical axis shows the number of the renewed individuals. From this figure, it is obvious that the number of the renewed individuals becomes small, especially in the latter part of the search in the Rastrigin function.

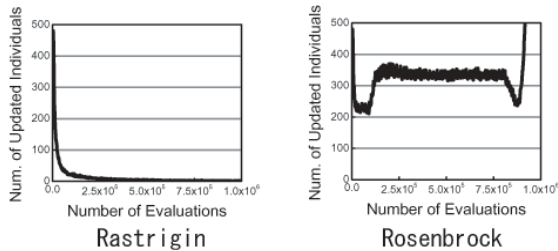


Fig. 4. History of number of updated individuals in archive of the best individuals

Conversely, in the Rosenbrock function, the number of the renewed individuals does not decrease and most of the individuals of the archive are always renewed.

In Figure 5, the history of the search is illustrated. The horizontal axis shows the number of evaluations and the vertical axis shows the average value of the evaluation in 20 trials. This is the minimizing problem; the smaller evaluation value indicates the better solution. The term "normal" in this figure indicates

the result of the normal model. "erase/10" indicates the result of the model where the archive of the individuals is eliminated every 10 generations.

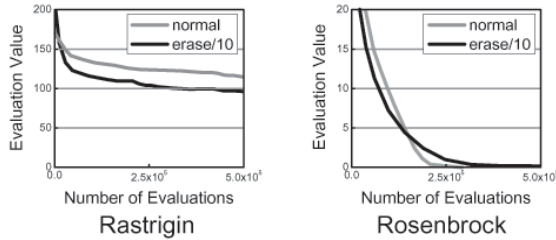


Fig. 5. History of average of evaluation values in the model in which archive is erased each 10 generation

From this figure, the result of the "erase/10" model is better than the normal model in the Rastrigin function. On the other hand, the result of the "normal" model is better than the "erase/10" in the Rosenbrock function. These results point out that the archive effect is worse in the Rastrigin function.

In conclusion, one of the reasons that the PCA operation prevents an effective search in the Rastrigin function is stagnation of the renewal of the individuals in the archive.

7 Discussion on Search Capability of DPMBGA for Functions Whose Optimum Is Located Near the Boundary

When a normal distribution is used in the crossover operation, it is often said that the real-coded GA is good at finding the solution in the problem where the optimum is located at the center of the search area but is not good at finding the solution in the problem where the optimum is located at the boundaries [14].

One solution to this problem is Boundary Extension by Mirroring (BEM) [15]. In the BEM, the solutions that violate the constraints can exist when these solutions are within certain distance. The distance is determined by the extension rate r_e ($0.0 < r_e < 1.0$).

The DPMBGA is one of the real-coded GAs that may be weak at finding the solutions in the problems where the optimum is located in the boundary. At the same time, when the optimum is located on the boundary, the probabilistic model may be different from the real distribution of the individuals. This situation may prevent an effective search. The search capability of the DPMBGA for problems where the optimum solutions is located near the boundary is discussed in this

section. The search capability of the DPMBGA is compared with the model using the BEM.

The test functions are modified to have their optimum solutions near their boundaries. The ranges of the functions are summarized in Table 3.

Table 3. Domain of objective functions

Function	Optimal solution	Domain
Rastrigin	0.0	[0 , 5.12]
Schwefel	420.968746	[-512, 421]
Rosenbrock	1.0	[-2.048, 1]
Ridge	0.0	[0, 64]

In Figure 6, the transition of the search is expressed. The horizontal axis shows the number evaluations and the vertical axis shows the average of the fitness values for 20 trials.

These figures illustrate that the proposed model where BEM is not used derives better solutions. In the proposed model, the individuals who are out of the feasible region are pulled back on the closest boundary of the feasible region. The search concentrates on the individual with a good evaluation value by the DPMBGA. When the optimum solution is on or near the boundary, the search is concentrated near the boundary. This may be a reason why the proposed model is better than the model using the BEM. Thus, the DPMBGA derives the good solution for the type of problems with constraints that are the upper and lower boundaries of the design variables.

8 Conclusions

In the DPMBGA, the correlation among the design variables is analyzed by PCA. The new individuals are generated from the probabilistic model that is formed with the individuals that have good fitness values. However, before generation, the individuals that are used for forming the probabilistic model are transferred by PCA into the space where there is no correlation among the design variables. Then the new individuals are generated and these are placed into the original space. From this operation, the generated new individuals may have the correlation among the design variables. Therefore, an effective search may be expected. At the same time, the island model is utilized for maintaining the diversity of the solutions during the search.

The DPMBGA is applied to find the optimum solutions of the test functions. Through these numerical experiments, the following four topics are made clarified.

Firstly, the DPMBGA with PCA operations is useful for finding the solutions in the test functions where there is a correlation relationship among the design

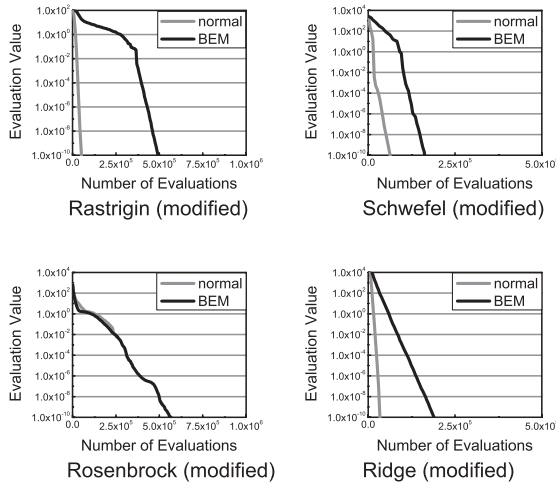


Fig. 6. History of the average of evaluation values on functions with an optimum at the edge of search space

variables. On the other hand, the PMBGA without PCA is good at finding the optimum in the functions where there is no correlation relationship among the design variables. From these results, the new model of PMBGA that is based on the distributed environment scheme where PCA is performed in only half of the subpopulations is proposed. The DPMBGA is very useful for finding the optimum in the functions whether or not there is a correlation relationship among the design variables or not.

Secondly, the results of the DPMBGA are compared to those of UNDX with MGG. This comparison shows that the DPMBGA has higher search capability.

In the DPMBGA, the Principle Component Analysis is used to analyze the correlation between the design variables. However, the PCA does not work effectively for finding the optimum solutions in some test functions. The reason for this problem is the third discussion. Numerical experiments conclude that one of the reasons the PCA operation prevents the effective search in the function is stagnation of the renewal of the individuals.

Finally, the DPMBGA is used to find the solutions in the functions where the optimums are located at the edge of the feasible region. In the DPMBGA, when the new individuals violate the constraints, the solutions are pulled back on the boundary. This operation is compared with the BEM. Numerical experiments illustrate that the operation in the proposed method is better than BEM in these test functions.

References

1. Goldberg, D.E.: Genetic Algorithms in Search Optimization and Machine Learning. Addison-Wesley (1989)
2. Pelikan, M., Goldberg, D.E., Lobo, F.: A Survey of Optimization by Building and Using Probabilistic Models. Technical Report 99018, IlliGAL (1999)
3. Larranaga, P., Lozano, J.A.: Estimation of Distribution Algorithms. A New Tool for Evolutionary Computation. Kluwer Academic Publishers (2001)
4. Ono, I., Satoh, H., Kobayashi, S.: A Real-Coded Genetic Algorithm for Function Optimization Using the Unimodal Normal Distribution Crossover. Transactions of the Japanese Society for Artificial Intelligence **14** (1999) 1146–1155
5. Takahashi, M., Kita, H.: A Crossover Operator Using Independent Component Analysis for Real-Coded Genetic Algorithms. Proc. 13th SICE Symposium on Decentralized Autonomous Systems (2001) 245–250
6. Eshelman, L. J. and Shaffer, J. D.: Real-coded Genetic Algorithms and Interval Schemata. Foundation of Genetic Algorithms Morgan Kaufmann Publishers (1993) 182–202
7. Bäck, T., Hoffmeister, F., Schwefel, H.P.: A Survey of Evolution Strategies. Proc. 4th International Conference on Genetic Algorithms (1991) 2–9
8. Hansen, N.: Invariance, Self-Adaptation and Correlated Mutations and Evolution Strategies. Parallel Problem Solving from Nature – PPSN VI, 6th International Conference, 2000, Proceedings (2000) 355–364
9. Tanese, R.: Distributed Genetic Algorithms. Proc. 3rd International Conference on Genetic Algorithms (1989) 434–439
10. Cantú-Paz, E.: Efficient and accurate parallel Genetic Algorithms. Kluwer Academic publishers (2000)
11. Miki, M., Hiroyasu, T., Kaneko, M., Hatanaka, K.: A Parallel Genetic Algorithm with Distributed Environment Scheme. IEEE Proceedings of Systems, Man and Cybernetics Conference SMC'99 (1999)
12. Sato, H., Ono, I., Kobayashi, S.: A New Generation Alternation Model of Genetic Algorithms and its Assessment. Transactions of the Japanese Society for Artificial Intelligence **12** (1997) 734–744
13. Yamamura, M., Satoh, H., Kobayashi, S.: An Analysis on Generation Alternation Models by using the Minimal Deceptive Problems. Transactions of the Japanese Society for Artificial Intelligence **13** (1998) 746–756
14. Someya, H., Yamamura, M.: Toroidal Search Space Conversion for Robust Real-Coded GA. Proc. 13th SICE Symposium on Decentralized Autonomous Systems (2001) 239–244
15. Tsutsui, S.: Multi-parent Recombination in Genetic Algorithms with Search Space Boundary Extension by Mirroring. Proc. the 5th International Conference on Parallel Problem Solving from Nature (PPSN V) (1998) 428–437

HEMO: A Sustainable Multi-objective Evolutionary Optimization Framework

Jianjun Hu^{1,2}, Kisung Seo¹, Zhun Fan¹, Ronald C. Rosenberg³, and Erik D. Goodman¹

¹Genetic Algorithms Research and Applications Group (GARAGe)

²Department of Computer Science & Engineering

³Department of Mechanical Engineering

Michigan State University, East Lansing, MI, 48824

{hujianju, ksseo, fanzhun, rosenber, goodman}@egr.msu.edu

Abstract. The capability of multi-objective evolutionary algorithms (MOEAs) to handle premature convergence is critically important when applied to real-world problems. Their highly multi-modal and discrete search space often makes the required performance out of reach to current MOEAs. Examining the fundamental cause of premature convergence in evolutionary search has led to proposing of a generic framework, named Hierarchical Fair Competition (HFC)^[9], for robust and sustainable evolutionary search. Here an HFC-based Hierarchical Evolutionary Multi-objective Optimization framework (HEMO) is proposed, which is characterized by its simultaneous maintenance of individuals of all degrees of evolution in hierarchically organized repositories, by its continuous inflow of random individuals at the base repository, by its intrinsic hierarchical elitism and hyper-grid-based density estimation. Two experiments demonstrate its search robustness and its capability to provide sustainable evolutionary search for difficult multi-modal problems. HEMO makes it possible to do reliable multi-objective search without risk of premature convergence. The paradigmatic transition of HEMO to handle premature convergence is that instead of trying to escape local optima from converged high fitness populations, it tries to maintain the opportunity for new optima to emerge from the bottom up as enabled by its hierarchical organization of individuals of different fitnesses.

1 Introduction

After a decade of intensive study on evolutionary multi-objective optimization (EMO), extensive insight has been obtained regarding convergence and the diversity of the Pareto front. Several successful multi-objective EAs have emerged, such as PESA [1], NSGA-II [4], and SPEA2 [13]. However, the capability to handle premature convergence for difficult multi-modal optimization problems has attracted insufficient attention. The performances of modern MOGAs are usually compared on some easy continuous test problems [5]. The scalability of MOEAs is focused on the scalability over the objective dimension rather than over the problem difficulty [11]. Unfortunately,

many real-world problems are characterized as highly multi-modal in highly discrete search spaces. Without careful attention to the premature convergence issues, modern MOGAs will easily fail to find the true Pareto fronts for these problems [3] and the performance comparison results will be misleading for MOGA practitioners.

Based on the research on dealing with premature convergence of single-objective EA search [9,10], a sustainable multi-objective optimization framework called HEMO (Hierarchical Evolutionary Multi-objective Optimization) is proposed in this paper. In addition to the external Pareto archive commonly found in PESA and SPEA, HEMO features hierarchically organized archives of individuals with different fitness ranks, a “workshop” subpopulation associated with each archive, and a random individual generator that continually feeds raw genetic material into the lowest-level archive. By incorporating favorable features from PESA [1], SPEA [12], and HFC (Hierarchical Fair Competition) EA model [9, 10], and extending ideas from the improved NSGA-II [3], this framework promises to have strong capability to avoid premature convergence in EMO and thus to constitute a sustainable search procedure for solving difficult real-world problems.

2 Convergence, Diversity, and Premature Convergence in EMO

From the first generation of modern MOGAs such as NSGA, SPEA, and PAES to the improved versions like NSGA-II, SPEA2 and PESA, much attention is allocated to diversity maintenance of the Pareto front by estimating the density of individuals along the Pareto front (SPEA2, NSGA-II, PESA), ensuring sufficient selection pressure in special cases (SPEA2), utilization of elitism (NSGA-II), and other efforts to obtain computational efficiency. However, the diversity along the Pareto front is different from the diversity required for avoiding premature convergence, which is labeled as lateral diversity in [3]. The capability to maintain lateral diversity varies widely among MOGAs, which contributes much to the performance differences for different test problems.

2.1 Performance Comparison of Modern MOGAs

In terms of lateral diversity maintenance, PESA, NSGA-II, and SPEA2 have different strategies, which largely determine their advantages and disadvantages. Among the three, PESA is the greediest algorithm. By selecting the mating pool only from the currently discovered Pareto front, it is on one extreme of elitism and depends strongly on the mutation operator for exploration. As a result, PESA has the fastest convergence speed, but is only good for continuous, relatively simple problems. It is shown to be inferior on the T4 test function, for example, which is a continuous multi-modal problem [1]. It can be expected that the uncontrolled, extreme elitism of PESA will make PESA unusable for highly discrete multi-modal problems.

By maintaining a constant size of the archive (parent) population, SPEA2 and NSGA-II allow the persistence of dominated individuals in cases in which the non-dominated individuals do not fill the archive population. So for some continuous multimodal test functions such as QV and KUR [14], SPEA2 and NSGA-II are shown to

be able to achieve good performance. However, for other multimodal problems in which there are too many non-dominated individuals, SPEA2 and NSGA-II will always select mating individuals from the current Pareto front, in effect degrading to the extreme elitism case of PESA. This uncontrolled elitism makes NSGA-II without mutation perform poorly on difficult multimodal problems such as ZDT4, ZDT6, and Griewank [3]. As high mutation rate is not the solution to premature convergence, even with mutation, NSGA-II will fail on other difficult multimodal problems. To explicitly maintain the dominated individuals to promote lateral diversity, Deb and Goel [3] proposed the controlled elitism NSGA-II, which turns out to be very successful. The basic idea is to allocate a predefined distribution of individuals to each current Pareto front in NSGA-II. However, as the fronts in NSGA-II usually move in clusters to better regions of the objective space based on limited evaluations (for minimization problems), there is increasing risk that all fronts get trapped in local Pareto fronts, and gradually, the exploratory capability will be lost. This is attributable to the fitness assignment scheme of NSGA-II, which is based on the relative fronts, and on the convergent nature of conventional GAs.

2.2 Premature Convergence and the Issue of Exploitation vs. Exploration

To a large extent, the premature convergence problem in EMO is similar to of the situation in single-objective EAs. Most previous studies attributed the cause of premature convergence to the loss of diversity of the population and proposed various diversity-oriented approaches to increase the population diversity by “brute force.” Representative methods include increasing the mutation rate; introducing random individuals into highly converged populations, and using diversity-detection and increasing techniques. All these methods are shown to ameliorate only partially the premature convergence problem. For example, in genetic programming, a high mutation rate usually destroys the good solutions evolved and, despite the diversity of the population, no progress can be made with this “brute-force” diversity maintenance.

Actually, the loss of diversity is only a symptom of premature convergence. The more fundamental reason is, instead, the loss of exploratory capability. In single-objective EAs, the absolute average fitness of the whole population is constantly increasing as the result of fitness-biased selection. The consequence is that “new explorer” individuals (i.e., early individuals in a new region of the search space), whether the offspring of mutation or crossover or randomly generated, find it increasingly hard to survive, since these explorer individuals usually have low fitness until sufficient exploration in the new search region is conducted. Rare high-fitness “explorer” individuals, due to their sparseness, will also have high risk of getting lost as the result of sampling bias in parent selection toward more crowded areas, similar to the analysis in [2]. To fight against this “unfair competition” among highly evolved individuals and new “explorers”, there must be some mechanism to protect new explorers. This is achieved to some extent by widely used approaches such as fitness sharing and crowding. However, using horizontal expansion in the genetic space, these techniques usually suffer from the problem of balancing a limited population size against a huge number of local optima in difficult multi-modal problems.

Another perspective on premature convergence can be obtained by examining building block concepts. The evolution process is widely seen as a process in which different building blocks become co-adapted to achieve higher and higher fitness by mixing and mutation. The higher the fitness of an individual, the stronger the coupling of its subcomponents, and the more difficult to make large modification of the highly evolved individual without destroying the co-adaptation relationship. So the exploratory capability decreases with increasing fitness of the population. It is similar to the Cambrian explosion in the evolution of living organisms, during which most existing species (body plan innovations) were created. However, by allocating all the search effort to highly evolved individuals, without control, conventional EAs essentially discard the low-fitness evolution stages after limited mixing experiments, and thus are essentially convergent algorithms.

NSGA-II, with its controlled elitism [3], is one of the first algorithms that pays special attention to dominated inferior individuals. However, while derived from the conventional EA framework, the improved NSGA-II still suffers from the tendency that all individuals in the fronts are moving toward the best yet-discovered regions of the objective space, based on limited mixing experiments, and the components are increasingly co-adapted to each other (Fig. 1). As the result, the exploratory capability of the population is gradually lost and premature convergence occurs. The distribution of individuals to the relatively diverse fronts is insufficient to avoid this kind of premature convergence.

Based on the analysis above, it turns out to be important to maintain intermediate individuals and to make the building block mixing process occur at all fitness levels. This naturally provides a mechanism to ensure fair competition and protects “explorer” individuals. At the same time, to reduce the large population size requirement

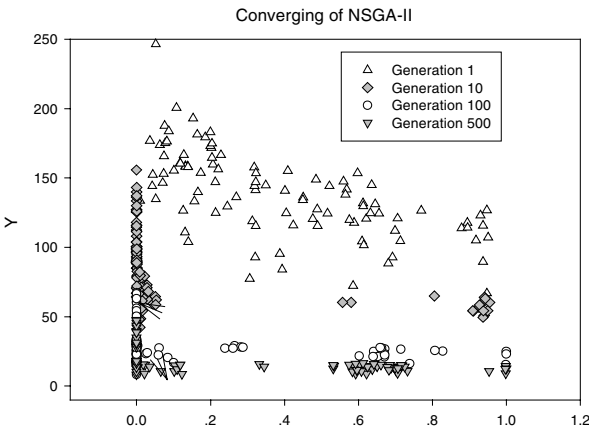


Fig. 1. The population of NSGA-II moves in clusters leaving the initial low objective value space and converging to the promising space. Even the maintenance of a predefined proportion of population into all fronts, but in the whole these fronts are converging to local areas, thus making it incapable to maintain the explorative capability in the long run.

[7] for difficult problems, it is desirable to continuously introduce random individuals into the lowest fitness levels to provide the required building blocks, rather than depending on a large initial population to identify them, as is done in messy GA [8]. This suggests the assembly line structure of the subpopulations in the HFC framework proposed in [9]. The HEMO framework is thus an extension of HFC to multi-objective optimization, incorporating ideas from SPEA, PESA and the improved NSGA-II.

2.3 Combining Ideas in SPEA, PESA, and the Improved NSGA-II

The different performances of SPEA, PESA and NSGA-II over different test functions reflect the unique, positive features of each approach. HEMO will attempt to capture some features of each of these. Specifically, the maintenance of an external Pareto archive and the breeding population first proposed in SPEA [12] is employed in the HEMO framework, but extended so that both the Pareto archive and archives of intermediate individuals are maintained. The elitism in the Pareto front update is supported by low-level HFC archives, as explained in the next section. For density estimation, the grid-based methods in PESA [1] are used, which are naturally suited for the absolute division of the objective space as required by the HEMO framework. This grid-based method is also demonstrated to have excellent performance in maintaining Pareto front diversity [11]. The distribution of individuals into all fronts in the improved NSGA-II is extended to all fitness levels.

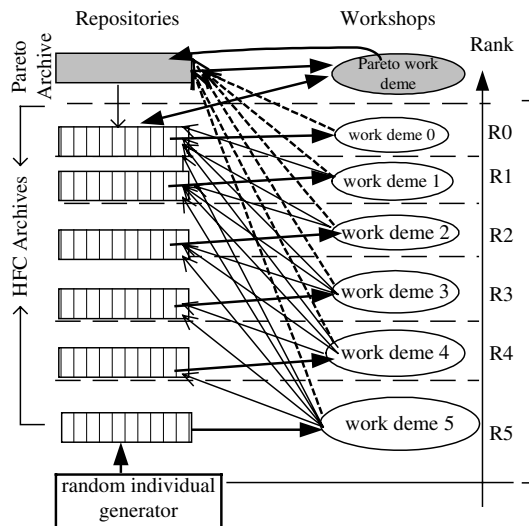


Fig. 2. The assembly line structure of the HEMO Framework. In HEMO, repositories are organized in a hierarchy with ascending fitness level (or rank level in the objective space as employed in this paper). Each level accommodates individuals within a certain fitness range (or belonging to a given rank level) as determined by the admission criteria.

3 HEMO: Hierarchical Evolutionary Multiobjective Optimization

Based on the analysis of the fundamental cause of premature convergence and drawing ingenious ideas from previous successful MOGAs, we propose the HEMO framework for difficult multi-objective problems in which the avoidance of premature convergence is of great concern. Essentially, it is an extension of PESA enhanced with the continuing search capability of HFC. In addition to the Pareto archive and the Pareto workshop population, a succession of archives for maintaining individuals of different fitness levels is added to allow mixing of lower- and intermediate-level building blocks. A random individual generator is located at the bottom to feed raw genetic material into this building block mixing machine continually. The structure of HEMO is illustrated in Fig. 2:

The HEMO algorithm proceeds as follows:

1) Initialization

- Determine the number of levels ($nLevel$) into which to divide the objective space for each objective dimension. Determine the grid divisions ($nGrid$) as in PESA. Note that $nLevel$ is different from $nGrid$. The first one is used to organize intermediate individuals into the hierarchical archives, while the latter is used to estimate the density of individuals.
- Determine the population sizes of the Pareto archive, HFC archive and corresponding workshop demes. The distribution of population sizes among archives (workshop demes) can be determined separately or using some special distribution scheme like the geometric distribution in (Deb and Goel, 2000).
- Initialize the workshop demes with random individuals. The archives are empty at the beginning.
- Evaluate all individuals and calculate the crowding factor of each individual according to the hyper-grid approach in PESA.
- Calculate the fitness range of each objective dimension for all individuals in the whole population:

$$[f_{\min}^i, f_{\max}^i] \text{ where } i = 0, \dots, ObjDim - 1$$

- Divide the fitness range into $nLevel$ Levels. For all individuals, calculate the objective ranks for each objective dimension, $r_i, i = 0, \dots, ObjDim - 1$, $r_i \in [0, nLevel - 1]$;
- For each individual, calculate its fitness rank = the average rank over all objective

$$\text{dimensions of each individual. } r^f = \frac{1}{ObjDim} \sum_i^{ObjDim-1} r_i$$

- Migrate (move out) individuals in the workshop demes to the corresponding HFC archives according to their fitness ranks r^f . Then add all non-dominated individuals of each workshop deme to the Pareto archive. There are two cases possible during these migrations. If the target archive is full, we will replace a selected individual according to the Pareto archive and HFC archive update procedures de-

scribed below; else, we simply add the migrating individual into the target repository.

2) Loop until meeting stopping criterion

A steady state evolutionary model is used in the HEMO framework. First,

- Compute the breeding probability of each workshop deme of the HFC rank levels. This is calculated as follows:

$$pBreed^l = \frac{Popsiz\text{ of workshop deme of level } l}{\sum_{k=1}^{nLevel-1} Popsiz\text{ of workshop deme of level } k}$$

These probabilities can instead be dynamically adjusted irrespective of the workshop deme sizes. These probabilities determine the allocation of search effort to each level, thus determining the greediness of the algorithm.

- Decide whether to do Pareto workshop breeding or HFC workshop deme breeding by probability $pParetoBreed$. If setting $pParetoBreed = 1$, then HEMO reduces to an algorithm similar to PESA. This parameter is used to control the greediness of the Pareto search.

If Pareto workshop breeding is to be done:

- Decide whether or not to do crossover according to its probability. Mutate each gene of the offspring with probability $pGeneMutate$.
- Select parents from the Pareto archive using tournament selection based on the crowding factors of individuals. The less crowded, the more chance an individual will get selected. When selecting parents for crossover or mutation, the probability to select only from the Pareto archive is $pSelectFromPareto$. The probability to select a second parent from the rank 0 HFC Archive is $1 - pSelectFromPareto$. When there is only one individual in the Pareto archive, the second parent for crossover is selected from the highest HFC archive.
- Create an offspring (two in crossover) and add it to the Pareto workshop deme. If the Pareto workshop deme is not full, simply add the new candidate to it; else, trigger the **Pareto Archive Update Procedure**. Then a migration process will move individuals of each HFC archive to their new qualified HFC archives because of the update of the objective ranges.

If HFC workshop deme breeding is to be done:

- Decide at which level (L) breeding will occur according to the probability $pBreed^l$
- Decide whether or not to do crossover according to its probability. Mutate each gene of the offspring with probability $pGeneMutate$.
- Select parents from the HFC archive of level L by tournament selection based on the crowding factors. The lower the crowding factor, the higher the probability to be selected. If there is only one parent in the current HFC archive, then the second parent will be selected from the next lower archive.
- Create an offspring (two in crossover) and add it to the workshop deme. If the workshop deme is not full, simply add to the end; else, trigger the **HFC Archive Update Procedure** and the **Pareto Archive Update Procedure**.

- With low probability $pRandomImport$, update $perRandomIn$ percent of the individuals of the lowest HFC archive with random individuals.

Pareto Archive Update Procedure ()

- Screen out the non-dominated individuals in the workshop deme.
- Update the objective ranges of the whole population with the non-dominated individuals.
- Recalculate the crowding factors of all individuals of the selected non-dominated individuals and the individuals in the Pareto archive.
- Update the Pareto archive with the selected non-dominated individuals. If the Pareto archive is full, truncate it by removing individuals with higher crowding factors.
- Empty the Pareto workshop deme.

HFC Archive Update Procedure ()

- Update the objective ranges of the whole population and recalculate the fitness ranks of all individuals in the workshop demes.
- Migrate individuals in the current HFC archives into their corresponding new levels. If the target HFC archive is full, replace an individual selected by tournament selection. The more offspring an individual produces, the higher the probability it will be replaced.
- Update the HFC archives with the individuals in the workshop deme. If the target HFC archive is full, replace an individual selected by tournament selection. The bigger the crowding factor is, the higher probability it will have to be replaced. Note that only higher archives are updated with the current workshop deme (unidirectional migration policy)

4 Experiments and Results

In this section, two multi-modal test functions are selected to demonstrate the exploratory capability of HEMO to avoid premature convergence. Here, HEMO is only compared to PESA, since HEMO is most closely derived from PESA.

1) Multi-objective Rastrigin's problem (ZDT4)

$$ZDT4: \begin{cases} \text{Minimize } f_1(x) = x_1 \\ \text{Minimize } f_2(x) = g(x)[1 - \sqrt{x_1 / g(x)}] \\ g(x) = 91 + \sum_{i=2}^{10} [x_i^2 - 10 \cos(4\pi x_i)] \\ x_1 \in [0, 1], x_i \in [-5, 5], i = 2, \dots, 10. \end{cases}$$

2) Multiobjective Griewank Problem (GWK)

GWK problem is constructed by replacing $g(x)$ in 1) with Griewank's function, where

$$g(x) = 2 + \sum_{i=2}^{10} x_i^2 / 4000 - \prod_{i=2}^{10} \cos(x_i / \sqrt{i}), x_1 \in [0, 1],$$

$$\text{where } x_i \in [-512, 511] \quad i = 2, \dots, 10$$

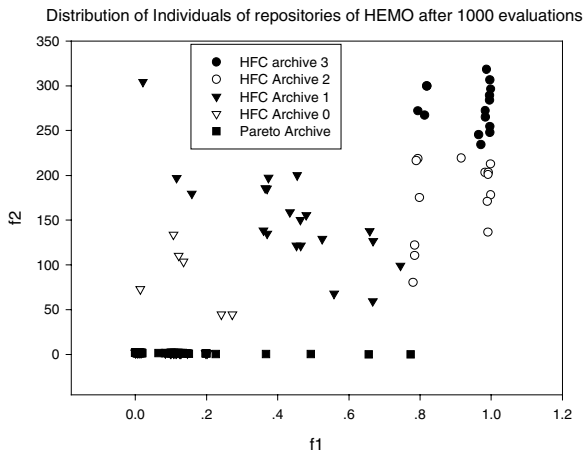


Fig. 3. Distribution of individuals over the objective space of GWK in HEMO after 1000 evaluations. Compared with Fig. 1. of NSGA-II, the difference is that the archive population of NSGA-II is drifting and has the risk of converging to a local area. HEMO has less tendency to converge, since it maintains representative individuals at many levels in the objective function space and continuously introduces new genetic material, thus providing the fitness gradient for new optima emerge in a bottom-up way.

Fig. 3 illustrates the distribution of individuals of HEMO during the evolutionary process. It is clear that HEMO works by trying to expand the individuals in its repositories evenly across the objective space, rather than by converging to the early-discovered high-fitness areas. This provides the necessary fitness gradient for new optima to emerge in a bottom-up way, from the bottom level HFC archive and workshop subpopulations.

The robustness of PESA and that of HEMO are compared by examining the relationship of performance and the mutation rates applied to each *type double* gene after crossover. We use the statistical comparison method of [1] to compare the Pareto fronts obtained with different mutation rates by PESA and HEMO (Table 1). Cells in the upper right triangle of the table hold the comparison results of different mutation rates for HEMO, while the lower left (shaded cells) are for PESA. The first entry in each cell represents the percentage of Pareto front solutions obtained with the row's mutation rate that are non-dominated, with 95% confidence, by the solutions obtained with the column's mutation rate. The second entry in each cell, similarly, shows the percentage of Pareto front solutions obtained with the column's mutation rate that are non-dominated, with 95% confidence, by the solutions obtained with the row's mutation rate. From [3], we know that for test function ZDT4, NSGA-II fails to find the true Pareto front. This is also the case for PESA, as illustrated in the first column. PESA without mutation is worse than any PESA configuration with mutation. It is also suggestive that for PESA, the performance varies greatly with different mutation rates, achieving best performance here with a mutation rate of 0.12. In contrast, HEMO is more robust over mutation rates. The performance difference with no muta-

tion is not much different from that with mutation rate 0.16. It is clear that results with mutation rate 0.16 is much better than 0.04, 0.08, 0.12. However, experiments with more evaluations show that they all achieve similar results.

Table 1. Comparison of the robustness of PESA (in shaded cells) and HEMO with test function ZDT4. First entry in each cell is percentage of solutions obtained with row’s mutation rate that are not dominated by those obtained with the column’s mutation rate, and vice versa for the second entry. PESA can be seen to depend strongly on mutation to maintain its exploratory capability. It is very sensitive to the mutation rate, for which the optimal value is hard to know in advance. HEMO is much less sensitive to the mutation rate, since it doesn’t depend on mutation to maintain the explorative capability.

Mutation Rate	0.00	0.04	0.08	0.12	<u>0.16</u>
0.00		99 100	99.5 100	99.7 100	97.5 100
0.04	100 50		99.8 100	99.9 100	0.5 100
0.08	100 7.3	100 99.7		100 100	2.4 99.7
0.12	100 3.2	100 50.0	100 70.6		2.4 99.9
0.16	100 7.3	100 50.1	98.7 100	100 100	

We also compared the best performance of PESA (mutation rate 0.12) with that of HEMO (mutation rate 0.16) for the same number (10,000) of evaluations (Table 2) . For ZDT4, the Pareto front found by HEMO was much better than PESA found. In the case of GWK, HEMO had limited advantage over PESA. The reason is that the statistical comparison procedure used here [6,1] compares the merged Pareto fronts found during 20 runs. PESA with different random seeds may converge to different points in the objective space, which on the whole comprise a good Pareto front. However, PESA is a poor opportunist in the sense that for both the ZDT4 and GWK functions, PESA converges to only one or two Pareto solutions in 6 or 7 runs of a total of 20. In contrast, HEMO always obtains diversified solutions in the Pareto archive.

5 Conclusions and Future Work

Current MOEAs still suffer from their convergent nature inherited from the conventional EA framework. The loss of population diversity turns out to be only a symptom of the phenomenon of premature convergence. Maintenance of exploratory capability is central to ensuring sustainable evolutionary search. A new evolutionary multi-objective framework named HEMO is introduced, featuring: a hierarchical organization of repositories of individuals of different fitness levels (defined as the composite objective ranks in the divided objective space), the continual introduction of raw genetic material at the bottom evolutionary level, and hyper-grid-based density estimation. Two experiments are reported to show the sustainable search capability of HEMO, demonstrated along with its robustness over a variety of mutation rates, as compared to PESA. The paradigmatic transition in handling premature convergence

Table 2. Opportunistic PESA and robust HEMO. HEMO obtains a much better Pareto front for ZDT4 and a small advantage for GWK. However, for each independent run, the frequency (in last row) with which PESA converges to one or two Pareto solutions is around 33%, while HEMO seldom converges to local Pareto fronts.

Test Function	ZDT4		GWK	
	PESA	HEMO	PESA	HEMO
% Non-Dominated Pareto Solutions in 20-Run Ensemble	0.3% by HEMO	100% by PESA	47.3% by HEMO	53.7% by PESA
Premature Convergence Frequency	6/20	0/20	7/20	0/20

from HEMO is: instead of trying to escape local optima from within converged, high-fitness populations, the continuing EA framework (as represented, for example, by HFC and HEMO here) ensure the opportunity for new optima to emerge from the bottom up, enabled by the hierarchical organization of individuals by fitness.

By combining features from PESA and SPEA and extending the ideas in the NSGA-II with controlled elitism, and including the HFC organization, HEMO is expected to be well suited for difficult multi-modal real-world problems in which premature convergence is of great concern. We also expect that HEMO will be especially advantageous in multi-objective genetic programming, where the highly multi-modal and discrete fitness landscape often makes modern MOEAs such as PESA fail by converging prematurely to a local Pareto front. It is interesting to sort the MOEAs by their capabilities to handle premature convergence. From the lowest to highest, we have PESA $\rightarrow$ SPEA2 $\rightarrow$ NSGA-II $\rightarrow$ NSGA-II with controlled elitism $\rightarrow$ HEMO, each improving the previous one by paying more attention to the non-inferior dominated individuals. However, HEMO differs from all the others in its continuing search nature without premature convergence, while the others are all based on the traditional, convergent EA framework.

As a generic framework, HEMO is easily applicable to other modern MOEAs such as SPEA-2 and NSGA-II. To improve running efficiency, a better density estimation method is needed. The scheme for organizing individuals by rank levels can also be improved. In addition, to distribute the individuals of the repositories more evenly in the objective space, the HFC archive update scheme needs further refinement. Especially, extensive comparative experiments with NSGA-II with controlled elitism and other MOEAs are required to fully demonstrate the potential of HEMO.

Acknowledgements. We are grateful to Dr. D. W. Corne and J. D. Knowles for their helpful PESA code and the statistical Pareto front comparison code. This work is supported by the National Science Foundation through grant DMII 0084934.

References

1. Corne, D. W., Knowles, J.D. and Oates, M. J. The Pareto Envelope-based Selection Algorithm for Multiobjective Optimization, In Marc Schoenauer, Kalyanmoy Deb, Guter Rudolph, Xin Yao, Evelyne Lutton, J. J. Merelo and Hans-Paul Schwefel (editors), *Proceedings of the Parallel Problem Solving from Nature VI Conference*. Springer. (2000): 839–848
2. Corne, D.W., Jerram, N.R., Knowles, J.D., and Oates, M.J. PESA-II: Region-Based Selection in Evolutionary Multiobjective Optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, , Morgan Kaufmann Publishers. (2001): 283–290
3. Deb K., Goel, T. Controlled elitist non-dominated sorting genetic algorithms for better convergence, in: E. Zitzler, K. Deb, L. Thiele, C.A. Coello Coello, D. Corne (eds.) *Proceedings of the First International Conference on Evolutionary Multi-Criterion Optimization*, Springer, Berlin (2001) : 67–81.
4. Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T. A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2) (2002): 182–197.
5. Deb, K., Thiele, L., Laumanns, M. and Zitzler, E. Scalable Multi-Objective Optimization Test Problems. In: *Proceedings of the 2002 IEEE Congress on Evolutionary Computation (CEC 2002)*, IEEE Press, Piscataway (NJ). (2002)
6. Fonseca, C.M. and Fleming, P.J. On the Performance Assessment and Comparison of Stochastic Multiobjective Optimizers. In Voigt, H-M., Ebeling, W., Rechenberg, I. And Schwefel, H-P, editors. *Parallel Problem Solving From Nature-PPSN IV*., Springer. (1995): 584–593.
7. Goldberg, D. Sizing Populations for Serial and Parallel Genetic Algorithms. In *Proc. Third International Conf. on Genetic Algorithms*. Morgan Kaufmann. (1989).
8. Goldberg, D. E. *The Design of Innovation: Lessons from and for Competent Genetic Algorithms*, Boston, MA: Kluwer Academic Publishers, (2002).
9. Hu, J., Goodman, E.D. Hierarchical Fair Competition Model for Parallel Evolutionary Algorithms. In *Proceedings, Congress on Evolutionary Computation, CEC 2002*, IEEE World Congress on Computational Intelligence, Honolulu, Hawaii. (2002)
10. Hu, J., Goodman, E. D., Seo, K., Pei, M. 2002. Adaptive Hierarchical Fair Competition (AHFC) Model for Parallel Evolutionary Algorithms. In *Proceedings of the Genetic and Evolutionary Computation Conference*: 772–779, GECCO-2002, New York.
11. Khare, V., Yao, X. and Deb, K. Performance Scaling of Multi-objective Evolutionary Algorithms. *Kangal Report No. 2002009*. (2000)
12. Zitzler, E. and Thiele, L. Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach. *IEEE Transactions on Evolutionary Computation*, 3(4), (1999):257–271.
13. Zitzler, E., Laumanns, M. and Thiele, L. *SPEA2: Improving the Strength Pareto Evolutionary Algorithm*. Technical Report 103, Computer Engineering and Communication Networks Lab (TIK), Swiss Federal Institute of Technology (ETH) Zurich, Gloriastrasse 35, CH-8092 Zurich. (2001)
14. Zitzler, E., Deb, K., and Thiele, L. Comparison of Multiobjective Evolutionary Algorithms: Empirical Results. *Evolutionary Computation*, 8(2) (2000): 173–195.

Using an Immune System Model to Explore Mate Selection in Genetic Algorithms

Chien-Feng Huang

Modeling, Algorithms, and Informatics Group (CCS-3),
Computer and Computational Sciences,
Los Alamos National Laboratory, MS B256,
Los Alamos, NM 87545, USA
cfhuang@lanl.gov

Abstract. When Genetic Algorithms (GAs) are employed in multimodal function optimization, engineering and machine learning, identifying multiple peaks and maintaining subpopulations of the search space are two central themes. In this paper, an immune system model is adopted to develop a framework for exploring the role of mate selection in GAs with respect to these two issues. The experimental results reported in the paper will shed more light into how mate selection schemes compare to traditional selection schemes. In particular, we show that dissimilar mating is beneficial in identifying multiple peaks, yet harmful in maintaining subpopulations of the search space.

1 Introduction

In the setting of multimodal function optimization, engineering and machine learning, there are two important issues when a GA is used: (1) how fast can a GA discover one or several peaks? And (2) can a GA maintain diverse subpopulations in different parts of the search space?¹ In this paper, we intend to use the mate-selection framework proposed in [7] and present the research work for investigating these two themes. In [7], it was shown that mate selection plays a crucial role in GA's search performance. In a nutshell, the dissimilarity-based mate selection schemes facilitate locating a single, best-so-far solution at the expense of generating lethal offspring; and the similarity-based mate selection schemes enhance selection pressure toward highly-fit individuals such that the GA's population converges rapidly to a certain region of a fitness landscape. As such, for the first question, we would expect the dissimilarity-based mate selection to improve the GA's search performance with respect to that metric. On the other hand, our empirical results so far have showed that simple GAs with the mate selection schemes are all subject to convergence (i.e., the simple

¹ The first issue was briefly discussed in [7]. For the second issue, there are some practical problems where maintaining subpopulations are critical. An example is the application of genetic approach to decentralized PI controller tuning for multivariable processes in [12].

GAs cannot maintain subpopulations). Thus for the second question, we intend to employ Smith et al.'s immune system model [11], which was shown to be able to maintain diverse subpopulations, in order to offer additional insights into how the mate selection schemes compare to traditional selection schemes. In particular, we are interested in studying how different mate choices affect the capability of Smith et al.'s approach for maintaining subpopulations. Since it has been shown, in [7], that the dissimilar mating mechanisms are harmful in the sense of producing more useless hybrids, we expect that such mating preferences will reduce the proportions of individuals in subpopulations. If so, the next question would be to study if reducing the probability of dissimilar mating (or increasing the probability of similar mating) can improve the capability for maintaining subpopulations.

This paper presents the preliminary results we obtained while investigating the role of mate selection in the two issues discussed above. Before delving fully into this paper, however, it is important to briefly review Goldberg and Richardson's fitness sharing mechanism [3] that serves as an idealized approach for maintaining population diversity, and present Smith et al.'s immune system model to discuss how it implements a form of implicit fitness sharing so as to facilitate formation of subpopulations. We then summarize the relevant framework for studying mate selection proposed in [7]. Section 3 presents experimental results that answer the two questions mentioned above. Finally, this paper is concluded with the insights obtained for the mate selection schemes and future research lines.

2 Relevant Work in Prior GA Research

2.1 Fitness Sharing

Fitness sharing was an idea motivated by Holland's discussion [6] in which the number of individuals occupying a niche is limited to that niche's carrying capacity. Goldberg and Richardson [3] then introduced a fitness sharing mechanism that induces population diversity by penalizing individuals for the presence of similar individuals in the population. The technique they proposed was shown to be an effective method for maintaining subpopulations over several high-fitness regions of the search space. However, it has two serious limitations: (1) the peaks must be equidistant or nearly so, and (2) setting σ_s (a critical parameter in the fitness sharing scheme that represents a cutoff distance, beyond which no sharing will occur) requires knowledge about the number of peaks in the search space. These limitations arise from the fact that fitness sharing is defined explicitly.

To avoid the difficulty of appropriately choosing σ_s Smith, Forrest and Perelson [11] introduced an algorithm that does not require explicit construction of the sharing function. Their approach can *implicitly* achieve fitness sharing that discovers for itself how many peaks are in the search space (including the case of not equally spaced peaks), and allocate trials appropriately. The idea is to use the metaphor of biological immune systems which can maintain the diversity

needed for it to detect multiple antigens. Then the GA, combined with the immune system idea, effectively distributes the population over several high-fitness areas of the search space.

2.2 Binary Immune System Model

The immune system model considered in this paper is based on a model introduced by Farmer et al. [1], where both antigens and antibodies are represented by binary strings. It is a simplification from the real biology in which genes are specified by a four-letter nucleic acid alphabet and recognition between antibodies and antigens is based on their three-dimensional shapes and physical properties. However, this abstract model of binary strings is rich enough for exploring how a relatively small number of recognizers (the antibodies) can evolve to recognize a much larger number of different patterns (the antigens).

In this binary immune system model, recognition is evaluated through a string matching procedure. The antigens are considered fixed, and a population of N antibodies is evolved to recognize the antigens using a GA. For any set of antigens, the goal is to obtain an antibody *cover*—a set of antibodies such that each antigen is recognized by at least one antibody in the population. Maintaining diverse antibodies is crucial for obtaining a cover [11].

An antibody is said to match an antigen if their bit strings are complementary (maximally different). Since each antibody may have to match against several different antigens simultaneously, we do not require perfect bit-wise matching. Many possible match rules are plausible physiologically (See [10] for examples). The degree of match is quantified by a class of match score functions $M : \text{Antigen} \times \text{Antibody} \rightarrow \mathbb{R}$. For instance, M can simply count the number of complementary bits or M can identify contiguous regions of complementary bitwise matches within the string.

Smith et al. [11] adopted a model in which a fixed set of antigens is given, and the antibodies are initialized either to be completely random (to see if the GA can learn the correct antibodies) or initially given the answer by setting the population to include the correct antibodies (to test the stability of the answer). Their mechanism for fitness scoring is as follows:

1. A single antigen is randomly selected from the antigen population.
2. From the population of N antibodies a randomly selected sample of size σ is taken without replacement.
3. For each antibody in the sample, match it against the selected antigen, determine the number of bits that match, and assign it a match score.
4. The antibody in the sample population with the highest match score is determined. Ties are broken at random.
5. The match score of the winning antibody is added to its fitness. The fitness of all other antibodies remains unchanged.
6. This process is repeated for C cycles (typically one to three times the number of antibodies).

In this scheme, since an antibody's fitness is increased only if it is the best matching antibody in the sample, the fitness values of antibodies are interdependent. In [11] Smith et al. showed analytically how this procedure implicitly embodies fitness sharing. Furthermore, Forrest et al. [2] reported that this scheme can maintain subpopulations of antibodies that cover a set of antigens.

2.3 Mate Selection Schemes

Based on the idea of "assortative mating" used in biology, [7] proposed a framework to investigate the role of mate selection in GA's search power.² Simply stated, the goal was to shed more light into how specific mate selection schemes compare to traditional selection schemes. In case of similar mating, similar individuals are chosen for mating; in case of dissimilar mating, dissimilar individuals will mate with each other. That is, the selection-for-mating step of a simple GA [9] is modified as:

During each mating event, a binary tournament selection³—with probability one the fitter of the two randomly sampled individuals is chosen—is run to pick out the first individual, then choosing the mate according to the following schemes:

Tournament Selection (TS): Run the binary tournament selection again to choose the mate.

Tournament Dissimilar Mating (TDM): Run the binary tournament selection two more times to choose two candidate partners; then the one more dissimilar to the first individual is selected for mating.

Tournament Similar Mating (TSM): Run the binary tournament selection two more times to choose two candidate partners; then the one more similar to the first individual is selected for mating.

Random Dissimilar Mating (RDM): Randomly choose two candidate partners; then the one more dissimilar to the first individual is selected for mating.

Random Similar Mating (RSM): Randomly choose two candidate partners; then the one more similar to the first individual is selected for mating.

We use the Hamming distance as the similarity metric. Notice that in the mate selection schemes above if the two candidates are of the same Hamming distance to the first individual, then one of them is randomly selected.

In the five approaches above, the first individual is always sampled by the regular tournament selection. For TDM and TSM, there are two ways to affect an individual's probability of being selected. The first results from the fitness evaluation explicitly defined by a given test function. The second is from the preference of each individual over other individuals that possess certain characteristics. The two sources complicate the probability of an individual being

² See [7] for a comprehensive literature review of the relating mate-selection work in prior GA research and a detailed discussion on why the framework was proposed.

³ Tournament selection is employed here for low computational cost.

selected for actual mating. It is expected that tournament selection contributes more selection pressure toward highly-fit individuals, and the mate preference refines the searching for mates. As for RDM and RSM, the selection pressure is reduced by removing the tournament selection acting upon the candidate mates. The only source that affects the mate selection probability is precisely the mating preference, which exerts a selection pressure on the population based on genotype.

3 Experimental Results

To illustrate the effects of mate selection on the subpopulation-maintaining ability of Smith et al.'s immune system algorithm (we call it the diversity algorithm from here on), we use a simple example in which antigen populations cannot be matched by a single antibody type. Consider an antigen population that is composed of 50% 000...000 (all 0's) and 50% 111...111 (all 1's). In order for an antibody population to recognize these antigens, there would need to be some antibodies that are all 1's and others that are all 0's. Thus, a solution to this problem requires the GA to maintain two different solutions simultaneously. This is an example of a "multiple peaks" problem because there are two incompatible solutions that are maximally different. Typically, on multiple-peaks problems it is difficult for simple GAs to distribute the population over several peaks of a fitness landscape (two different subpopulations of antibodies that match two types of antigens, in this case). This is because the selection pressure in a simple standard GA usually entails strong convergence tendency to only one peak. Even without selection pressure, genetic drift due to sampling error can still lead the GA to converge on one of the peaks [4].

Forrest et al. [2] reported in their numerical experiments that the GA with the diversity algorithm can effectively avoid strong convergence to one peak and distribute the population over multiple peaks. As has been discussed in the beginning of this paper, we expect the mate selection schemes play an important role in maintaining subpopulations. In particular, our objective is to address the following questions concerning the capability of the GA, along with Smith's algorithm, for maintaining subpopulations:

- Can the GA with different mate selection schemes maintain stable subpopulations of antibodies for recognizing different antigens, or does it always converge on one peak? If it can maintain diverse subpopulations, then
- Is the proportion of antibodies in each subpopulation being affected by different mating preferences?⁴
- Do different mating preferences have influence on the discovery time of antigens?

In light of pattern-recognition, Forrest et al. [2] pointed out that the immune system needs to recognize bacteria partially on the basis of the existence of

⁴ How many antibody representatives must be in the population for an antigen to be identified is critical. See [2] for a detailed discussion.

certain unusual molecules that are inherently different from human cells, since many bacteria have cell walls made from polymers that do not occur in humans. With this as motivation, we study the GA’s ability to detect common patterns (building blocks) in the antigen population and adopt the building-block idea in [6] to calculate fitnesses of antibodies.

Table 1. Building blocks of antigens

$b_1 =$	11111*****	$s_1 =$	10
$b_2 =$	*****11111*****	$s_2 =$	10
$b_3 =$	*****11111*****	$s_3 =$	10
$b_4 =$	*****11111*****	$s_4 =$	10
$b_5 =$	00000*****	$s_5 =$	10
$b_6 =$	*****00000*****	$s_6 =$	10
$b_7 =$	*****00000*****	$s_7 =$	10
$b_8 =$	*****00000*****	$s_8 =$	10

Table 1 illustrates the building blocks of antigens 111 . . . 1 and 000 . . . 0 (string length is of 20 bits⁵). An antibody is said to match an antigen if its bit string is complementary to the antigen at certain building blocks. Specifically, the match score function M_b is to identify the building blocks for which an antibody matches an antigen, and then assign corresponding scores to that antibody. For example, given an antigen 111 . . . 1, an antibody with the first five and the last five bits being all 0’s will receive score $s_1 + s_4 = 20$, since these ten bits are complementary to those of the antigen.

Smith et al. [11] considered two cases for the score calculation of antibodies—perfect match and partial match. In case of perfect match, an antibody receives a non-zero score only if it perfectly matches the antigen. In case of partial match, an antibody receives a non-zero score if it partially matches the antigen. In terms of the distance d_{ij} between antibody i and antigen j , partial match indicates the degree by which an antibody matches an antigen—i.e., the number of bits of an antibody that are complementary to the corresponding bits of an antigen. The degree of match determines the specificity of an antibody. For example, if $d_{ij} = 0$, the matching is completely specific (that is, the antibody must perfectly match the antigen), but if $d_{ij} \neq 0$, it is partially matched. The consequence of a partial matching rule is that there is a trade-off between the number of antibodies used and their specificity—as the specificity of antibodies increases, so does the number of antibodies required to achieve a certain level of detection [5].

For the scoring rule discussed in the building-block-based recognition problem, we can also expand its definition by allowing partial match. In other words,

⁵ The small string length here serves well for illustrating the effect of the mate selection schemes. We current have some results for larger string lengths that are consistent with the results obtained for the small string length.

Table 2. Illustration of the immune-based GAs.

1. Randomly generate an initial population of n antibodies.
2. Evaluate antibodies' fitnesses by the six steps of the diversity algorithm.
3. Repeat until n offspring have been created.
 - a. select a pair of parents for mating by particular selection schemes;
 - b. apply crossover operator;
 - c. apply mutation operator.
4. Reset all the new individuals' fitnesses to zero and replace the current population with the new population.
5. Go to Step 2 until terminating condition.

if an antibody matches an antigen at all the bits of a building block, it is a perfect building-block match; if not all the bits of that building block are required for matching, it constitutes a partial building-block match. Therefore, the perfect building-block match case is that an antibody scores if all of its bits at a building block are complementary to those of an antigen. On the other hand, a case for partial match could allow an antibody to score with only 80% bits (i.e., 4 bits in case of the building blocks shown in Table 1) of a building block at which it matches an antigen. The result of this flexible scoring is a smaller population size required to achieve a certain level of recognition performance. In this paper, we mostly concentrate on this latter case for calculating antibody scores. (In case of 100% building-block match, a few experiments conducted so far have shown similar qualitative results as the 80% building-block match case, but it requires much larger population sizes, i.e., much higher computational costs, to achieve similar levels of performance.)

3.1 Effects of Mate Selection on Maintaining Subpopulations

To address the questions mentioned in the beginning of this section we conduct a series of GA experiments using the diversity algorithm. The illustration of the immune-based GAs is shown in Table 2.⁶ Our first objective is to investigate effects of mate selection on the diversity algorithm's subpopulation-maintaining

⁶ Since in the diversity algorithm the match scores of winning antibodies are continuously accumulated, after each generation their fitness values can be large. Thus at step 4 of Table 2 we reset the fitnesses of the new population's individuals to zero after each generation to prevent fitnesses from unlimited increase.

ability. Unless stated otherwise, these experiments use an antibody population size of 100, crossover rate of 0.7, mutation rate of 0.005, and ran for 150 generations. The antigen population is 50% 000...0 and 50% 111...1, and both antigens and antibodies are binary strings of length 20. The number of samples, σ , is 10, which is 10% of the population size. We choose this value because Smith et al.'s analysis suggests that too small or too large a sample size cannot show fitness sharing's effect. In addition, as mentioned in the preceding section, the number of cycles (C) does not have a bearing on the antibodies' expected fitnesses, 100 cycles (i.e., population size) used for each generation turned out to serve well for displaying subpopulation-maintaining results. Thus the total function evaluations for each run are generations $\times$ cycles $\times$ sample size, which equal 150,000.

Fig. 1 illustrates the experimental results of the diversity algorithm (averaged over 50 runs), evolved by the GAs with TS, TDM, TSM, RDM and RSM.

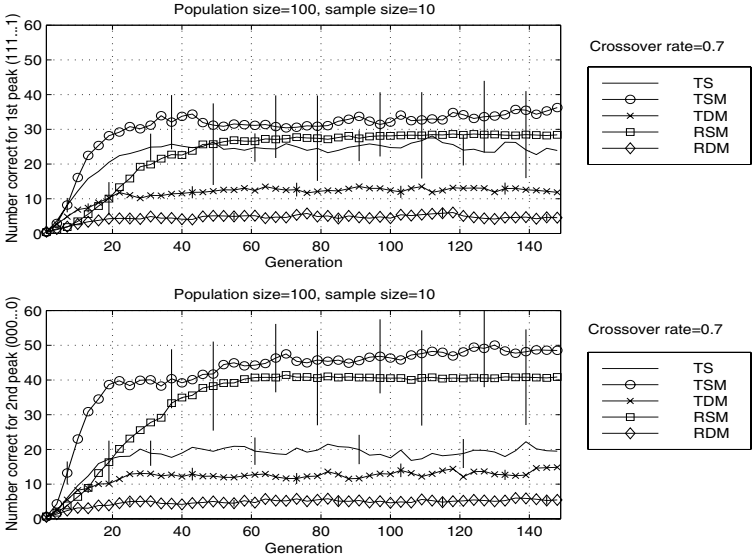


Fig. 1. The number of antibodies that correctly recognize antigens

These are the results for the numbers of antibodies that recognize antigens when all four building blocks are 80% correctly matched. Note that only the curves with small error bars (95% confidence intervals⁷) can be used for reliable judgements (we will discuss the reason for the larger error bars shortly), and thus the results for TS, TDM and RDM can be compared. It is clear that the

⁷ The vertical bars overlaying the metric curves throughout this paper represent the 95-percent confidence intervals calculated from Student's *t*-statistic [8].

dissimilar mating schemes, TDM and RDM, generate less desired antibodies than the regular tournament selection. The reason is in the following:

When crossover is turned on (crossover rate is .7, in this case), the dissimilarity-based mate selection increases the probability of producing useless hybrids—e.g., given an individual 111...1, and two candidate mates 111...1 and 000...0, the GAs with the dissimilar mating schemes tend to choose 000...0 for mating with 111...1, and the crossing-over between these two strings generates offspring that fall into the valley between the two peaks. Therefore, TDM and RDM maintain a smaller fraction of desired antibodies.

On the other hand, we see that TDM generates a larger fraction of desired antibodies than RDM. The difference between these two schemes is the method of selecting the second individual for mating—that is, in TDM fitter individuals have higher probabilities of being selected as mates, but this is not the case for RDM. As a result, TDM can pick out more individuals from the two peaks than RDM, which in turn increases the proportion of desired antibodies.

A remedy for the problem of producing useless hybrids would be to reduce dissimilar mating rates. In terms of the example above, the regular tournament selection confers 111...1 and 000...0 with equal probability of being selected for mating, thereby reducing the likelihood of two mating individuals chosen from the two peaks. However, if individuals tend to select similar mates, the selection pressure toward these individuals may be strong enough that the GA's population converges on only one peak. If this is the case, the diversity algorithm's capability for maintaining subpopulation is degraded.

The larger error bars for TSM and RSM in Fig. 1 illustrate this situation. Since TSM and RSM induce too strong a selection pressure, most of the GA's population members converge to only one peak. At generation 150, the GA with TSM has 20 (out of 50) runs in which most of the individuals converge to all 1's, and in 14 (out of 50) runs most of the individuals converge to all 0's, and there are 16 runs in which the two peaks are present, simultaneously. In case of RSM, there are 17 runs in which most of the individuals converge to all 1's, 21 runs in which most of the individuals converge to all 0's, and 12 runs where the two peaks are lost.

As a further illustration, Fig. 2 is the experimental results of a typical run for the number of desired antibodies obtained based on TSM. This figure shows that 000...0 are drown out by 111...1 after generation 60, although they do show up in earlier generations. This is because in TSM, similar individuals are always chosen as mates (with probability one)—a selection pressure toward similar mates enhances the convergence on one peak.

3.2 Effects of Mate Selection on the Discovery of Peaks

In the immune system problem considered, thus far we have been concerned with maintaining desired antibody subpopulations. However, there is another relevant issue we have not yet studied: the formation of the antibody subpopulations requires these antibodies to be discovered first. This is equivalent to the problem

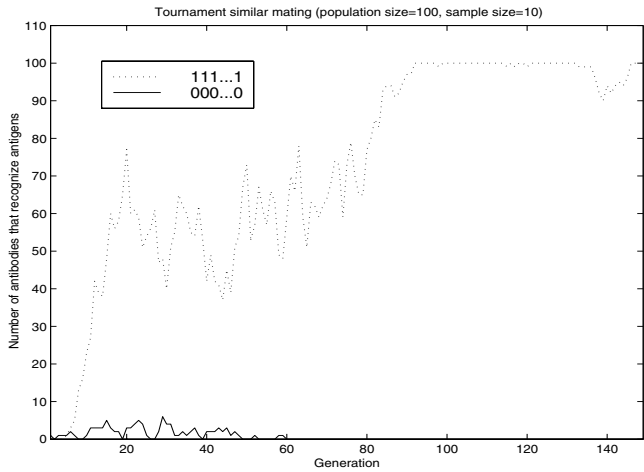


Fig. 2. The number of antibodies that correctly recognize antigens (based on the tournament similar mating), where all portion of the solid line (i.e., corresponding to 000...0) after generation 10 is on the 0 level

of finding multiple peaks. Since it has been shown, in [7], that the dissimilarity-based mate selection facilitates locating a single, best-so-far solution, we are interested in investigating if dissimilar mating is also more beneficial in finding multiple peaks than traditional selection schemes.

Table 3 displays the averaged mean function evaluations (over 50 runs) of discovering 111...1 and 000...0. These results show no obvious difference between various mate selection schemes for finding the two peaks, except that there are two runs where 000...0 was not found by the RSM GA, and this GA used a bit more evaluations to locate 111...1 than the other GAs. A closer inspection again shows the selection pressure toward similar individuals led the two particular runs of the GA to converge on 111...1, thereby precluding the discovery of the other peak. However, as population size decreases, the discrepancies between these mating schemes become more obvious. Table 4 illustrates the results for the number of runs (out of 50) in which antibodies 111...1 and 000...0 are discovered, respectively, based on population size 20 and sample size 2 (other parameter values remain unchanged). It is clear that the dissimilarity-based mating preferences facilitate locating two peaks. This is again because the similar mating schemes introduce a selection pressure strong enough that the corresponding GAs show inferior performance. All this confirms with our expectation that the dissimilarity-based mate selection is beneficial in locating multiple peaks.

Table 3. The mean function evaluations of discovering antibodies 111...1 and 000...0 (over 50 runs)

Antibody	TS	TDM	RDM	TSM	RSM
111...1	2340 (368)	2460 (333)	2460 (272)	2440 (204)	3060 (338)
000...0	2300 (206)	2540 (323)	2320 (270)	2180 (224)	48 runs reached

Table 4. The number of runs (out of 50) in which antibodies 111...1 and 000...0 are discovered

Antibody	TS	TDM	RDM	TSM	RSM
111...1	20	34	40	18	23
000...0	28	37	34	23	23

4 Conclusions and Future Work

In this paper, we have described Smith et al.'s immune system model in which subpopulations can be maintained through specific interactions among the strings. We have emphasized the performance of the GA in the binary immune system model, investigating how mate selection affects the GA's subpopulation-maintaining ability and the effects of mate selection on the discovery of multiple peaks. Both of these issues are important in the setting of multimodal function optimization, engineering and machine learning.

In studying the subpopulation-maintaining problem, the results illustrate that the dissimilar mating schemes are harmful in the sense of producing more lethal offspring. Consequently, the proportion of individuals that are representatives of different antibodies is reduced. We then showed that reducing the probability of dissimilar matings can remedy this problem. We also hoped to improve the GAs' performance by further increasing similar mating rates. However, as shown by the results obtained for TSM and RSM, they introduce a selection pressure strong enough that the population converges on only one peak.

In studying the peaks-identifying problem, we showed that the dissimilarity-based mate selection schemes facilitate locating multiple peaks of the fitness landscape. This is a crucial extension of the results obtained in [7], where dissimilar mating is shown to be more advantageous in finding a single, best-so-far solution.

Since the pattern-recognition strategy in our approach was based on schema detection, it is worth further exploration because in real problems when there are many more antigens than antibodies, antibodies need to detect common regions. In future work, we also hope to extend the results of schema detection and multiple-peaks identification to more realistic scale of antigens and antibodies. Finally, we would like to develop an analytical analysis to enhance our understanding for mate selection in the context of the immune-GA-based system.

Acknowledgments. The author would like to thank John Holland, Rick Riolo for their advice, and Bob Lindsay, Ted Belding, Leeann Fu, Tom Bersano-Begey and Bill Rand for their comments and suggestions.

References

1. Farmer, J. D., Packard, N. H., and Perelson, A. S.: The Immune System, Adaptation, and Machine Learning. In D. Farmer, A. Lapedes, N. Packard, and B. Wendroff (Eds.): *Evolution, Games and Learning*. North-Holland (1986). (Reprinted from *Physica*, 22D, 187–204)
2. Forrest, S., Javornik, B., Smith, R. E., and Perelson, A. S.: Using Genetic Algorithms to Explore Pattern Recognition in the Immune System. *Evolutionary Computation*, 1(3) (1993) 191–211.
3. Goldberg, D. E. and Richardson, J.: Genetic Algorithms with Sharing for Multimodal Function Optimization. *Genetic Algorithms and Their Applications: Proceedings of the Second International Conference on Genetic Algorithms* (1987) 41–49.
4. Goldberg, D. E. and Segrest, D.: Finite Markov Chain Analysis of Genetic Algorithms. *International Conference on Genetic Algorithms*, 2 (1987) 1–8.
5. Hofmeyr, S. A., and Forrest, S.: Architecture for an artificial immune system. *Evolutionary Computation*, 8(4) (2000) 443–473.
6. Holland, J. H.: *Adaptation in Natural and Artificial Systems*. Ann Arbor, MI: University of Michigan Press (1975).
7. Huang, C.-F.: *A Study of Mate Selection in Genetic Algorithms*. Doctoral dissertation. Ann Arbor, MI: University of Michigan, Electrical Engineering and Computer Science (2002).
8. Miller, R. G.: *Beyond ANOVA, Basics of Applied Statistics*. John Wiley and Sons (1986).
9. Mitchell, M.: *An Introduction to Genetic Algorithms*. Cambridge, MA: MIT Press (1996).
10. Perelson, A. S.: Immune network theory. *Immunol. Rev.*, 110 (1989) 5–36.
11. Smith, R., Forrest, S., and Perelson, A. S.: Searching for Diverse, Cooperative Populations with Genetic Algorithms. *Evolutionary Computation*, 1(2) (1993) 127–149.
12. Vlachos, C., Williams, D., and Gomm, J. B.: Genetic Approach to Decentralized PI Controller Tuning for Multivariable Processes. *IEE Proc. Control Theory and Applications*, 146 (1999), 58–64.

Designing a Hybrid Genetic Algorithm for the Linear Ordering Problem

Gaofeng Huang¹ and Andrew Lim²

¹ Department of Computer Science, National University of Singapore
3 Science Drive 2, Singapore 117543

² Department of Industrial Engineering and Engineering Management
Hong Kong Univ of Sci and Tech, Clear Water Bay, Kowloon, Hong Kong

Abstract. The Linear Ordering Problem(LOP), which is a well-known $\mathcal{NP}$ -hard problem, has numerous applications in various fields. Using this problem as an example, we illustrate a general procedure of designing a hybrid genetic algorithm, which includes the selection of crossover/mutation operators, accelerating the local search module and tuning the parameters. Experimental results show that our hybrid genetic algorithm outperforms all other existing exact and heuristic algorithms for this problem.

Keywords: Linear Ordering Problem, Genetic Algorithm, Hybridization.

1 Introduction

The Linear Ordering Problem(LOP) has numerous applications in economics, archaeology, scheduling, the social sciences, and aggregation of individual preferences[5,6,7,8]. Of all these applications, the most famous one may be “the Triangulation for Input-Output Matrices”[7], which measures the movement of goods from one “sector” to another in economics research.

Mathematically, LOP can be formulated as :

Instance : a matrix $C = \{c_{ij}\}_{n \times n}$
Solution : $p = (p_1, p_2, \dots, p_n)$, a permutation of $1 \dots n$
Objective : to maximize $C(p) = \sum_{i=1}^n \sum_{j=i+1}^n c_{p_i, p_j}$

This problem is known to be $\mathcal{NP}$ -hard[9]. Many exact and heuristic algorithms have been proposed to solve it.

Several exact methods have been devised based on the integer programming technique. Grotschel *et al.* first proposed a cutting plane algorithm in [6,7]; Mitchell and Borchers[8] improved the result by combining the cutting plane with the interior point algorithm. However, all of these exact algorithms are extremely time-consuming.

On the other hand, heuristic algorithms seem to be more practical in solving large instances of this problem[1,2,3,4,5]. The heuristic proposed by Chanas and Kobylanski[5] can usually provide acceptable solutions both in terms of time and quality. Laguna *et al.*[1] applied Tabu Search technique with path relinking strategy on this problem successfully, while Campos *et al.*[2] used Scatter Search to solve this problem. These two approaches are often regarded as the best at this moment.

The purpose of this paper includes: (1) to develop an effective heuristic for LOP and (2) to illustrate a general procedure of designing a hybrid genetic algorithm. Genetic algorithms(GA) have shown to be competitive technique for solving general combinatorial optimization problems. However, it is possible to incorporate problem-specific knowledge into GA so that the results can be further improved. The hybridization between GA and Local Search is such a method. Experimental results show that our hybrid genetic algorithm outperforms all other existing exact and heuristic algorithms for LOP.

The rest of this paper is organized as follows. We first present the details of our hybrid GA implementation in Section 2. In Section 3, preliminary experiments are conducted to tune our algorithm. The final computational results and comparisons are reported in Section 4. In the last section, we present our conclusions.

2 Hybrid Genetic Algorithm

Many variations of hybridization between Genetic Algorithms and Local Search have been proposed[11,13,14]. In this paper, our hybrid GA has the following structure:

Algorithm 1 Hybrid Genetic Algorithm

```

1: Initialize population_size, crossover_rate, mutation_rate
2: Generate Initial Generation gen
3: while not TerminateConditions() do
4:   for pop  $\leftarrow$  1 to population_size do
5:     Randomly selected two individuals from gen according to the fitness, say, genx and geny
6:     if random[0, 1) < crossover_rate then
7:       next_genpop  $\leftarrow$  CrossoverOperator(genx, geny)
8:     else
9:       next_genpop  $\leftarrow$  Copy(better(genx, geny))
10:    end if
11:    if random[0, 1) < mutation_rate then
12:      next_genpop  $\leftarrow$  MutationOperator(next_genpop)
13:    end if
14:    next_genpop  $\leftarrow$  LocalSearch(next_genpop)
15:    if fitness(next_genpop) > history_best then
16:      history_best  $\leftarrow$  fitness(next_genpop)
17:      solution  $\leftarrow$  next_genpop
18:    end if
19:  end for
20:  gen  $\leftarrow$  next_gen
21: end while

```

In this structure, we include all the basic components of a typical genetic algorithm, such as Initial Generation, Terminate Conditions, CrossoverOperator, MutationOperator, Fitness Function etc. In addition, we have also incorporated a local search module to improve the quality of each individual in the population. More details are provided in the following subsections.

2.1 Genetic Algorithm

- **Individual Representation and Fitness Function :** In LOP, any permutation $p = (p_1 p_2 \dots p_n)$ is a feasible solution. It is very natural to use a n length vector p as the representation (chromosome) of an individual. The objective function $C(p)$ acts as the fitness function of chromosome.
- **Initial Generation :** For the initial generation, each individual is set using a permutation that is randomly generated, after which a local search is applied to it to improve its quality. Therefore, every individual in the initial generation is already a local maximum in the solution space.
- **Terminate Conditions :** We do not have duplicate detection scheme in our genetic algorithm. The evolution process is to converge to a “best” solution. When this happens, the algorithm terminates. However, checking for convergence can be time consuming. An alternative is to use the fitness function to approximate convergence.
- **Crossover Operators :** Three crossover operators, including PMX, CX and OX, are implemented in our algorithm and tested by our experiments.
 - Partially Mapped Crossover(PMX) operator was proposed by Goldberg and Lingle[10]. The basic idea of PMX is to exchange a partial segment between two parents. However, a skill named mapping is used in order to keep the results still as feasible permutations.
 - Cycle Crossover(CX) operator was first used in [12]. It first finds all the mapping cycles between two parents. After that, for each cycle, it randomly selects one of the two parents, and copies its elements to the offspring in corresponding position.
 - Order Crossover(OX) operator[13] first randomly selects several same elements in both parents, and then exchanges are made between two parents in those positions in order.
- **Mutation Operators :** We experiment with the two most commonly used mutation operators, DM and k-EM.
 - Displacement Mutation(DM) operator[14] randomly select a segment from the chromosome sequence and insert it into another randomly selected position.
 - k-EM is a variation of Exchange Mutation(EM) operator, where k exchange(swap) operations are performed synchronously in $2 \times k$ random selected positions.

2.2 Local Search

The local search strategy we used in LOP is based on the idea of iterative improvement. It starts with an initial solution(permutation p), and tries to improve the solution via a series of neighbour moves until no improvement can be made.

Neighbouring Move : In each iteration, the solution p is improved by a “neighbouring move”. We use the INSERT move (also named DELETE-INSERT or SHIFT) and the EXCHANGE move (also named SWAP or Pairwise-Interchange) that are commonly used in permutation problems. For more details please see Table 1.

Table 1. Two commonly used neighbouring moves

neighbouring move	explanation	examples: $p = (5, 3, 1, 2, 4)$
INSERT(p, i, j)	insert p_i to position j	INSERT($p, 4, 1$) = $(2, 5, 3, 1, 4)$
EXCHANGE(p, i, j)	exchange p_i and p_j	EXCHANGE($p, 4, 1$) = $(2, 3, 1, 5, 4)$

The following is an interesting observation.

Theorem 1. *For the LOP, INSERT move subsumes EXCHANGE move.*

Proof. – **If a solution p can be improved by the INSERT move, it may not be improved by the EXCHANGE move.**

An easy counterexample can be constructed when $n = 3$:

consider $p = (1, 3, 2)$, $c_{2,1} - c_{1,2} = 10$, $c_{3,1} - c_{1,3} = -1$, $c_{2,3} - c_{3,2} = -100$,

An INSERT move can be made to improve the result since

$$\text{INSERT}(p, 1, 3) - p = C(3, 2, 1) - C(1, 3, 2) = (c_{3,1} - c_{1,3}) + (c_{2,1} - c_{1,2}) = -1 + 10 = 9 > 0.$$

Consider the EXCHANGE move,

$$\text{EXCHANGE}(p, 1, 2) - p = C(3, 1, 2) - C(1, 3, 2) = -1 < 0,$$

$$\text{EXCHANGE}(p, 1, 3) - p = C(2, 3, 1) - C(1, 3, 2) = -100 + 10 - 1 < 0,$$

$$\text{EXCHANGE}(p, 2, 3) - p = C(1, 2, 3) - C(1, 3, 2) = -100 < 0.$$

Hence, no EXCHANGE move can be made to improve the solution p .

– **If a solution p can be improved by the EXCHANGE move, it can always be improved by the INSERT moves.**

Suppose p can be improved by EXCHANGE(p, i, j)($i < j$):

$$\text{Let } p'' = \text{INSERT}(p, i, j - 1)$$

$$p''' = \text{INSERT}(p, j, i)$$

$$\text{Obviously, } p' = \text{EXCHANGE}(p, i, j) = \text{INSERT}(p'', j, i) = \text{INSERT}(p''', i + 1, j)$$

$$\begin{aligned} \text{So, } C(p') - C(p) &= (C(p') - C(p'')) + (C(p'') - C(p)) \\ &= (C(p') - C(p''')) + (C(p''') - C(p)) \end{aligned}$$

A important property of LOP is that

$$C(p') - C(p''') = C(p'') - C(p) = \sum_{k=i+1}^{j-1} (c_{p_k, p_i} - c_{p_i, p_k})$$

Therefore, $C(p') - C(p) = (C(p'') - C(p)) + (C(p''') - C(p))$.

Since solution p' is better than p , $C(p') - C(p) > 0$, we have $C(p'') - C(p) > 0$ or $C(p''') - C(p) > 0$, that is, p can also be improved by INSERT. $\square$

As a result, only INSERT moves are considered in our algorithm. The gain in objective function after one INSERT move is given as:

$$dC_{i,j} = C(\text{INSERT}(p, i, j)) - C(p) = \begin{cases} \sum_{k=i+1}^j (c_{p_k, p_i} - c_{p_i, p_k}) & \text{for } i < j \\ 0 & \text{for } i = j \\ \sum_{k=j}^{i-1} (c_{p_i, p_k} - c_{p_k, p_i}) & \text{for } j < i \end{cases} \quad (1)$$

Search Strategy : In each iteration, there are $n(n-1)$ choices for an INSERT move. The FirstFit and BestFit are two widely adopted strategies.

As shown in Algorithm 2, the FirstFit search strategy always uses the first found move which can lead to a better solution.

Algorithm 2 FirstFit

```

1:  $p \leftarrow$  Initialize Solution(permutation)
2:  $finishFlag \leftarrow false$ 
3: while not  $finishFlag$  do
4:    $finishFlag \leftarrow true$ 
5:   for  $i \leftarrow 1$  to  $n$  do
6:     for  $j \leftarrow 1$  to  $n$  do
7:       compute  $dC_{i,j}$  using Equation(1)
8:       if  $dC_{i,j} > 0$  then
9:          $p \leftarrow \text{INSERT}(p, i, j)$  {update the solution}
10:       $finishFlag \leftarrow false$ 
11:    end if
12:  end for
13: end for
14: end while

```

In this algorithm, the computation of $dC_{i,j}$ using Equation(1) is very time-consuming, since each entry takes $O(n)$ time. If the “WHILE-LOOP” runs T_1 times, the whole algorithm will need $T_1 n^2 O(n) = T_1 O(n^3)$ time.

$$dC_{i,j} = \begin{cases} dC_{i,j-1} + (c_{p_j, p_i} - c_{p_i, p_j}) & \text{for } i < j \\ 0 & \text{for } i = j \\ dC_{i,j+1} + (c_{p_i, p_j} - c_{p_j, p_i}) & \text{for } j < i \end{cases} \quad (2)$$

Unlike FirstFit, BestFit search strategy uses Equation(2) to compute all n^2 $dC_{i,j}$ entries and finds the best solution among all n^2 possible moves. Since each entry can be computed in $O(1)$ time, if the whole algorithm terminates after t_2 iterations, it will take $t_2 O(n^2)$ time.

It’s expected that $t_2 < T_1 n^2$ because BestFit tends to take the “faster” ascent direction to a local maximum. On the other hand, usually $t_2 \gg T_1$ since in each “round”, FirstFit has up to n^2 INSERT moves, while BestFit makes 1 best INSERT move. This may explain the results in [1] that BestFit even takes more real CPU time than FirstFit.

We propose a FastFit search strategy that takes advantage of the strengths of both approaches. By using the idea of the “cache”, we set a “dirty” flag in our algorithm. As long as no actual INSERT move is made, the $dC_{i,j}$ cache

Algorithm 3 FastFit

```

1:  $p \leftarrow$  Initialize Solution(permutation)
2:  $finishFlag \leftarrow false$ 
3: while not finishFlag do
4:    $finishFlag \leftarrow true$ 
5:   for  $i \leftarrow 1$  to  $n$  do
6:      $dirtyFlag \leftarrow false$ 
7:      $dC_{i,i} \leftarrow 0$ 
8:     for  $j \leftarrow i + 1, i + 2, \dots, n; i - 1, i - 2, \dots, 1$  do
9:       if dirtyFlag then
10:        compute  $dC_{i,j}$  using Equation(1)
11:         $dirtyFlag \leftarrow false$ 
12:       else
13:        compute  $dC_{i,j}$  using Equation(2)
14:       end if
15:       if  $dC_{i,j} > 0$  then
16:         $p \leftarrow \text{INSERT}(p, i, j)$  {update the solution}
17:         $finishFlag \leftarrow false$ 
18:         $dirtyFlag \leftarrow true$ 
19:       end if
20:     end for
21:   end for
22: end while

```

is always “clean”, so we can compute the next dC entry in $O(1)$ time. Only after one actual INSERT move, the dC cache becomes “dirty” so that $O(n)$ time is needed to compute $dC_{i,j}$. If the “WHILE-LOOP” runs T_3 times, the whole FastFit algorithm would need $T_3O(n^2) + t_3O(n)$, where t_3 is the number of INSERT moves made to reach a local maximum.

It is expected that $T_3 \approx T_1$, as FastFit and FirstFit are very similar. We have $t_2 \gg T_1$, so $T_3 \ll t_2$. Therefore, FastFit is expected to be much faster than BestFit and FirstFit.

3 Preliminary Experiments

A typical Genetic Algorithm may have 100 individuals, while `crossover_rate` and `mutation_rate` are set to be 0.8 and 0.05 respectively. However, performance of a Genetic Algorithm may be very sensitive to the settings of these parameters. And there also may be some interaction effects between crossover type, mutation type, crossover rate and mutation rate. Since it’s too computationally exhaustive to experiment all such combinations, therefore, in this section, preliminary experiments are conducted to tune our GA with FastFit local search strategy.

25 instances from [2] are used for our preliminary experiments. These instances are the same as those instances with $n = 75$ in “Random Instances Set B” (see Section 4.3).

3.1 Combination of Crossover Operators and Mutation Operators

First, we experiment the combination of crossover and mutation operators with the configuration : `crossover_rate` = 0.8, `mutation_rate` = 0.05,

Table 2. Combination of Crossover operators and Mutation operators

	OX	CX	PMX	average	std dev
DM	32916465.40	32915105.08	32916825.64	32916132.04	907.43
1-EM	32916657.56	32916144.60	32916548.52	32916450.23	270.24
2-EM	32916467.28	32915031.28	32917855.44	32916451.33	1412.15
5-EM	32917524.20	32915519.84	32918195.64	32917079.89	1392.13
10-EM	32917299.60	32914183.04	32917273.64	32916252.09	1791.90
average	32916882.81	32915196.77	32917339.78		
std dev	495.66	718.71	687.97		

population_size = 100. Table 2 shows the average results of all 25 instances for each combination.

From Table 2, the performance of different operators do not vary much. Among these, the combination of crossover operator PMX and mutation operator 5-EM provides the best average result. However, PMX operator seems to be slightly better than the other two crossover operators.

3.2 Tuning of Parameter mutation_rate

Using the combination of PMX and 5-EM operator(denoted as GA_PMX_5-EM), now we consider the mutation_rate parameter with crossover_rate = 0.80 and population_size = 100.

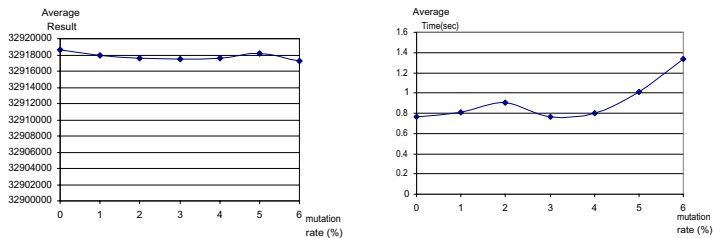


Fig. 1. Tuning of Parameter mutation_rate

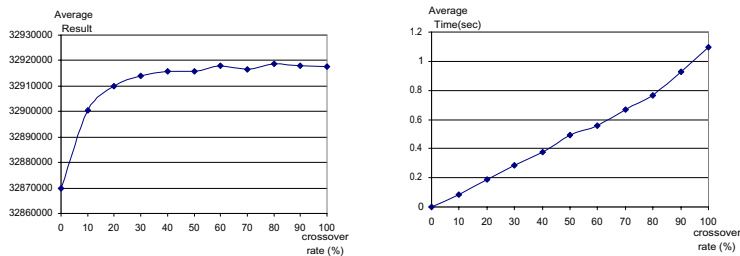


Fig. 2. Tuning of Parameter crossover_rate

Figure 1 shows the average results and CPU time taken for different mutation_rate. It's reasonable that CPU time increases with the growth of

`mutation_rate`, because more mutation means more computation. But as you can see in Fig. 1, the average result is nearly the same, which means the mutation in our GA does not improve the solution's quality significantly, hence we set `mutation_rate` = 0.

3.3 Tuning of Parameter `crossover_rate`

Parameter `crossover_rate` determines the probability of crossover happens in GA. Fig. 2 shows the average results and consumed CPU time under configuration : GA_PMX, `mutation_rate` = 0, `population_size` = 100.

As shown in Fig. 2, the average result increases with the growth of `crossover_rate`, especially when it is less than 0.5. On the other hand, the CPU time increases with `crossover_rate` in an approximately linear manner. With the compromise of time and solution quality, we choose `crossover_rate` = 0.5.

3.4 Tuning of Parameter `population_size`

`population_size` may be the most influential parameter in a Genetic Algorithm. If `population_size` is too small, there will be insufficient genetic diversity to search the feasible solution space thoroughly. On the other hand, the algorithm may be extremely slow when `population_size` is too large.

Fig. 3 shows the preliminary results with different `population_size` under the configuration : GA_PMX, no mutation, `crossover_rate` = 0.50. With the growth of `population_size`, the average result increase consistently although the increase becomes slower, while the CPU time consumed is approximately linear to `population_size`. Therefore, it's reasonable to choose `population_size` = 40, as it provide a good balance between solution quality and computing time.

3.5 Summary: Effect of Genetic Algorithm and Local Search

In our preliminary experiments, we examine the effects of both the Genetic Algorithm and Local Search. The following four algorithms are compared:

- SimpleGA is the simple Genetic Algorithm without local search.
- LocalSearch(random) is a multi-round local search algorithm. In each round, it starts with a random initial permutation and uses FastFit strategy to improve the solution.
- hGA(untuned) is a hybrid Genetic Algorithm with the configuration : GA with OX crossover operator, DM mutation operator, BestFit local search strategy, `crossover_rate` = 0.8, `mutation_rate` = 0.05, `population_size` = 100. This stands for an untuned hybrid Genetic Algorithm.
- hGA(tuned) is our final hybrid Genetic Algorithm with the configuration : GA with PMX crossover operator, no mutation operator, FirstFit local search strategy, `crossover_rate` = 0.5, `population_size` = 40.

Fig. 4 shows the time-quality curve for these 4 algorithms. The vertical axis shows the CPU time consumed, while the horizontal axis shows the average result for the 25 instances that we experimented with.

The following remarks can be made from Fig. 4:

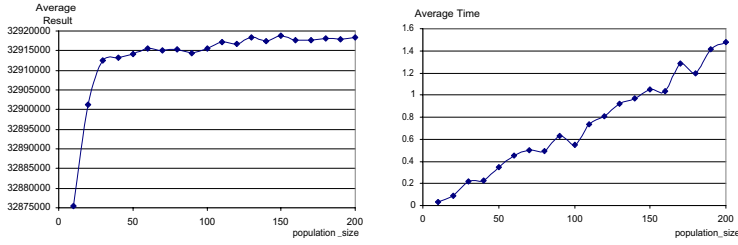


Fig. 3. Tuning of Parameter `population_size`

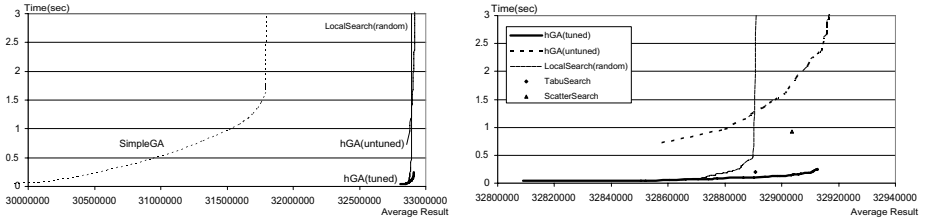


Fig. 4. Time-Quality Curve

- SimpleGA is rather bad in performance.
- LocalSearch(random) has the ability to find relatively good solutions in a short time. Unfortunately, even if much more time is given, this algorithm cannot improve the best solution by much. For the 25 instances that we have tested, no better solutions can be found after running 0.5 second.
- hGA(untuned) shows the power of hybridization of Genetic Algorithm and Local Search. Given enough time, it is capable of finding competitive solutions. However, it's slower than LocalSearch(random) since even the initialization alone will take more than 0.5 second.
- hGA(tuned) is about 10 times faster than hGA(untuned), which shows the power of proper tuning a search method. Consequently, hGA(tuned) outperforms Tabu Search and Scatter Search both in time and solution quality.

Guided by the Genetic Algorithm, it is possible for Local Search to improve the quality of solution consistently; and with the help of Local Search, Genetic Algorithm becomes competitive in finding good solutions.

4 Computational Results

Three widely-used sets of instances are tested to demonstrate the effectiveness of our hybrid Genetic Algorithm. All the codes are implemented in C/C++ and run on a PentiumIII 800Mhz PC. And for each instance, our hGA is run only once.

However, the computing machines used in previous papers are different, i.e., Intel Pentium 166Mhz PC, Intel PentiumIII 500Mhz, Sun SPARC 20 Model 71. Therefore, in order to compare the CPU time, a scaling scheme is used according to SPEC(<http://www.specbench.org/osg/cpu2000/>)¹

4.1 LOLIB Instances

LOLIB may be the most widely-used testing instances for LOP. It contains 49 instances of real-world input-output tables from sectors in the European and United States Economies. The data together with optimal solutions can be obtained from www.iwr.uni-heidelberg.de/groups/comopt/software/LOLIB/

Nearly all papers on LOP use LOLIB as test instances: in [1], the results of Chanas-Kobylanski algorithm(CK) and Tabu Search with path relinking(TS-LOP) are reported. [2] shows computational results of two Scatter Search versions(SS, SS10). A Variable Neighborhood Search(VNS-LOP) algorithm is applied in [3], while [4] gives the results of a Lagrangian Based Heuristic(LH-VP).

Table 3. LOLIB (49 instances)

	Avg.Obj. Value	Deviation from Opt.	No. of Optimal	Avg.CPU Times(sec)		
				in P166	PIII500	PIII800
CK	22018008.35	0.15%	11	0.1	5.581	
SS	22041229.8	0.01%	42	3.82		
SS10	22041232.3	0.01%	43	14.28		
LH-VP		0.00%	43			
VNS-LOP	22041260.8	0.00%	44	0.87	< 0.05	0.0247
TS-LOP	22041261.51	0.00%	47	0.93		
hGA	22041263.82	—	49	< 0.2		

Table 3 shows for each algorithm the average objective function value over all 49 instances, the average percent deviation from the optimal solution, the number of optimal solutions a particular algorithm can reach, and the average CPU time. It's evident that our hGA outperforms all other algorithms, since not only is it faster, but also it's the only one algorithm which can provide all optimal solutions for 49 instances.

4.2 Random Instances Set A

This set of instances is randomly generated by J. E. Mitchell and B. Borchers[8]. There are 30 instances, where the size of instances vary from 100 to 250. More specifically, there are 5 instances with size $n = 100$ and $n = 250$ respectively, and 10 instances with size $n = 150$ and $n = 200$. Both the instances and optimal values are available at the website: www.rpi.edu/~mitchj/generators/linord/

The results of Interior Point algorithm, Simplex Cutting Plane algorithm and the combined of these two are reported in [8], while in [4] two Lagrangian Based Heuristics(LH-PC and LH-VP) are applied to this set of instances.

¹ SPEC(Standard Performance Evaluation Corporation) points out that PIII 800 is not more 2 times faster than PIII500, not more 8 times faster than P166 and not more 12 times faster than SPARC 20/71

Table 4. Random Instances Set A (30 instances)

	Avg.Obj. Value	Deviation from Opt.	No. of Optimal	Avg.CPU Times(sec)		
				SPARC20	PIII500	PIII800
Interior Simplex Combined	—		26	2390.11		
	—		30	2697.33		
	—		30	356.16		
LH-PC		0.227%	16		605.13	
LH-VP		0.014%	28		105.46	
hGA	777324.17	—	30	< 24	< 4	1.9665

As you can see from Table 4, our hGA is 10 ~ 100 times faster than those exact algorithms and more than 25 times faster than Lagrangian Based Heuristics. What’s more important, with respect to all these 30 instances, the solutions obtained by our hGA are all optimal.

4.3 Random Instances Set B

This set includes 75 instances generated by Laguna *et al.*[1]², which consists of 25 instances for each problem size $n = 75, 150, 200$ with each entries of the cost matrix c_{ij} randomly distributed in (0, 25000). No optimal solutions are reported for these instances by now, while the Tabu Search with path relinking strategy (TS-LOP) gives the best result before our experiment.

Table 5. Random Instances Set B (75 instances)

	Avg.Obj. Value	Deviation from best	No. of best	Avg. Times(sec) in P166 PIII800	
CK	128663947.3	0.64%	0	10.67	
CK-10	128919838.0	0.39%	0	108.44	
TS-LOP	129269367.5	0.11%	5	20.19	
hGA	129437686.3	—	75	< 18	2.2402

Table 5 shows the results of our experiment. As you can see, our hGA algorithm gives better solutions than TS-LOP for 70 instances, for the other 5 instances the solutions between hGA and TS-LOP are same. It is clear that our hGA outperforms TS-LOP.

5 Conclusion

The Linear Order Problem is studied in this paper. We designed a hybrid Genetic Algorithm by integrating Genetic Algorithm and Local Search strategy successfully. Experiments indicate that parameters are influential for genetic algorithm. We described the procedure of tuning the parameters and develop a FastFit algorithm to accelerate the local search. As a result, the algorithm after tuning becomes 10 times faster than before.

Computational results show that our hybrid genetic algorithm outperforms all other existing exact and heuristic algorithms.

² The authors would like to express special thanks to Prof. Manuel Laguna for providing the instances and detail results of TS-LOP.

References

1. M. Laguna, R. Martí and V. Campos: *Intensification and diversification with elite tabu search solutions for the linear ordering problem*, Computer and Operation Research, vol.26, pp. 1217–1230, (1999)
2. V. Campos, M. Laguna and R. Martí: *Scatter Search for the Linear Ordering Problem*, New Ideas in Optimization, D. Corne, M. Dorigo and F. Glover (Eds.), McGraw-Hill, pp. 331–339 (1999)
3. Carlos García González and Dionisio Pérez-Brito: *A Variable Neighborhood Search for Solving the Linear Ordering Problem*, In the Proceedings of MIC'2001 - 4th Metaheuristics International Conference, pp. 181–185, Porto, Portugal, (2001)
4. Alexandre Belloni and Abilio Lucena: *Lagrangian Based Heuristics for the Linear Ordering Problem*, In the Proceedings of MIC'2001 - 4th Metaheuristics International Conference, pp. 445–449, Porto, Portugal, (2001)
5. S. Chanas and P. Kobylanski: *A New Heuristic Algorithm Solving the Linear Ordering Problem*, Computational Optimization and Applications, vol.6, pp. 191–205, (1996)
6. M. Grotschel, M. Junger and G. Reinelt: *A Cutting Plane Algorithm for the Linear Ordering Problem*, Operations Research, vol.32, no.6, pp. 1995–1220, (1984)
7. M. Grotschel, M. Juenger and G. Reinelt: *Optimal Triangulation of Large Real World Input-Output Matrices*, Statistische Hefte 25, 261–295, (1984)
8. J. E. Mitchell and B. Borchers: *Solving linear ordering problems with a combined interior point/simplex cutting plane algorithm*, High Performance Optimization, Kluwer Academic Publishers, Dordrecht, The Netherlands, H. Frenk et al. (Eds.), pp. 349–366, (2000)
9. R.M. Karp: *Reducibility among combinatorial problems*, In R.E. Miller and J.W. Thatcher (Eds.), Complexity of Computer Computations, pp. 85–103, New York, (1972)
10. D. Goldberg and R. Lingle: *Alleles, loci, and the traveling salesman problem*, in the Proceedings of the 1st International Conference on Genetic Algorithms and Their Applications, pp. 154–159, Lawrence Erlbaum, Hillsdale, New Jersey, (1985).
11. D. Goldberg: *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley, Reading, MA, (1989)
12. I. Oliver, D. Smith, and J. Holland: *A study of permutation crossover operators on the tsp*, in the Proceedings of the 2nd International Conference on Genetic Algorithms and Their Applications, pp. 224–230, Hillsdale, New Jersey, (1987).
13. G. Syswerda: *Schedule Optimization Using Genetic Algorithms*, in L.Davis (Ed.) Handbook of Genetic Algorithms, pp. 332–349, New York, (1991)
14. Z. Michalewicz: *Genetic Algorithms + Data Structure = Evolution Programs*, Springer-Verlag, Berlin Heidelberg, (1992)

A Similarity-Based Mating Scheme for Evolutionary Multiobjective Optimization

Hisao Ishibuchi and Youhei Shibata

Department of Industrial Engineering, Osaka Prefecture University,
1-1 Gakuen-cho, Sakai, Osaka 599-8531, Japan
{hisaoi, shibata}@ie.osakafu-u.ac.jp

Abstract. This paper proposes a new mating scheme for evolutionary multiobjective optimization (EMO), which simultaneously improves the convergence speed to the Pareto-front and the diversity of solutions. The proposed mating scheme is a two-stage selection mechanism. In the first stage, standard fitness-based selection is iterated for selecting a pre-specified number of candidate solutions from the current population. In the second stage, similarity-based tournament selection is used for choosing a pair of parents among the candidate solutions selected in the first stage. For maintaining the diversity of solutions, selection probabilities of parents are biased toward extreme solutions that are different from prototypical (i.e., average) solutions. At the same time, our mating scheme uses a mechanism where similar parents are more likely to be chosen for improving the convergence speed to the Pareto-front. Through computational experiments on multi-objective knapsack problems, it is shown that the performance of recently proposed well-known EMO algorithms (SPEA and NSGA-II) can be improved by our mating scheme.

1 Introduction

Evolutionary multi-objective optimization (EMO) algorithms have been applied to various problems for efficiently finding their Pareto-optimal or near Pareto-optimal solutions. Recent EMO algorithms usually share some common ideas such as elitism, fitness sharing and Pareto ranking for improving both the diversity of solutions and the convergence speed to the Pareto-front (e.g., see Coello et al. [1] and Deb [3]). In some studies, local search was combined with EMO algorithms for further improving the convergence speed to the Pareto-front [10, 12–14]. While mating restriction has been often discussed in the literature, its effect has not been clearly demonstrated. As a result, it is not used in many EMO algorithms as pointed out in some reviews on EMO algorithms [6, 17, 21]. The aim of this paper is to clearly demonstrate that the search ability of EMO algorithms can be improved by appropriately choosing parent solutions. For this aim, we propose a new mating scheme that is applicable to any EMO algorithms. For maintaining the diversity of solutions, the selection probabilities of parent solutions are biased toward extreme solutions that are different from prototypical (i.e., average) solutions in our mating scheme. At the same time,

our mating scheme uses a mechanism where similar parents are more likely to be chosen for improving the convergence speed to the Pareto-front.

Mating restriction was suggested by Goldberg [7] and used in EMO algorithms by Hajela & Lin [8] and Fonseca & Fleming [5]. The basic idea of mating restriction is to ban the crossover of dissimilar parents from which good offspring are not likely to be generated. In the implementation of mating restriction, a user-definable parameter σ_{mating} called the mating radius is usually used for banning the crossover of two parents whose distance is larger than σ_{mating} . The distance between two parents is measured in the decision space or the objective space. The necessity of mating restriction in EMO algorithms was also stressed by Jaszkiewicz [13] and Watanabe et al. [18]. On the other hand, Zitzler & Thiele [20] reported that no improvement was achieved by mating restriction in their computational experiments. Moreover, there was also an argument for the selection of dissimilar parents. Horn et al. [9] argued that information from very different types of tradeoffs could be combined to yield other kinds of good tradeoffs. Schaffer [16] examined the selection of dissimilar parents but observed no improvement.

In our previous study [11], we demonstrated positive and negative effects of mating restriction on the search ability of EMO algorithms through computational experiments on knapsack problems and flowshop scheduling problems. The positive effect of the recombination of similar parents is the improvement in the convergence speed to the Pareto-front while its negative effect is the decrease in the diversity of solutions. On the other hand, the positive effect of the recombination of dissimilar parents is the improvement in the diversity while its negative effect is the deterioration in the convergence speed. In this paper, we propose a new mating scheme for simultaneously improving the convergence speed and the diversity. The effect of the proposed mating scheme on the performance of the SPEA [21] and the NSGA-II [4] is examined through computational experiments on knapsack problems in Zitzler & Thiele [21]. Experimental results show that the search ability of those EMO algorithms on the two-objective and three-objective knapsack problems is significantly improved by the proposed mating scheme.

2 Proposed Mating Scheme

We describe our mating scheme using the following k -objective optimization problem:

$$\text{Optimize } \mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), \dots, f_k(\mathbf{x})), \quad (1)$$

$$\text{subject to } \mathbf{x} \in \mathbf{X}, \quad (2)$$

where $\mathbf{f}(\mathbf{x})$ is the objective vector, $f_i(\mathbf{x})$ is the i -th objective to be minimized or maximized, $\mathbf{x}$ is the decision vector, and $\mathbf{X}$ is the feasible region in the decision space. Let us denote the distance between two solutions $\mathbf{x}$ and $\mathbf{y}$ as $|\mathbf{f}(\mathbf{x}) - \mathbf{f}(\mathbf{y})|$ in the objective space. In this paper, the distance is measured by the Euclidean distance as

$$|\mathbf{f}(\mathbf{x}) - \mathbf{f}(\mathbf{y})| = \sqrt{|f_1(\mathbf{x}) - f_1(\mathbf{y})|^2 + \dots + |f_k(\mathbf{x}) - f_k(\mathbf{y})|^2}. \quad (3)$$

We propose a two-stage mating scheme illustrated in Fig. 1. The selection in the second stage (i.e., upper layer) is based on the similarity between solutions while the selection in the first stage (i.e., lower layer) uses the fitness value of each solution. Our mating scheme is applicable to any EMO algorithms because an arbitrary fitness definition can be directly used with no modification in its lower layer. For choosing the first parent (i.e., Parent A in Fig. 1), the standard fitness-based binary tournament selection with replacement is iterated α times for choosing α candidates (say $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_\alpha$). Next the center vector over the chosen α candidates is calculated in the objective space as

$$\bar{\mathbf{f}}(\mathbf{x}) = (\bar{f}_1(\mathbf{x}), \bar{f}_2(\mathbf{x}), \dots, \bar{f}_k(\mathbf{x})), \quad (4)$$

where

$$\bar{f}_i(\mathbf{x}) = \frac{1}{\alpha} \sum_{j=1}^{\alpha} f_i(\mathbf{x}_j) \quad \text{for } i = 1, 2, \dots, k. \quad (5)$$

Then the most dissimilar solution to the center vector $\bar{\mathbf{f}}(\mathbf{x})$ is chosen as Parent A in Fig. 1 among the α candidates. That is, the most extreme solution with the largest distance from the center vector $\bar{\mathbf{f}}(\mathbf{x})$ in the objective space is chosen as the first parent A in Fig. 1. When multiple solutions have the same largest distance, one solution is randomly chosen among them (i.e., random tiebreak). The choice of the first parent is illustrated for the case of $\alpha = 3$ in Fig. 2 (a) where three solutions $\mathbf{x}_1, \mathbf{x}_2$ and $\mathbf{x}_3$ are selected as candidates of the first parent. The most dissimilar solution $\mathbf{x}_3$ to the center vector $\bar{\mathbf{f}}(\mathbf{x})$ is chosen as the first parent in Fig. 2 (a).

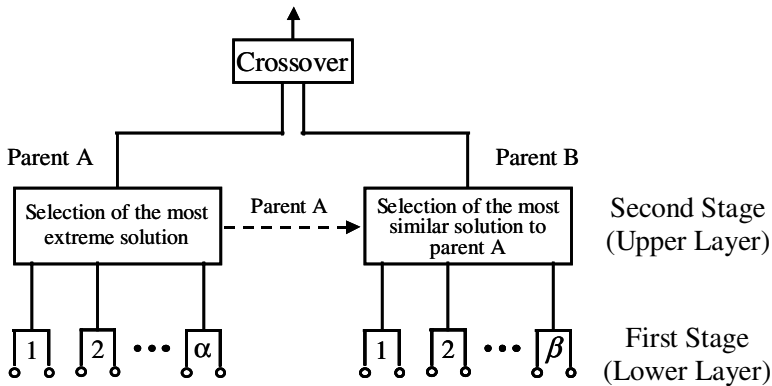


Fig. 1. The proposed mating scheme.

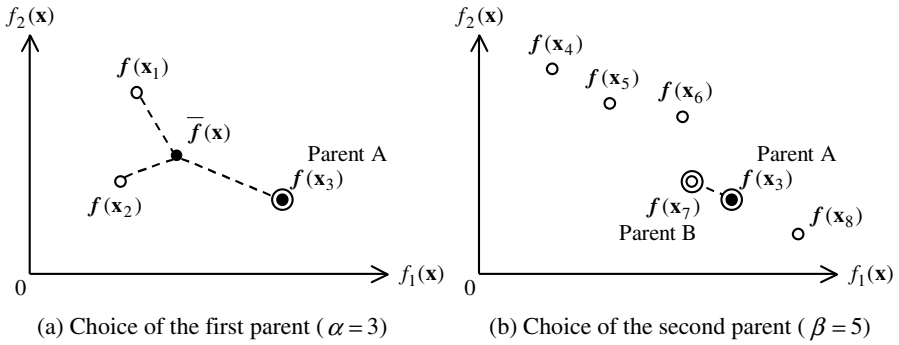


Fig. 2. Illustration of the proposed mating scheme.

When $\alpha = 1$, the choice of the first parent is the same as the standard binary tournament selection. The case of $\alpha = 2$ is actually the same as the standard binary tournament selection because two candidates always have the same distance from their center vector. Selection probabilities are biased toward extreme solutions only when $\alpha \geq 3$.

On the other hand, the standard fitness-based binary tournament selection with replacement is iterated β times for choosing β candidates of the second parent (i.e., Parent B in Fig. 1). Then the most similar solution to the first parent (i.e., Parent A in Fig. 1) is chosen as Parent B among the β candidates. That is, the solution with the smallest distance from Parent A is chosen. In this manner, similar parents are recombined in our mating scheme. The choice of the second parent is illustrated in Fig. 2 (b) for the case of $\beta = 5$. The most similar solution x_7 to the first parent (i.e., x_3) is selected as the second parent among the five candidates ($x_4, \dots, x_8$) in Fig. 2 (b). A crossover operation is applied to x_3 and x_7 for generating new solutions.

The mating scheme in our former study [11] corresponds to the case of $\alpha = 1$. That is, the first parent was chosen by the standard fitness-based binary tournament selection. Not only the choice of the most similar solution as the second parent but also the choice of the most dissimilar solution was examined. Experimental results in our former study [11] suggested that the choice of similar parents improved the convergence speed to the Pareto-front while it had a negative effect on the diversity of solutions. On the other hand, the choice of dissimilar parents improved the diversity of solutions while it had a negative effect on the convergence speed to the Pareto-front. The main motivation to propose the new mating scheme in Fig. 1 is to simultaneously improve the diversity and the convergence speed by appropriately choosing parents for recombination.

Our mating scheme has high applicability and high flexibility. The positive aspects of our mating scheme are summarized as follows:

- (1) Our mating scheme is applicable to any EMO algorithms because an arbitrary fitness definition can be directly used with no modification.
- (2) The specification of the mating radius σ_{mating} is not necessary.

- (3) The selection pressure toward extreme solutions is adjustable by the specification of the parameter α .
- (4) The selection pressure toward similar solutions is adjustable by the specification of the parameter β .
- (5) The proposed mating scheme has high flexibility. For example, not only the binary tournament selection but also other selection mechanisms can be used for choosing candidate solutions. The distance between solutions can be measured in the decision space as well as in the objective space. The choice of dissimilar parents can be also examined using our mating scheme.

On the other hand, the negative aspects of the proposed mating scheme are as follows:

- (i) Appropriate specifications of the two parameters α and β seem to be problem-dependent. The sensitivity of the performance of EMO algorithms on the parameter specifications will be examined in the next section.
- (ii) Additional computational load is required for performing our mating scheme in EMO algorithms. The increase in CPU time will be also examined in the next section. In general, the computational overhead caused by our mating scheme is negligible when the evaluation of each solution needs long CPU time.

3 Computational Experiments

In this section, we examine the effect of our mating scheme on the performance of EMO algorithms through computational experiments. For this purpose, we combined our mating scheme with recently developed well-known EMO algorithms: SPEA [21] and NSGA-II [4]. It should be noted that our mating scheme is the same as the standard binary tournament selection when the two parameters α and β are specified as $\alpha=1$ and $\beta=1$. In this case, the modified SPEA and NSGA-II algorithms with our mating scheme are the same as their original versions. Using 100 combinations of α and β (i.e., $\alpha=1,2,\dots,10$ and $\beta=1,2,\dots,10$), we examine the effect of our mating scheme on the performance of the EMO algorithms.

3.1 Test Problems and Parameter Specifications

In our computational experiments, we used four knapsack problems in Zitzler & Thiele [21]: two-objective 250-item, two-objective 500-item, three-objective 250-item, and three-objective 500-item test problems. Each solution in an m -item knapsack problem was coded as a binary string of the length m . Thus the search space size was 2^m . Each string was evaluated in the same manner as in Zitzler & Thiele [21].

The modified SPEA and NSGA-II algorithms with our mating scheme were applied to the four knapsack problems under the following parameter specifications:

Crossover probability: 0.8,
 Mutation probability: $1/m$ where m is the string length,
 Population size in NSGA-II: 200,
 Population size in SPEA: 100,
 Population size of the secondary population in SPEA: 100,
 Stopping condition: 2000 generations.

3.2 Performance Measures

Various performance measures have been proposed in the literature for evaluating a set of non-dominated solutions. As explained in Knowles & Corne [15], no single performance measure can simultaneously evaluate various aspects of a solution set. Moreover, some performance measures are not designed for simultaneously comparing many solution sets but for comparing two solution sets with each other.

For comparing various combinations of α and β , we use the average distance from each Pareto-optimal solution to its nearest solution in a solution set. This performance measure was used in Czyzak & Jaszkievicz [2] and referred to as $D1_R$ in Knowles & Corne [15]. The $D1_R$ measure needs all Pareto-optimal solutions of each test problem. For the two-objective 250-item and 500-item knapsack problems, the Pareto-optimal solutions are available from the homepage of the first author of [21]. For the three-objective 250-item and 500-item knapsack problems, we found near Pareto-optimal solutions using the SPEA and the NSGA-II. These algorithms were applied to each test problem using much longer CPU time and larger memory storage (e.g., 30000 generations with the population size 400 for the NSGA-II) than the other computational experiments (see Subsection 3.1). We also used a single-objective genetic algorithm with a secondary population where all the non-dominated solutions were stored with no size limitation. Each of the three objectives was used in the single-objective genetic algorithm. This algorithm was applied to each three-objective test problem 30 times (10 times for each objective using the same stopping condition as the NSGA-II: 30000 generations with the population size 400). The SPEA and the NSGA-II were also applied to each test problem 10 times. Thus we obtained 50 solution sets for each test problem. Then we chose non-dominated solutions from the obtained 50 solution sets as near Pareto-optimal solutions. The number of the Pareto-optimal or near Pareto-optimal solutions of each test problem in our computational experiments is as follows:

567 solutions (2/250 test problem),
 1427 solutions (2/500 test problem),
 2158 solutions (3/250 test problem),
 2142 solutions (3/500 test problem),

where the k/m test problem means the k -objective m -item test problem.

3.3 Experimental Results

The modified SPEA and NSGA-II algorithms with our mating scheme were applied to the four test problems using 100 combinations of α and β . For each combination, we performed ten runs from different initial populations for each test problem.

Average values of the $D1_R$ measure over ten runs are summarized in Figs. 3–6 where smaller values of the $D1_R$ measure (i.e., shorter bars) mean better results. In each figure, the result by the original EMO algorithm corresponds to the bar at the top-right corner where $\alpha = 1$ and $\beta = 1$. From these figures, we can see that our mating scheme improved the performance of the SPEA and the NSGA-II over a wide range of combinations of α and β . Especially the performance of the original EMO algorithms on the three-objective test problems (i.e., Fig. 5 and Fig. 6) was improved by our mating scheme in almost all combinations of α and β . This is also the case for the performance of the NSGA-II on the 2/500 test problem (i.e., Fig. 4 (b)). The significant deterioration in the performance was observed only when the value of α in the modified SPEA was too large in Fig. 3 (a) and Fig. 4 (a).

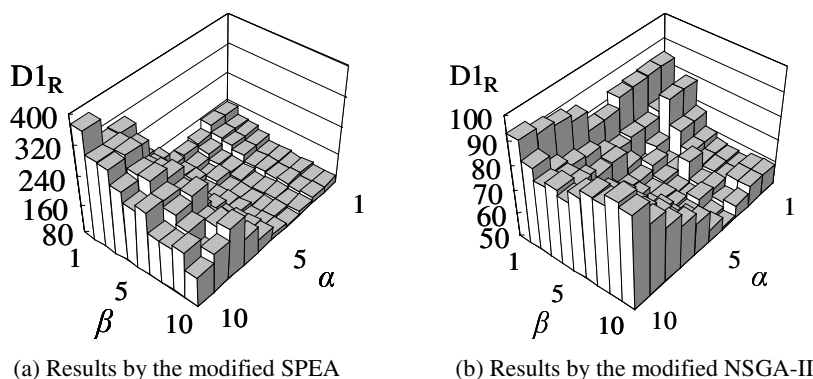


Fig. 3. Average values of the $D1_R$ measure for the two-objective 250-item problem.

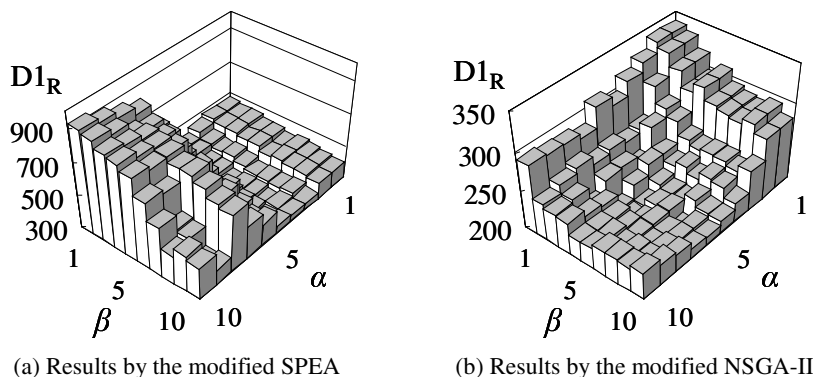
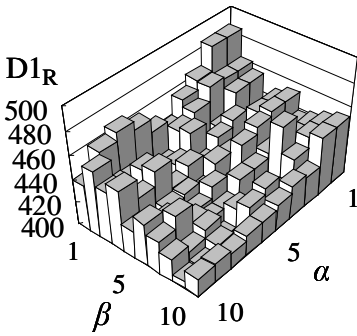
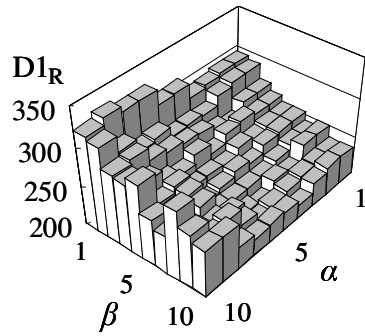


Fig. 4. Average values of the $D1_R$ measure for the two-objective 500-item problem.

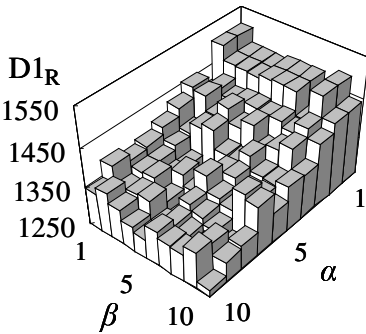


(a) Results by the modified SPEA

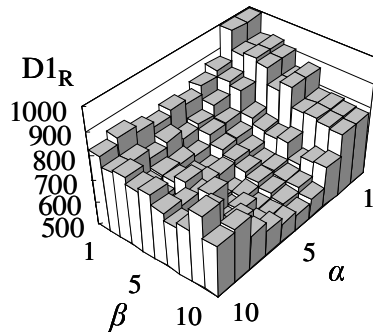


(b) Results by the modified NSGA-II

Fig. 5. Average values of the $D1_R$ measure for the three-objective 250-item problem.



(a) Results by the modified SPEA



(b) Results by the modified NSGA-II

Fig. 6. Average values of the $D1_R$ measure for the three-objective 500-item problem.

Using the Mann-Whitney U test, we examined the statistical significance of the improvement in the $D1_R$ measure by the proposed mating scheme. More specifically, the results by the original EMO algorithms (i.e., $\alpha = 1$ and $\beta = 1$) were compared with those by their modified versions (i.e., $\alpha \geq 2$ and/or $\beta \geq 2$) for examining the statistical significance of the improvement by the proposed mating scheme for three confidence levels (90%, 95% and 99%). Confidence levels of the improvement are summarized in Table 1 for the 2/250 test problem and Table 2 for the 3/500 test problem. From those tables, we can see that the performance of the SPEA and the NSGA-II was significantly improved by our mating scheme in many cases. As shown in our experimental results in Figs. 3-6 and Tables 1-2, the selection bias toward either extreme solutions (i.e., $\alpha \geq 3$ and $\beta = 1$) or similar parents (i.e., $\alpha = 1$ and $\beta \geq 2$) improved the performance of the EMO algorithms. It is, however, clearly shown by some experimental results (e.g., Fig. 4 (b), Fig. 5 (a) and Fig. 6 (a)) that the simultaneous use of them (i.e., $\alpha \geq 3$ and $\beta \geq 2$) improved their performance more significantly. For example, the best result was obtained in Fig. 6 (a) from the combination of $\alpha = 10$ and $\beta = 10$.

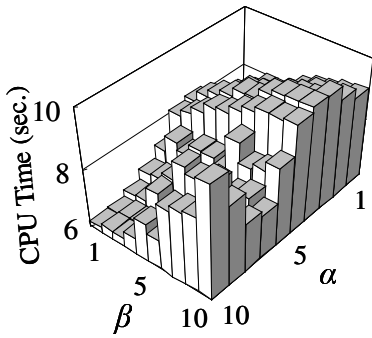
Table 1. Confidence levels of the improvement for the two-objective 250-item test problem.
 (* means that the corresponding confidence level is less than 90%)

(a) Results for the SPEA											(b) Results for the NSGA-II										
α	The value of β										α	The value of β									
	1	2	3	4	5	6	7	8	9	10		1	2	3	4	5	6	7	8	9	10
1	*	95	99	99	99	99	99	99	99	99	1	*	99	90	99	99	99	99	99	99	99
2	*	95	99	99	99	99	99	99	99	99	2	*	99	90	99	99	99	99	99	99	99
3	99	99	99	99	99	99	99	99	99	99	3	*	99	99	99	99	99	99	99	99	99
4	99	99	99	99	99	99	99	99	99	99	4	*	99	99	99	99	99	99	99	99	99
5	99	99	99	99	99	99	99	99	99	99	5	95	99	99	99	99	99	99	99	99	99
6	*	99	99	99	99	99	99	99	99	99	6	90	99	99	99	99	99	99	99	99	99
7	*	*	*	99	*	*	95	90	95	*	7	*	99	99	99	99	99	99	99	99	99
8	*	*	*	*	*	*	*	*	*	*	8	*	95	99	99	99	99	99	99	99	99
9	*	*	*	*	*	*	*	*	*	*	9	*	95	95	95	99	99	99	95	90	*
10	*	*	*	*	*	*	*	*	*	*	10	*	*	90	*	90	*	*	*	*	*

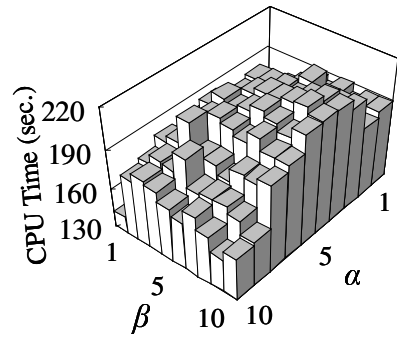
Table 2. Confidence levels of the improvement for the three-objective 500-item test problem.

(a) Results for the SPEA											(b) Results for the NSGA-II										
α	The value of β										α	The value of β									
	1	2	3	4	5	6	7	8	9	10		1	2	3	4	5	6	7	8	9	10
1	*	*	*	90	*	*	*	99	*	95	1	*	95	90	95	99	99	99	99	99	99
2	*	*	*	90	*	*	*	99	*	95	2	*	95	90	95	99	99	99	99	99	99
3	99	95	99	95	95	99	99	99	95	99	3	99	99	99	99	99	99	99	99	99	99
4	90	95	99	99	99	99	99	99	99	99	4	99	99	99	99	99	99	99	99	99	99
5	99	99	99	99	99	99	99	99	99	99	5	99	99	99	99	99	99	99	99	99	99
6	99	99	99	99	99	99	99	99	99	99	6	99	99	99	99	99	99	99	99	99	99
7	99	99	99	99	99	99	99	99	99	99	7	99	99	99	99	99	99	99	99	99	99
8	99	99	99	99	99	99	99	99	99	99	8	99	99	99	99	99	99	99	99	99	99
9	99	99	99	99	99	99	99	99	99	99	9	99	99	99	99	99	99	99	99	99	99
10	99	99	99	99	99	99	99	99	99	99	10	99	99	99	99	99	99	99	99	95	99

As mentioned in Section 2, additional computation load is required for executing our mating scheme. For evaluating such a computational overhead, we measured the average CPU time for each combination of α and β . Experimental results are summarized in Fig. 7 for the SPEA and Fig. 8 for the NSGA-II. Since the CPU time of the original and modified SPEA algorithms totally depends on the computation load for the clustering of non-dominated solutions in the secondary population, it is not easy to evaluate the pure effect of our mating scheme (see Fig. 7). On the other hand, the evaluation of the computational overhead caused by our mating scheme is easy for the NSGA-II as shown in Fig. 8 where we observe the linear increase in the average CPU time with the increase in the values of α and β . The increase in the average CPU time from the original NSGA-II with $\alpha = 1$ and $\beta = 1$ to the modified NSGA-II with $\alpha = 10$ and $\beta = 10$ was 15.7% in Fig. 8 (a) and 7.4% in Fig. 8 (b).

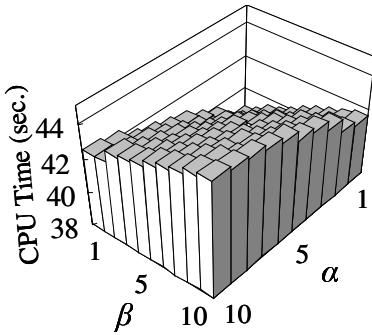


(a) Two-objective 250-item test problem

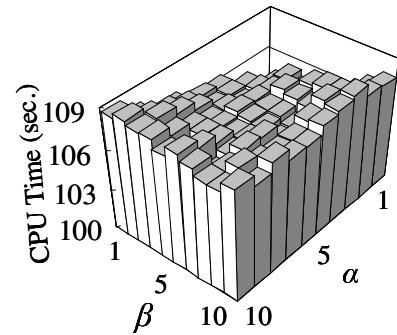


(b) Three-objective 500-item test problem

Fig. 7. Average CPU time of the original and modified SPEA.

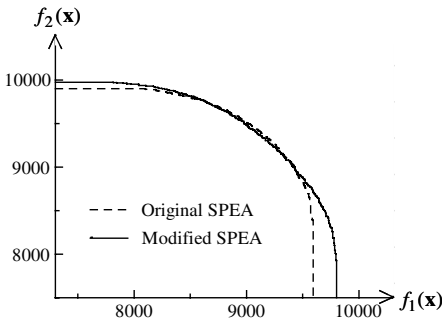


(a) Two-objective 250-item test problem

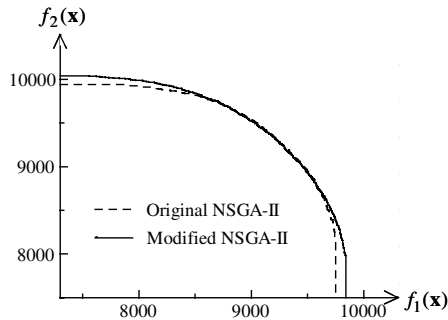


(b) Three-objective 500-item test problem

Fig. 8. Average CPU time of the original and modified NSGA-II.



(a) SPEA and its modified version



(b) NSGA-II and its modified version

Fig. 9. 50% attainment surface for the two-objective 250-item test problem.

For visually demonstrating the improvement in the performance of the EMO algorithms by our mating scheme, we show the 50% attainment surface (e.g., see [3]) obtained by the original EMO algorithms and the modified EMO algorithms in Fig. 9

for the 2/250 problem. The best values of α and β with the smallest DI_R measures in Fig. 3 were used in Fig. 9 for the modified SPEA and NSGA-II algorithms. We can see from Fig. 9 that better attainment surfaces were obtained by the modified algorithms. Similar improvement was also observed for the 2/500 problem.

4 Concluding Remarks

We proposed a two-stage mating scheme for simultaneously improving the diversity of solutions and the convergence speed to the Pareto-front. The basic idea is to bias selection probabilities toward extreme solutions for preserving the diversity and toward similar parents for improving the convergence speed. The effect of our mating scheme was examined through computational experiments on multiobjective knapsack problems where our mating scheme was combined with the two well-known EMO algorithms (i.e., SPEA and NSGA-II). It was shown that the performance of those EMO algorithms was improved by our mating scheme. It was also shown that the increase in the CPU time caused by our mating scheme was small if compared with the total CPU time (e.g., 7.4% increase). The simultaneous improvement in the diversity and the convergence speed is usually very difficult. This is also the case in our mating scheme. In our mating scheme, the two parameters (i.e., α and β) should be carefully adjusted to strike a balance between the diversity and the convergence speed. Further studies are needed for automatically specifying these parameter values appropriately. Further studies are also needed for examining the effectiveness of our mating scheme for recently developed other EMO algorithms such as SPEA2 [22].

Our mating scheme can be viewed as assigning a selection probability to each pair of solutions (not to each individual solution). Pairs of similar solutions tend to have higher selection probabilities than those of dissimilar solutions. At the same time, pairs of extreme solutions tend to have higher selection probabilities than those of prototypical solutions. While various sophisticated methods for assigning a fitness value to each individual solution have been proposed in the literature on EMO algorithms, the assignment of a fitness value (or a selection probability) to each pair of solutions has not been studied well. Experimental results in this paper clearly show that such an idea of fitness assignment has a possibility to improve EMO algorithms with sophisticated fitness assignment schemes to each individual solution.

The authors would like to thank the financial support from Japan Society for the Promotion of Science (JSPS) through Grand-in-Aid for Scientific Research (B): KAKENHI (14380194).

References

1. Coello Coello, C. A., Van Veldhuizen, D. A., and Lamont, G. B.: *Evolutionary Algorithms for Solving Multi-Objective Problems*, Kluwer Academic Publishers, Boston (2002).
2. Czyzak, P., and Jaskiewicz, A.: Pareto-Simulated Annealing – A Metaheuristic Technique for Multi-Objective Combinatorial Optimization, *Journal of Multi-Criteria Decision Analysis* 7 (1998) 34–47.

3. Deb, K.: *Multi-Objective Optimization Using Evolutionary Algorithms*, John Wiley & Sons, Chichester (2001).
4. Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T.: A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II, *IEEE Trans. on Evolutionary Computation* 6 (2002) 182–197.
5. Fonseca, C. M., and Fleming, P. J.: Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization, *Proc. of 5th International Conference on Genetic Algorithms* (1993) 416–423.
6. Fonseca, C. M., and Fleming, P. J.: An Overview of Evolutionary Algorithms in Multiobjective Optimization, *Evolutionary Computation* 3 (1995) 1–16.
7. Goldberg, D. E.: *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, Reading (1989).
8. Hajela, P., and Lin, C. Y.: Genetic Search Strategies in Multicriterion Optimal Design, *Structural Optimization* 4 (1992) 99–107.
9. Horn, J., Nafpliotis, N., and Goldberg, D. E.: A Niche Pareto Genetic Algorithm for Multi-Objective Optimization, *Proc. of 1st IEEE International Conference on Evolutionary Computation* (1994) 82–87.
10. Ishibuchi, H., and Murata, T.: A Multi-Objective Genetic Local Search Algorithm and Its Application to Flowshop Scheduling, *IEEE Trans. on Systems, Man, and Cybernetics - Part C: Applications and Reviews* 28 (1998) 392–403.
11. Ishibuchi, H., and Shibata, Y.: An Empirical Study on the Effect of Mating Restriction on the Search Ability of EMO Algorithms, *Proc. of Second International Conference on Evolutionary Multi-Criterion Optimization* (2003 in press).
12. Ishibuchi, H., Yoshida, T., and Murata, T.: Balance between Genetic Search and Local Search in Memetic Algorithms for Multiobjective Permutation Flowshop Scheduling, *IEEE Trans. on Evolutionary Computation* (2003 in press).
13. Jaskiewicz, A.: Genetic Local Search for Multi-Objective Combinatorial Optimization, *European Journal of Operational Research* 137 (2002) 50–71.
14. Knowles, J. D., and Corne, D. W.: M-PAES: A Memetic Algorithm for Multiobjective Optimization, *Proc. of 2000 Congress on Evolutionary Computation* (2000) 325–332.
15. Knowles, J. D., and Corne, D. W.: On Metrics for Comparing Non-Dominated Sets, *Proc. of 2002 Congress on Evolutionary Computation* (2002) 711–716.
16. Schaffer, J. D.: Multiple Objective Optimization with Vector Evaluated Genetic Algorithms, *Proc. of 1st International Conference on Genetic Algorithms and Their Applications* (1985) 93–100.
17. Van Veldhuizen, D. A., and Lamont, G. B.: Multiobjective Evolutionary Algorithms: Analyzing the State-of-the-Art, *Evolutionary Computation* 8 (2000) 125–147.
18. Watanabe, S., Hiroyasu, T., and Miki, M.: LCGA: Local Cultivation Genetic Algorithm for Multi-Objective Optimization Problem, *Proc. of 2002 Genetic and Evolutionary Computation Conference* (2002) 702.
19. Zitzler, E., Deb, K., and Thiele, L.: Comparison of Multiobjective Evolutionary Algorithms: Empirical Results, *Evolutionary Computation* 8 (2000) 173–195.
20. Zitzler, E., and Thiele, L.: Multiobjective Optimization using Evolutionary Algorithms – A Comparative Case Study, *Proc. of 5th International Conference on Parallel Problem Solving from Nature* (1998) 292–301.
21. Zitzler, E., and Thiele, L.: Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach, *IEEE Transactions on Evolutionary Computation* 3 (1999) 257–271.
22. Zitzler, E., Laumanns, M., and Thiele, L.: SPEA2: Improving the Performance of the Strength Pareto Evolutionary Algorithm, *Technical Report 103*, Computer Engineering and Communication Networks Lab, Swiss Federal Institute of Technology, Zurich (2001).

Evolutionary Multiobjective Optimization for Generating an Ensemble of Fuzzy Rule-Based Classifiers

Hisao Ishibuchi and Takashi Yamamoto

Department of Industrial Engineering, Osaka Prefecture University,
1-1 Gakuen-cho, Sakai, Osaka 599-8531, Japan
{hisaoi, yama}@ie.osakafu-u.ac.jp

Abstract. One advantage of evolutionary multiobjective optimization (EMO) algorithms over classical approaches is that many non-dominated solutions can be simultaneously obtained by their single run. In this paper, we propose an idea of using EMO algorithms for constructing an ensemble of fuzzy rule-based classifiers with high diversity. The classification of new patterns is performed based on the vote of multiple classifiers generated by a single run of EMO algorithms. Even when the classification performance of individual classifiers is not high, their ensemble often works well. The point is to generate multiple classifiers with high diversity. We demonstrate the ability of EMO algorithms to generate various non-dominated fuzzy rule-based classifiers with high diversity by their single run. Through computational experiments on some well-known benchmark data sets, it is shown that the vote of generated fuzzy rule-based classifiers leads to high classification performance on test patterns.

1 Introduction

A promising approach to the design of reliable classifiers is to combine multiple classifiers into a single one [2], [6]. Several methods have been proposed for generating multiple classifiers such as bagging [3] and boosting [8]. In the **bagging** (**bootstrap aggregating**) algorithm of Breiman [3], different data sets are generated by bootstrapping (i.e., random sampling with replacement from the whole data set) for the design of multiple classifiers. Thus the design of multiple classifiers can be performed in parallel. On the other hand, multiple classifiers are sequentially designed in boosting methods such as the **AdaBoost** (**Adaptive Boosting**) algorithm of Freund & Schapire [8]. After one classifier is designed, the weight of each training pattern is updated based on the classification result (i.e., correct classification or misclassification) in the AdaBoost algorithm. The training patterns with the updated weights are used for designing another classifier. The design of a classifier and the weight update of training patterns are iterated for generating multiple classifiers. Classifier aggregation has been studied in various fields [19], [21]. For example, evolutionary computation is used for generating multiple classifiers [20], [22]. In the

field of neural networks, the aggregation of multiple classifiers is often referred to as “mixture of local experts [17], [18]”. Classifier aggregation has also been studied in the field of fuzzy logic [4], [12].

The point in classifier aggregation is to generate an ensemble of classifiers with high diversity. Ideally the classification errors by each classifier should be uncorrelated. In this paper, we propose the use of evolutionary multiobjective optimization (EMO) algorithms for generating an ensemble of classifiers with high diversity. In our computational experiments, we apply the NSGA-II algorithm of Deb et al. [5] to a three-objective rule selection problem [13] for generating a number of non-dominated fuzzy rule-based classifiers with respect to the classification accuracy on training patterns, the number of fuzzy rules, and the total length of fuzzy rules. Of course, we can apply other EMO algorithms to our task. We use the NSGA-II because its implementation is relatively easy and its high performance is well-known [5].

One advantage of EMO algorithms over classical approaches is that many non-dominated solutions (i.e., classifiers in the context of this paper) can be obtained by their single run. That is, multiple classifiers are obtained by applying an EMO algorithm to training patterns just once. Through computational experiments on some well-known benchmark data sets, it is shown that high classification performance on test patterns (i.e., high generalization ability) can be obtained from the vote of non-dominated fuzzy rule-based classifiers. That is, we can design a high-performance aggregated fuzzy rule-based classifier using an EMO algorithm for generating multiple classifiers and the majority rule for classifying new patterns.

In this paper, we first briefly describe fuzzy rule-based classifiers in Section 2. Then we explain our two-stage approach [15], [16] to the design of fuzzy rule-based classifiers in Section 3. In the first stage, a pre-specified number of fuzzy rules are extracted as candidate rules from training patterns using a data mining technique. In the second stage, a number of non-dominated rule sets are found from the candidate rules by the NSGA-II algorithm. Experimental results on some well-known benchmark data sets are reported in Section 4 where the generalization ability of an ensemble of obtained non-dominated rule sets for each data set is examined using the majority rule for classifying new patterns. Finally Section 5 summarizes this paper.

2 Fuzzy Rule-Based Classifiers

Let us assume that we have m training patterns $\mathbf{x}_p = (x_{p1}, \dots, x_{pn})$, $p = 1, 2, \dots, m$ from M classes where x_{pi} is the attribute value of the p -th training pattern for the i -th attribute ($i = 1, 2, \dots, n$). For our n -dimensional M -class pattern classification problem, we use fuzzy rules of the following form:

$$\text{Rule } R_q: \text{ If } x_1 \text{ is } A_{q1} \text{ and } \dots \text{ and } x_n \text{ is } A_{qn} \text{ then Class } C_q \text{ with } CF_q, \quad (1)$$

where R_q is the label of the q -th rule, $\mathbf{x} = (x_1, \dots, x_n)$ is an n -dimensional pattern vector, A_{qi} is an antecedent fuzzy set, C_q is a class label, and CF_q is a rule weight.

We define the compatibility grade of each training pattern $\mathbf{x}_p$ with the antecedent part $\mathbf{A}_q = (A_{q1}, \dots, A_{qn})$ of the fuzzy rule R_q in (1) using the product operator as

$$\mu_{\mathbf{A}_q}(\mathbf{x}_p) = \mu_{A_{q1}}(x_{p1}) \cdot \mu_{A_{q2}}(x_{p2}) \cdot \dots \cdot \mu_{A_{qn}}(x_{pn}), \quad p = 1, 2, \dots, m, \quad (2)$$

where $\mu_{A_{qi}}(\cdot)$ is the membership function of A_{qi} .

For determining the consequent class C_q and the rule weight CF_q , we first calculate the confidence of the fuzzy association rule “ $\mathbf{A}_q \Rightarrow \text{Class } h$ ” for each class h by extending its original definition for non-fuzzy association rules [1] as

$$c(\mathbf{A}_q \Rightarrow \text{Class } h) = \frac{\sum_{\mathbf{x}_p \in \text{Class } h} \mu_{\mathbf{A}_q}(\mathbf{x}_p)}{\sum_{p=1}^m \mu_{\mathbf{A}_q}(\mathbf{x}_p)}, \quad h = 1, 2, \dots, M. \quad (3)$$

The confidence $c(\cdot)$ can be viewed as a fuzzy conditional probability of Class h . The consequent class C_q is specified as the class with the maximum confidence:

$$c(\mathbf{A}_q \Rightarrow \text{Class } C_q) = \max \{c(\mathbf{A}_q \Rightarrow \text{Class } h) \mid h = 1, 2, \dots, M\}. \quad (4)$$

On the other hand, the rule weight CF_q is specified as

$$CF_q = c(\mathbf{A}_q \Rightarrow \text{Class } C_q) - \sum_{\substack{h=1 \\ h \neq C_q}}^M c(\mathbf{A}_q \Rightarrow \text{Class } h). \quad (5)$$

The rule weight of each fuzzy rule has a large effect on the classification ability of fuzzy rule-based classifiers [11].

Let S be a fuzzy rule-based classifier (i.e., a set of fuzzy rules). When an input pattern $\mathbf{x}_p$ is to be classified, a single winner rule R_w is chosen from the rule set S as

$$\mu_{\mathbf{A}_w}(\mathbf{x}_p) \cdot CF_w = \max \{\mu_{\mathbf{A}_q}(\mathbf{x}_p) \cdot CF_q \mid R_q \in S\}. \quad (6)$$

The input pattern $\mathbf{x}_p$ is assigned to the consequent class C_w of the winner rule R_w .

In this paper, we use multiple fuzzy rule-based classifiers. An input pattern is classified by each individual classifier using the single winner-based method as shown in (6). Then the final classification is performed through the majority rule (i.e., simple majority vote scheme) based on the classification result by each individual classifier (see [12] for various voting methods for fuzzy rule-based classification).

3 Heuristic Rule Extraction and Genetic Rule Selection

Genetic rule selection was proposed for designing fuzzy rule-based classifiers with high accuracy and high comprehensibility in [14] where a scalar fitness function was defined as a weighted sum of two objectives: to maximize the number of correctly classified training patterns and to minimize the number of fuzzy rules. A two-

objective genetic algorithm was used in [10] for finding non-dominated rule sets with respect to these two objectives. Genetic rule selection was further extended to the following three-objective optimization problem in [13]:

$$\text{Maximize } f_1(S), \text{ minimize } f_2(S), \text{ and minimize } f_3(S), \quad (7)$$

where S is a subset of candidate rules, $f_1(S)$ is the number of correctly classified training patterns by the rule set S , $f_2(S)$ is the number of fuzzy rules in S , and $f_3(S)$ is the total rule length of fuzzy rules in S . The number of antecedent conditions of each fuzzy rule is referred to as the rule length. It should be noted that the third objective $f_3(S)$ is not the average rule length but the total rule length. While we use the average rule length for describing each rule set in some parts of this paper, its use as $f_3(S)$ leads to many meaningless non-dominated rule sets [13].

A two-stage approach to the three-objective fuzzy rule selection problem in (7) was proposed for handling high-dimensional classification problems in [15], [16]. This approach is briefly explained in this section (for details, see [15], [16]).

3.1 Heuristic Rule Extraction

When we use K linguistic values and “*don’t care*” as antecedent fuzzy sets for each of n attributes, the total number of possible combinations of those $(K + 1)$ antecedent fuzzy sets is $(K + 1)^n$. Among those combinations, a pre-specified number of candidate rules are generated in a heuristic manner using a data mining criterion. In the field of data mining, association rules are often evaluated by two rule evaluation criteria: support and confidence. In the same manner as the fuzzy version of the confidence in (3), the definition of the support [1] is also extended as

$$s(\mathbf{A}_q \Rightarrow \text{Class } h) = \frac{1}{m} \sum_{\mathbf{x}_p \in \text{Class } h} \mu_{\mathbf{A}_q}(\mathbf{x}_p). \quad (8)$$

The support $s(\cdot)$ can be viewed as measuring the coverage of training patterns by the fuzzy rule. We use the following rule evaluation criterion in this paper:

$$f_{\text{SLAVE}}(R_q) = s(\mathbf{A}_q \Rightarrow \text{Class } C_q) - \sum_{\substack{h=1 \\ h \neq C_q}}^M s(\mathbf{A}_q \Rightarrow \text{Class } h). \quad (9)$$

This is a modified version of a rule evaluation criterion used in an iterative fuzzy GBML (genetics-based machine learning) algorithm called SLAVE [9].

In our heuristic rule extraction, a pre-specified number of candidate rules with the largest values of the SLAVE criterion are found for each class. For designing fuzzy rule-based classifiers with high comprehensibility, only short fuzzy rules are examined as candidate rules. This restriction on the rule length is consistent with the third objective (i.e., the total rule length) of our three-objective rule selection problem.

3.2 Genetic Rule Selection

Let us assume that N fuzzy rules have been extracted as candidate rules using the SLAVE criterion. A subset S of the N candidate rules is handled as an individual in EMO algorithms, which is represented by a binary string of the length N as

$$S = s_1 s_2 \cdots s_N, \quad (10)$$

where $s_j = 1$ and $s_j = 0$ mean that the j -th candidate rule is included in S and excluded from S , respectively.

As in our former studies [15], [16], we use two problem-specific heuristic tricks together with the NSGA-II [5] for efficiently finding non-dominated rule sets. One trick is the biased mutation where a larger probability is assigned to the mutation from 1 to 0 than that from 0 to 1. This is for efficiently decreasing the number of fuzzy rules in each rule set. The other trick is the removal of unnecessary rules, which is a kind of local search. Since we use the single winner-based method for classifying each pattern by the rule set S , some fuzzy rules in S may be chosen as winner rules for no training patterns. We can remove those fuzzy rules without degrading the first objective (i.e., the number of correctly classified training patterns). At the same time, the second objective (i.e., the number of fuzzy rules) and the third objective (i.e., the total rule length) are improved by removing unnecessary rules. Thus we remove all fuzzy rules that are not selected as winner rules for any training patterns from the rule set S . The removal of unnecessary rules is performed after the first objective is calculated for each rule set and before the second and third objectives are calculated.

4 Computational Experiments

4.1 Data Sets

We use six data sets with many numerical attributes: Wisconsin breast cancer, Diabetes, Glass, Cleveland heart disease, Sonar, and Wine. These data sets are available from the UCI ML repository (<http://www.ics.uci.edu/~mllearn/>). In our former study [16], we examined the performance of individual non-dominated rule sets (i.e., individual fuzzy rule-based classifiers) on each data set. In this paper, we examine the performance of their ensemble (i.e., their aggregation using the simple majority vote scheme). We evaluate the performance of the aggregated classifier on each data set by comparing it with the reported results on the same data set in Elomaa & Rousu [7] where six variants of the C4.5 algorithm [23] were examined. The performance of each variant was evaluated by ten independent iterations (with different data partitions) of the whole ten-fold cross-validation (10-CV) procedure (i.e., 10×10 -CV) in [7]. We use the same performance evaluation procedure as [7].

Incomplete patterns with missing values are included in the Wisconsin breast cancer data set and the Cleveland heart disease data set. Those patterns were not used in our computational experiments as in [16]. See UCI ML repository and [16] for details of each data set.

4.2 Experimental Conditions

As in Elomaa & Rousu [7] and our former study [16], we iterated the whole 10-CV procedure ten times using different data partitions into ten subsets. Since the whole 10-CV procedure consisted of ten iterations of the design of a classifier ensemble and its performance evaluation, the NSGA-II was employed 100 times for each data set. A number of non-dominated rule sets were simultaneously obtained from each run of the NSGA-II. Among those non-dominated rule sets, too small rule sets were excluded from the classifier ensemble. More specifically, we used the number of classes as the lower bound on the number of fuzzy rules. That is, we excluded non-dominated rule sets with less than M fuzzy rules for an M -class classification problem. After the classifier ensemble was designed, each pattern was independently classified by each individual non-dominated rule set in the ensemble. Then the majority class was chosen as the final classification result of that pattern by the classifier ensemble.

Our computational experiments in this paper were performed in the same manner as in our former study [16] where the performance of individual non-dominate rule sets was examined. Here we briefly describe the experimental conditions (for details, see [16]). All attribute values of each data set were normalized into real numbers in the unit interval $[0, 1]$. As antecedent fuzzy sets, we used “*don’t care*” and 14 triangular fuzzy sets generated from four fuzzy partitions with different granularities in Fig. 1. We generated 300 fuzzy rules for each class as candidate rules in a greedy manner using the SLAVE criterion. Thus the total number of candidate rules was $300M$ where M is the number of classes. The upper bound on the length of candidate rules was two for the Sonar data set and three for the other data sets.

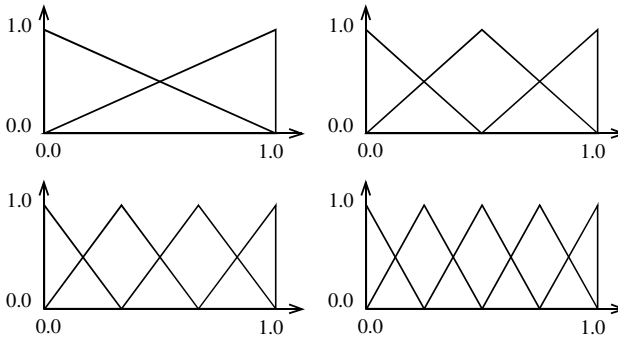


Fig. 1. Four fuzzy partitions used in our computer simulations.

The NSGA-II was employed for finding non-dominated rule sets from $300M$ candidate rules. We used the following parameter values in the NSGA-II:

Population size: 200 strings,
Crossover probability: 0.8,

Biased mutation probabilities: $p_m(0 \rightarrow 1) = 1/300M$ and $p_m(1 \rightarrow 0) = 0.1$,
 Stopping condition: 5000 generations.

4.3 Experimental Results

Wisconsin Breast Cancer Data Set: Average error rates by our classifier ensembles on training patterns and test patterns are shown by the solid lines in Fig. 2 (a) and Fig. 2 (b), respectively. They were 2.32% on training patterns in Fig. 2 (a) and 3.75% on test patterns in Fig. 2 (b). Average error rates of individual classifiers (i.e., individual non-dominated rule sets) are shown by closed and open circles in each figure. Closed circles are used for indicating individual classifiers with low error rates on training patterns. As shown in Fig. 2 (and other figures in this paper), individual classifiers with low error rates on training patterns did not always have low error rates on test patterns. This makes it very difficult to choose a single classifier from multiple alternative ones. The aggregation of many classifiers using a voting scheme avoids this difficult task (i.e., the choice of a single classifier). The performance of individual classifiers (i.e., closed and open circles) was examined in our former study [16]. For the Wisconsin breast cancer data set, the best and worst error rates among the six variants of the C4.5 algorithm were reported as 5.1% and 6.0% by Elomaa & Rousu [7], respectively. Those results are shown by the two dotted lines in Fig. 2 (b).

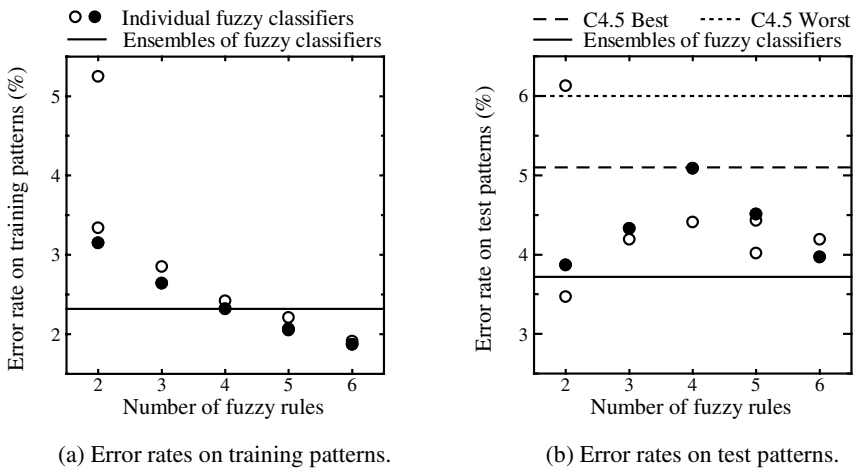


Fig. 2. Experimental results on the Wisconsin breast cancer data set.

From Fig. 2 (b), we can see that the performance of our classifier ensembles was much better than the best result of the C4.5 algorithm in [7]. We can also see that the performance of our classifier ensembles was better than many individual classifiers while it was slightly inferior to the best individual classifier in Fig. 2 (b).

Diabetes Data Set: In the same manner as Fig. 2, experimental results on the diabetes data set are summarized in Fig. 3. As in Fig. 2 (b), we can observe a positive effect of aggregating multiple non-dominated rule sets in Fig. 3 (b). That is, the performance of our classifier ensembles (i.e., 25.5% error rate) was better than many individual classifiers. We can also see that the performance of our classifier ensembles was close to the reported best result in [7] by the C4.5 algorithm (i.e., 25.0% error rate) and much better than the worst result (i.e., 27.2% error rate).

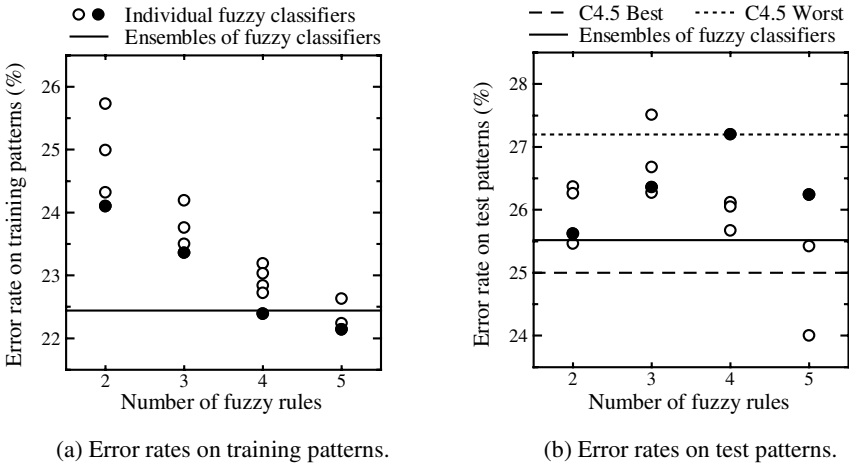


Fig. 3. Experimental results on the diabetes data set.

Glass Data Set: Experimental results on the glass data set are summarized in Fig. 4. Since the performance of our classifier ensembles was significantly inferior to the reported results of the C4.5 algorithm in [7], the best result (27.3%) of the C4.5 algorithm is not shown in Fig. 4 (b). To the best of our knowledge, good results have not been reported on the glass data set by descriptive fuzzy rules of the form in (1). Thus we feel that descriptive fuzzy rules with homogeneous fuzzy partition are not suitable for the glass data set. Further studies may be required for improving the performance of descriptive fuzzy rules on the glass data set.

Cleveland Heart Disease Data Set: Experimental results on the Cleveland heart disease data set are summarized in Fig. 5. The average error rate of our classifier ensembles on test patterns in Fig. 5 (b) was 46.6%. The best and worst reported results of the C4.5 algorithm in [7] were 46.3% and 47.9%, respectively.

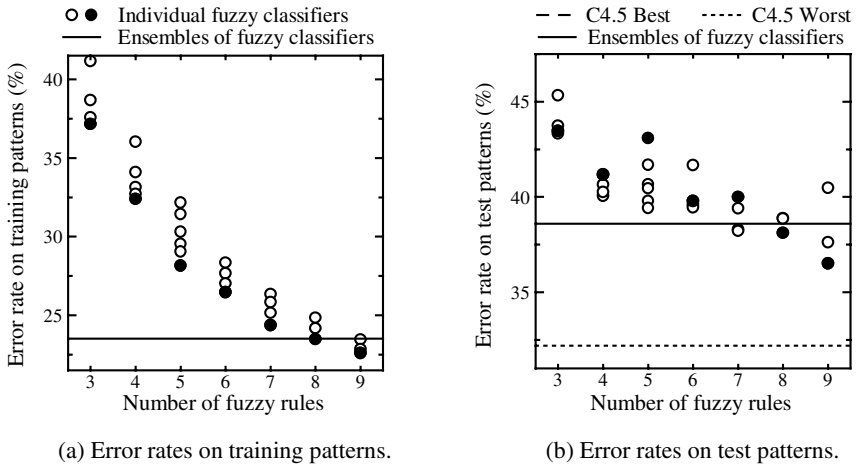


Fig. 4. Experimental results on the glass data set.

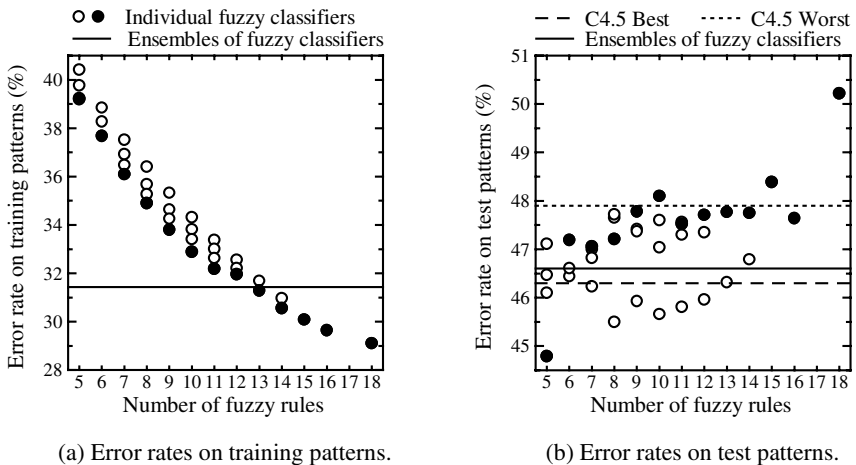


Fig. 5. Experimental results on the Cleveland heart disease data set.

Sonar Data Set: Experimental results on the sonar data set are summarized in Fig. 6. The reported worst result (35.8%) of the C4.5 algorithm in [7] is not shown in Fig. 6 (b) because it is out of the range of the figure. In Fig. 6 (b), the average error rate of our classifier ensembles was 22.74%, which outperformed almost all the individual classifiers and the best result (24.6%) of the C4.5 algorithm in [7].

Wine Data Set: Experimental results on the wine data set are summarized in Fig. 7. The best and worst results of the C4.5 algorithm reported in [7] were 5.6% and 8.8%, respectively. The average error rate of our classifier ensembles on test patterns was 4.21%, which was better than the reported best result of the C4.5 algorithm. In

Fig. 7 (b), the performance of our classifier ensembles was better than many individual classifiers while it was slightly inferior to the best individual classifier.

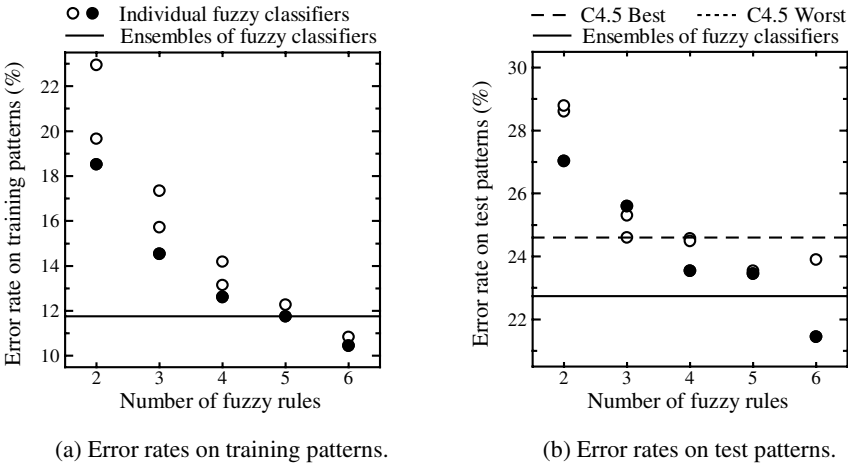


Fig. 6. Experimental results on the sonar data set.

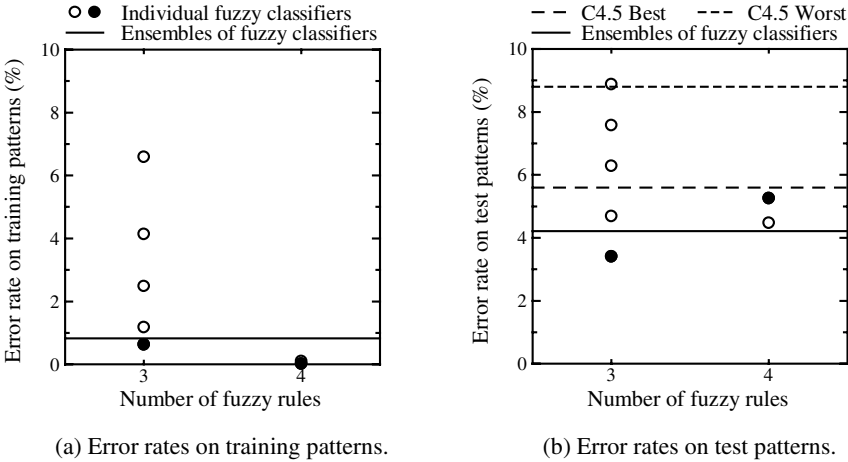


Fig. 7. Experimental results on the wine recognition data set.

5 Concluding Remarks

We proposed an idea of using EMO algorithms for designing an ensemble of classifiers with high diversity. EMO algorithms seem to be suitable for this task

because a number of classifiers can be simultaneously obtained from their single run. Moreover, many EMO algorithms have some mechanisms for maintaining the diversity of populations (i.e., maintaining the diversity of classifiers). In our computational experiments, we generated a number of non-dominated fuzzy rule-based classifiers by applying the NSGA-II algorithm to the three-objective fuzzy rule selection problem. Of course, other EMO algorithms are applicable to our classifier generation task. Experimental results on six well-known benchmark data sets showed that the performance of classifier ensembles was better than many individual classifiers. It was also shown that the performance of classifier ensembles was comparable with or superior to the reported best results of the C4.5 algorithm in [7] for five benchmark data sets (except for the glass data set).

Our experimental results in this paper suggest that the use of EMO algorithms is a promising approach to the design of classifier ensembles. Moreover, the aggregation of non-dominated classifiers avoids the difficult task of choosing a single classifier from multiple alternatives. As shown in many figures in this paper (and discussed in our former study [16]), low error rates of classifiers on training patterns do not always mean low error rates on test patterns. On the contrary, the minimization of the error rate on training patterns often leads to the deterioration in the error rate on test patterns due to the overfitting to training patterns. Thus it is very difficult to choose a single classifier from multiple alternatives based on their classification performance on training patterns. The proposed idea avoids this difficulty. The proposed idea can be also used as a simple performance measure of EMO algorithms for classification problems because the performance of many non-dominated classifiers can be summarized as an aggregated scalar measure (i.e., the average error rate of their ensemble).

Our experimental results can be further improved in several manners because we used very simple settings for generating classifier ensembles and classifying new patterns. For example, careful selection of classifiers from non-dominated rule sets may improve the performance of classifier ensembles. Adjustment of rule weights and/or membership functions in each individual classifier may also improve the performance of classifier ensembles. We can also use a weighted vote scheme (or other voting schemes [19], [21]) instead of the simple majority vote scheme for improving the performance of ensembles.

The authors would like to thank the financial support from Japan Society for the Promotion of Science (JSPS) through Grand-in-Aid for Scientific Research (B): KAKENHI (14380194).

References

1. Agrawal, R., Mannila, H., Srikant, R., Toivonen, H., and Verkamo, A. I.: Fast Discovery of Association Rules, in Fayyad, U. M., Piatetsky-Shapiro, G., Smyth, P., and Uthurusamy, R. (eds.) *Advances in Knowledge Discovery and Data Mining*, AAAI Press, Menlo Park (1996) 307–328.
2. Bauer, E., and Kohavi, R.: An Empirical Comparison of Voting Classification Algorithms: Bagging, Boosting, and Variants, *Machine Learning* 36 (1999) 105–139.
3. Breiman, L.: Bagging Predictors, *Machine Learning* 24 (1996) 123–140.

4. Cho, S. B., and Kim, J. H.: Combining Multiple Neural Networks by Fuzzy Integral for Robust Classification, *IEEE Trans. on Systems, Man, and Cybernetics* 25 (1995) 380–384.
5. Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T.: A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II, *IEEE Trans. on Evolutionary Computation* 6 (2002) 182–197.
6. Dietterich, T. G.: An Experimental Comparison of Three Methods for Constructing Ensembles of Decision Trees: Bagging, Boosting, and Randomization, *Machine Learning* 40 (2000) 139–157.
7. Elomaa, T., and Rousu, J.: General and Efficient Multisplitting of Numerical Attributes, *Machine Learning* 36 (1999) 201–244.
8. Freund, Y., and Schapire, R. E.: A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting, *Journal of Computer and System Sciences* 55 (1997) 119–139.
9. Gonzalez, A., and Perez, R.: SLAVE: A Genetic Learning System Based on an Iterative Approach, *IEEE Trans. on Fuzzy Systems* 7 (1999) 176–191.
10. Ishibuchi, H., Murata, T., and Turksen, I. B.: Single-Objective and Two-Objective Genetic Algorithms for Selecting Linguistic Rules for Pattern Classification Problems, *Fuzzy Sets and Systems* 89 (1997) 135–149.
11. Ishibuchi, H., and Nakashima, T.: Effect of Rule Weights in Fuzzy Rule-Based Classification Systems, *IEEE Trans. on Fuzzy Systems* 9 (2001) 506–515.
12. Ishibuchi, H., Nakashima, T., and Morisawa, T.: Voting in Fuzzy Rule-Based Systems for Pattern Classification Problems, *Fuzzy Sets and Systems* 103 (1999) 223–238.
13. Ishibuchi, H., Nakashima, T., and Murata, T.: Three-Objective Genetics-Based Machine Learning for Linguistic Rule Extraction, *Information Sciences* 136 (2001) 109–133.
14. Ishibuchi, H., Nozaki, K., Yamamoto, N., and Tanaka, H.: Selecting Fuzzy If-Then Rules for Classification Problems Using Genetic Algorithms, *IEEE Trans. on Fuzzy Systems* 3 (1995) 260–270.
15. Ishibuchi, H., and Yamamoto, T.: Fuzzy Rule Selection by Data Mining Criteria and Genetic Algorithms, *Proc. of Genetic and Evolutionary Computation Conference* (2002) 399–406.
16. Ishibuchi, H., and Yamamoto, T.: Effects of Three-Objective Genetic Rule Selection on the Generalization Ability of Fuzzy Rule-based Systems, *Proc. of 2nd International Conference on Evolutionary Multi-criteria Optimization* (2003) (in press).
17. Jacobs, R. A., Jordan, M. I., Nowlan, S. J., and Hinton G. E.: Adaptive Mixtures of Local Experts, *Neural Computation* 3 (1991) 79–87.
18. Jordan, M. I., and Jacobs, R. A.: Hierarchical Mixtures of Experts and the EM Algorithm, *Neural Computation* 6 (1994) 181–214.
19. Kittler, J., Hatef, M., Duin, R. P. W., and Matas, J.: On Combining Classifiers, *IEEE Trans. on Pattern Analysis and Machine Intelligence* 20 (1998) 226–239.
20. Kuncheva, L. I., and Jain, L. C.: Designing Classifier Fusion Systems by Genetic Algorithms, *IEEE Trans. on Evolutionary Computation* 4 (2000) 327–336.
21. Lam, L., and Suen, C. Y.: Optimal Combinations of Pattern Classifiers, *Pattern Recognition Letters* 16 (1995) 945–954.
22. Langdon, W.: A Hybrid Genetic Programming Neural Network Classifier for Use in Drug Discovery, *Proc. of 2nd International Conference on Hybrid Intelligence Systems* (2002) 6 (Prenary Presentation).
23. Quinlan, J. R.: *C4.5: Programs for Machine Learning*, Morgan Kaufmann, San Mateo (1993).

Voronoi Diagrams Based Function Identification

Carlos Kavka¹ and Marc Schoenauer²

¹ LIDIC, Departamento de Informática, Universidad Nacional de San Luis
D5700HHW, San Luis, Argentina
`ckavka@unsl.edu.ar`

² Projet Fractales, INRIA Rocquencourt
BP 105, 78153 Le Chesnay Cedex, France
`Marc.Schoenauer@inria.fr`

Abstract. Evolutionary algorithms have been applied to function identification problems with great success. This paper presents an approach in which the individuals represent a partition of the input space in Voronoi regions together with a set of local functions associated to each one of these regions. In this way, the solution corresponds to a combination of local functions over a spatial structure topologically represented by a Voronoi diagram. Experiments show that the evolutionary algorithm can successfully evolve both the partition of the input space and the parameters of the local functions in simple problems.

1 Introduction

The objective when dealing with a function identification problem is to find an approximation that matches as closely as possible an unknown function defined in a certain domain. There are methods that can be used to optimize the parameters of the unknown function given a model, and also methods that can get both the model and the parameters at the same time. A well known example is the least squares method (LSM), that can compute the coefficients of a linear combination of base functions. Also non linear regression methods do exist, but they tend to be very time consuming when compared with the linear approaches. Neural networks have also been used for function approximation [7]. A neural network can be considered as a model of connected units, where usually the unknown parameters are the so called weights. The usual training algorithms can be used to obtain the weights, but there are also methods that can obtain both the connection pattern and the weights.

Evolutionary algorithms have been shown very effective in function identification problems in a wide range of domains [3] [1]. The Genetic Programming approach [8] uses a tree structure to represent an executable object (model) that can be a function – and has been successful addressing regression problems. Classifier systems have also been used for function approximation. In the XCFS classifier system [11], the value of the dependent variable is considered as a payoff to be learned, given the values of the independent variables.

But the choice of a method to solve a given function identification problem also depends on the available data: when *examples* of input/output patterns of

the unknown function are given, function identification, also then called *data fitting*, can be addressed by standard deterministic numerical methods. This is not the case, however, for the so-called *inverse problems*: In control for instance, candidate controllers cannot be specified just by a set of samples of values of the independent and dependent variables; but a good simulation of the plant is usually available, and allows one to evaluate candidate controllers. Even worse, it is usual that some regions of the space are oversampled, while others are undersampled.

This work is part of a series of investigations regarding the usefulness of a particular representation and operators defined to solve the inverse approximation problem with evolutionary algorithms. The objective is to find both a partition of the domain into a set of regions and a set of functions to be applied in each region. In this way, instead of the solution being defined by a single *global* function, it is defined by a set of *local* functions. The partition of the domain is defined in terms of Voronoi regions, a geometric structure that proved to be useful in interpolation [4] [2] and evolution applied to structural mechanics problems [10]. No restrictions are defined for the local functions, but they are expected to be simple functions. Two possible rules for their combination are proposed in order to get a solution in the whole domain.

The paper is organized as follows: section 2 presents the details on the domain partition based on Voronoi regions, section 3 presents details on local approximators and the way in which they are combined, section 4 introduces the representation and the evolutionary operators, section 5 some numerical results on simple problems and section 6 presents the conclusions and the current lines of research.

2 Domain Partition

The domain partition strategy is based on Voronoi diagrams. A Voronoi diagram induces a subdivision of the space based on a set of points called *sites*. An important property is that a number of operations can be executed on its topological structure just by operating with the sites.

Formally [5], a Voronoi diagram of a set of n points P is the subdivision of the plane into n cells, one for each site in P , with the property that a point q lies in the cell corresponding to a site p_i if and only if the distance between q and p_i is smaller than the distance between q and p_j for each $p_j \in P$ with $j \neq i$. In other words, the cell of the site p_i contains all the points in the plane for which p_i is the closest site. The Voronoi diagram of P will be denoted by $\text{Vor}(P)$ and the cell (or region) that corresponds to p_i by $\text{Vor}(p_i)$. Figure 1 illustrates an example of a Voronoi diagram in $\mathbb{R}^2$. The definition can be straightforward extended to $\mathbb{R}^n$, with $n \geq 2$.

A related concept that will be used in the paper is the so called Delaunay triangulation. A triangulation [5] of a set of points P is defined as the maximal planar subdivision whose vertex set is P . A maximal planar subdivision S is a subdivision such that no edge connecting two vertices can be added to S without

destroying its planarity. In other words, any edge that is not in S intersects one of the existing edges. A triangulation $\mathcal{T}$ of a set of points P is a Delaunay triangulation if and only if the circumcircle of any triangle in $\mathcal{T}$ does not contain a point of P in its interior. A circumcircle of a triangle is defined as the circle that goes through its three summits. Figure 1 illustrates an example of a Delaunay triangulation in $\mathbb{R}^2$.

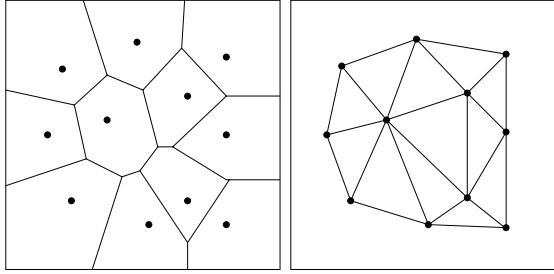


Fig. 1. An example of a Voronoi diagram (left) and a Delaunay triangulation (right) for a set of points in $\mathbb{R}^2$

3 Approximation

A solution to the function identification problem is defined by a partition of the domain in Voronoi regions together with a set of local functions. Two ways of recombining local functions are proposed: a non continuous combination, where each function defines the approximation in its own region, and a continuous combination, where the local functions are adapted in order to get a continuous approximation in the complete domain.

3.1 The Non-continuous Combination

Given a set of points $P = \{p_1, p_2, \dots, p_n\}$ and a set of continuous local functions $F = \{f_1, f_2, \dots, f_n\}$, each f_i being defined on $Vor(p_i)$, the value of the approximation $\mathcal{F}$ is defined as follows:

$$\mathcal{F}(x) = f_i(x) \quad \text{if } x \in Vor(p_i). \quad (1)$$

The value of the approximation $\mathcal{F}$ is computed as the value of the local function f_i associated to the Voronoi region to which the input value x belongs. The definition of $\mathcal{F}$ presents the following properties:

- It is defined on every point in the domain, provided that the local functions are defined, since the Voronoi diagram induces a partition of the complete domain.
- It is a (possibly) discontinuous function, presenting discontinuities at the Voronoi region boundaries.

3.2 The Continuous Combination

Given a set of points $P = \{p_1, p_2, \dots, p_n\}$, a set of continuous local functions $F = \{f_1, f_2, \dots, f_n\}$ and a set of real values $V = \{v_1, v_2, \dots, v_n\}$, the value of the approximation $\mathcal{F}$ is defined as follows:

$$\mathcal{F}(x) = f_i(x) * \mathcal{D}(x) + \mathcal{L}(x) \quad \text{if } x \in \text{Vor}(p_i). \quad (2)$$

The value of the approximation $\mathcal{F}$ is computed as the value of the local function f_i associated to the Voronoi region to which the input value x belongs to, scaled by a distance factor $\mathcal{D}$, with the addition of the value of a global function $\mathcal{L}$ evaluated on x .

The distance factor $\mathcal{D}$ is defined as follows:

$$\mathcal{D}(x) = \begin{cases} \phi\left(\frac{d(x, \text{boundary}(x))}{d(p_i, \text{boundary}(x))}\right) & \text{if } \text{boundary}(x) \text{ is defined} \\ 1 & \text{if } \text{boundary}(x) \text{ is not defined} \end{cases} \quad (3)$$

where p_i is the center of the Voronoi region to which x belongs to, $\text{boundary}(x)$ the point that corresponds to the Voronoi regions boundary, $d(a, b)$ the Euclidean distance between the points a and b , and ϕ a continuous function defined in the range $[0..1]$ such that $\phi(0) = 0$ and $\phi(1) = 1$. $\text{boundary}(x)$ is defined as the intersection of the line joining p_i and x and the boundaries of $\text{Vor}(p_i)$, if that intersection lies within the domain of the definition of the problem.

The global function $\mathcal{L}$ is defined based on the Delaunay triangulation as follows:

$$\mathcal{L}(x) = \sum_{j=1}^{d+1} v_j * l_j. \quad (4)$$

where d is the dimension of the input space, $l_1, \dots, l_{d+1}$ are the barycentric coordinate of x in simplex T (triangle the case if $d = 2$), and $v_1, \dots, v_{d+1}$ are given values, each one associated with the corresponding Voronoi site (see section 4.1). The definition of $\mathcal{F}$ presents the following properties:

- It is defined on every point in the domain, provided that the local functions are defined, since the Voronoi diagram induces a partition for the complete domain, and $\mathcal{D}$ and $\mathcal{L}$ are continuous functions.
- The value of the first term is the local approximator in the center of the region and 0 in the boundaries, since the distance factor $\mathcal{D}$ produces a value in the range $[0..1]$, with the value 0 in the boundaries of the Voronoi region and the value 1 in the center. The function ϕ controls the shape of the change of $\mathcal{D}$ between 0 and 1. It can be linear, quadratic, exponential, ... Ultimately, different functions could be attached to different Voronoi sites. In the external area of unbounded Voronoi regions the value of $\mathcal{D}$ is 1.
- The function $\mathcal{L}$ is global since it depends on the values v_i associated to the sites and their positions in the space. It is a piecewise linear continuous function. The vertices of the triangles defined by the Delaunay triangulation are the sites from P . Given a point x that belongs to the triangle T in the

Delaunay triangulation, the value of the global function $\mathcal{L}$ is computed by performing an interpolation of the values associated to the vertices by using the barycentric coordinates of the point x in the triangle. The barycentric coordinates of a point x are the local coordinates of x in T representing the ratios of the areas of the sub-triangles formed by x and the sides of the triangle. It corresponds to what is called a triangular element in the Finite Elements Method terminology [12]. As an example, there are three barycentric coordinates for a point in $\mathbb{R}^2$: l_1, l_2, l_3 , with $\sum l_i = 1$. The value of one of each of the barycentric coordinates is 1 in a vertex while the value of the others barycentric coordinates is 0. Figure 2 presents the value of the three barycentric coordinates for a triangle in $\mathbb{R}^2$, together with an example of the global function $\mathcal{L}$ evaluated on the same triangle. Note that even if

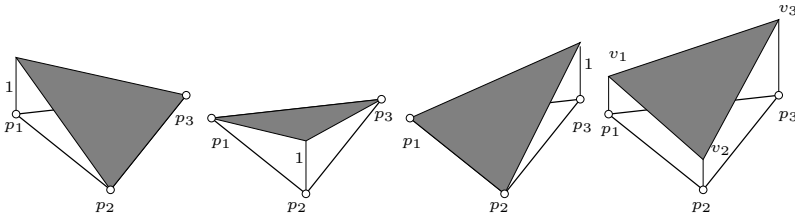


Fig. 2. Barycentric coordinates in a triangle in $\mathbb{R}^2$ (from left to right): barycentric coordinate l_1 , barycentric coordinate l_2 , barycentric coordinate l_3 , the global function $\mathcal{L}$ in the triangle T

it is true that every point in the domain belongs to a Voronoi region, it is not the case with the Delaunay triangulation. Some points that belong to unbounded Voronoi regions do not belong to any triangle defined by the Delaunay triangulation (see figure 1). In order to define $\mathcal{L}$ on these points, a large triangle covering the whole domain and containing all points in P is defined with values 0 associated to its vertices [5].

4 The Evolutionary Algorithm

This section introduces the representation used for the individuals and the definition of the evolutionary operators.

4.1 The Representation

Each individual has to represent a complete solution, or in other words, a complete approximation $\mathcal{F}$. A convenient representation is a list of variable length of Voronoi sites, local approximator parameters and global function values (if used), represented as real values.

In order to formalize the definition, let us define a *local vector* as the vector that contains all parameters associated to a Voronoi region: the coordinates that define the site, the parameters of the local approximator and the value of the global function (if applicable). A local vector lv_i of an individual *ind* with non continuous approximation is defined as the vector:

$$lv_i = [c_1, \dots, c_d, par_1, par_2, \dots, par_m]$$

or as the following vector with the continuous approach:

$$lv_i = [c_1, \dots, c_d, v, par_1, par_2, \dots, par_m]$$

where the site coordinates are $(c_1, \dots, c_d)$, d the dimension of the domain, v the value of the global function on the site and $par_1, \dots, par_m$ the m parameters that defines the local approximator.

An individual *ind* can be represented as a vector of variable length that is built from the concatenation of the local vectors that represent the local approximators in the solution:

$$ind = lv_1 + lv_2 + \dots + lv_n$$

As an example, an individual in $\mathbb{R}$ with 3 local approximators defined with two parameters (a and b) each one is:

with non continuous approximation: $\underbrace{[p_1, a_1, b_1]}_{lv_1}, \underbrace{[p_2, a_2, b_2]}_{lv_2}, \underbrace{[p_3, a_3, b_3]}_{lv_3}$

with continuous approximation: $\underbrace{[p_1, v_1, a_1, b_1]}_{lv_1}, \underbrace{[p_2, v_2, a_2, b_2]}_{lv_2}, \underbrace{[p_3, v_3, a_3, b_3]}_{lv_3}$

The representation proposed does not impose restrictions on the shape of the local approximators. In this work, we have used linear, quadratic and RBF approximators. They were defined in $\mathbb{R}^2$ respectively as $f(x) = ax+b$, $f(x) = ax^2+bx+c$ and $f(x) = \exp(-(x-p)^2/a^2)b+c$, where a , b and c are the parameters, and p the center of the corresponding Voronoi region.

4.2 The Operators

Three mutation operators and one crossover operator have been specifically defined for individuals that represent a Voronoi diagram. They are described below:

Voronoi crossover: This crossover operator is based on the crossover defined in [10] and exchanges the local vectors of the individuals by using geometric properties of the Voronoi diagrams. A random hyperplane h of dimension $d-1$ is randomly defined. The first child receives the local vectors from the first parent that lie on the left of h , and the local vectors from the second parent that lie on the right of h . The second child receives the remaining local vectors. Figure 3 presents an example of the application of this operator

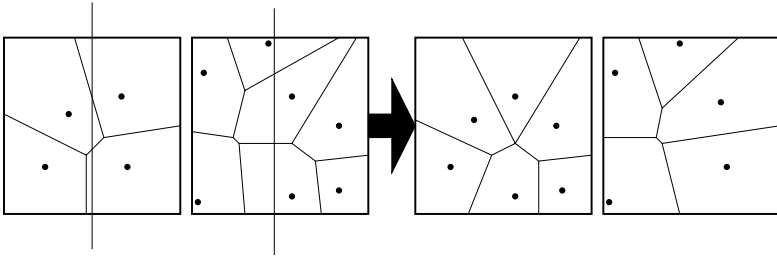


Fig. 3. An example of the application of the Voronoi crossover in $\mathbb{R}^2$

in $\mathbb{R}^2$. Formally, given two individuals ind^1 and ind^2 with n and m local vectors respectively:

$$ind^1 = [lv_1^1, \dots, lv_n^1] \quad ind^2 = [lv_1^2, \dots, lv_m^2]$$

The two children $child^1$ and $child^2$ are:

$$child^1 = \{lv_i^1, lv_j^2 / lv_i^1 \in Left(h), lv_j^2 \in Right(h)\}$$

$$child^2 = \{lv_i^1, lv_j^2 / lv_i^1 \in Right(h), lv_j^2 \in Left(h)\}$$

where h is a random hyperplane of dimension $d-1$, a local vector belongs to $Left(h)$ if its site is on the left of h and a local vector belongs to $Right(h)$ if its site is on the right of h . The concept of being on the right or on the left of an hyperplane can be found in computational geometry textbooks, like for example [5].

Mutation: This operator modifies the coordinates of the Voronoi site, the parameters of the approximator or the global function value (if used) by updating the values as follows:

$$x_i(t) = \begin{cases} x_i(t) + \Delta(t, ub_j - x_i(t)) & u < 0.5 \\ x_i(t) - \Delta(t, x_i(t) - lb_i) & u \geq 0.5 \end{cases} \quad (5)$$

where $x_i(t)$ is the i -th parameter at time t , u is a uniformly generated random number in $[0 : 1]$, lb_i and ub_i are respectively the lower bound and the upper bound for the parameter x_i , and Δ is a function defined as follows:

$$\Delta(t, y) = yv(1 - t/T)^b. \quad (6)$$

where $v \in [0 : 1]$ is a random number obtained from a gaussian distribution with mean 0 and standard deviation 0.3, T the maximum number of generations and b a parameter that controls the degree of nonuniformity. This operator is used to search the space uniformly in the first generations and very locally at the end, in order to fine tune the values in the individual. It is based on the nonuniform mutation operator defined by Michalewicz [9].

Add mutation: This mutation operator is used to add a new random local vector to the individual.

Del mutation: This mutation operator is used to remove a local vector from an individual.

5 Numerical Results

The objective of this section is to evaluate the representation and the operators described in the previous sections, when applied to simple function identification problems.

The first experiment consists in the approximation of the following function in $\mathbb{R}$, with linear, quadratic and RBF approximators, using both the non continuous and the continuous approach:

$$f(x) = \begin{cases} \frac{\sin(3\pi x)^2}{x} & -1 \leq x < 0 \\ \frac{1}{x + \exp(-x)} - 1 & 0 \leq x \leq 1 \end{cases} \quad (7)$$

The algorithm implemented is a classical generational genetic algorithm with population size 100 and tournament selection (100 parents give birth to 100 offspring that replace the 100 parents). For the experimental test, the crossover rate set to 0.7, mutation rate to 0.1, mutation b parameter to 0.2, adding mutation rate equal to 0.1, delete mutation rate equal to 0.1, the maximum number of function evaluations to 50 000. Individuals are evaluated by computing the error on the approximation of a set of points, selected randomly for each individual on each evaluation. It means that the individuals are not evaluated with the same dataset – as would happen in inverse problems. The function ϕ (see equation (3)) is the identity. The results are summarized in table 1, and corresponds to 10 independent runs for each set of parameters. The errors are specified as a percentage of the output range, divided by 100. The error is computed by evaluating the approximation obtained in 1000 points evenly selected from the whole domain, and comparing its value with the real value obtained from the target function. As a consequence of this fact, the fitness values are not the same as the errors presented in this table. The figure 4 presents examples of the kind of solutions obtained with the different kind of approximators. Each plot shows the function to be approximated and the approximation obtained. The boxes in the diagram correspond to the Voronoi regions in which the domain is partitioned.

The solutions obtained by all methods are comparable in quality. Better solutions can be found by increasing the limit on the number of function evaluations. The main difference is on the size of the approximators obtained. A continuous combination can produce solutions with less local approximators, maintaining the same quality in the approximation. The second experiment concerns the crossover operator. Since the representation allows the use of standard one point crossover, the objective of the second experiment is to compare the Voronoi crossover with the one point crossover. The one point crossover tends to generate very long individuals, or in other words, individuals with a big number of Voronoi regions. It was not possible to use just the one point crossover, without adding a regularization term [7] in order to penalize long individuals. The results are summarized in table 2, and corresponds to 10 independent runs for each set of parameters. The value of the regularization parameter corresponds to the factor used to weight the size of the individual against the error in order

Table 1. Solutions found by the evolutionary algorithm for the first experiment. The smallest (resp. largest) size values correspond to the size of the smallest (resp. largest) best individuals found in the set of runs

		approximator	continuity	best error	std. dev.	smallest	largest
linear	no	0.017882	0.00562	12	23		
quadratic	no	0.017015	0.00693	12	29		
RBF	no	0.013958	0.00561	11	53		
linear	yes	0.018051	0.00577	6	11		
quadratic	yes	0.011043	0.01041	5	13		
RBF	yes	0.011468	0.00353	4	16		

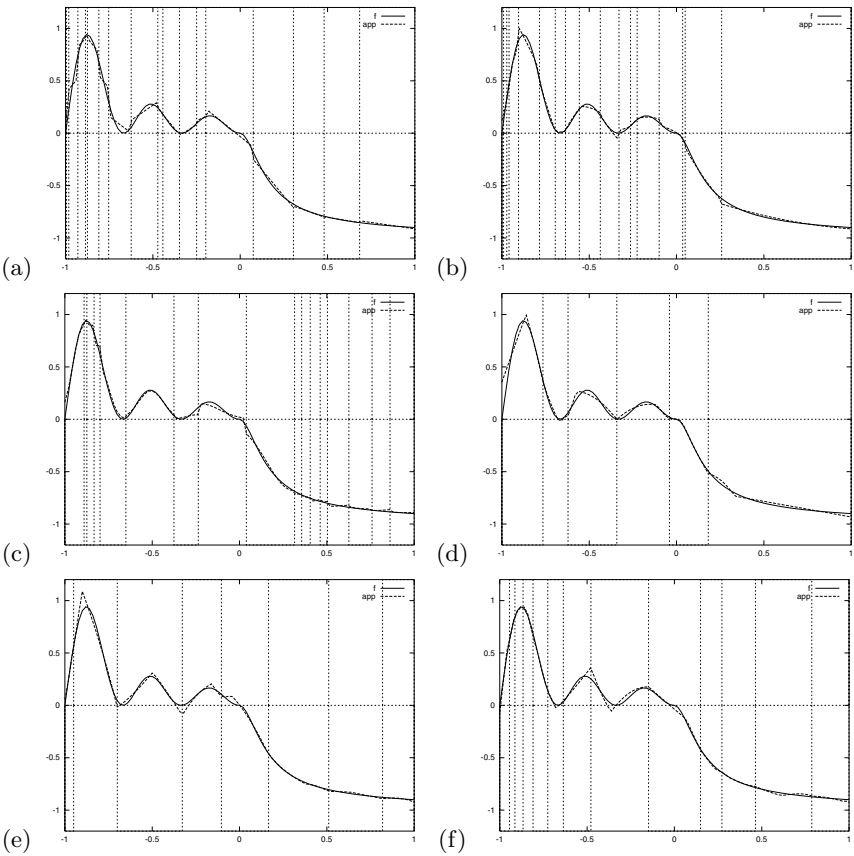


Fig. 4. Example of approximations obtained for the first function by using (a) linear, (b) quadratic and (c) RBF local approximators with no continuity, and using (d) linear, (e) quadratic and (f) RBF local approximators with continuity

Table 2. Solutions found for the second experiment

approximator	continuity	crossover	regularization	best error	smallest	largest
linear	yes	voronoi	no	0.018051	6	11
linear	yes	one point	no	-	-	-
linear	yes	one point	0.995	0.024664	23	41
linear	yes	one point	0.990	0.030236	11	29

to compute the fitness. The third experiment is intended to analyze the shape of the Voronoi partitions obtained in the approximation of the following functions in $\mathbb{R}^2$, by using linear approximators and continuous combination:

$$f(x, y) = x \exp(-x^2 - y^2) \qquad f(x, y) = \sin(\sqrt{x^2 + y^2}) . \tag{8}$$

The same parameters of the first experiment are used. The results are summarized in table 3, and correspond to 5 independent runs. The figure 5 presents the plot of the target functions, examples of the approximations obtained by the algorithm and their corresponding Voronoi partitions.

Table 3. Solutions found by the evolutionary for the third experiment

approximator	continuity	crossover	regularization	best error	std.dev.
linear	yes	voronoi	no	0.021192	0.01092
linear	yes	voronoi	no	0.028055	0.02411

6 Conclusions

This work presents preliminary results on the use of evolutionary computation and Voronoi diagrams in function identification. The proposed representation and the operators allow the algorithm to evolve both the partition of the input space and the local functions to be applied in each region. The partition of the input space is performed through Voronoi diagrams. Two ways to combine the local approximators are proposed. The representation of the partition does not depend on the type of local approximators used. In this work, linear, quadratic and RBF local approximators were used, but other kind of approximators can also be used.

Experiments on simple problems show that good approximations can be obtained with linear, quadratic and RBF local approximators. They also show that continuity in the combination of local functions can help to obtain approximations with smaller number of local approximators. The Voronoi crossover showed

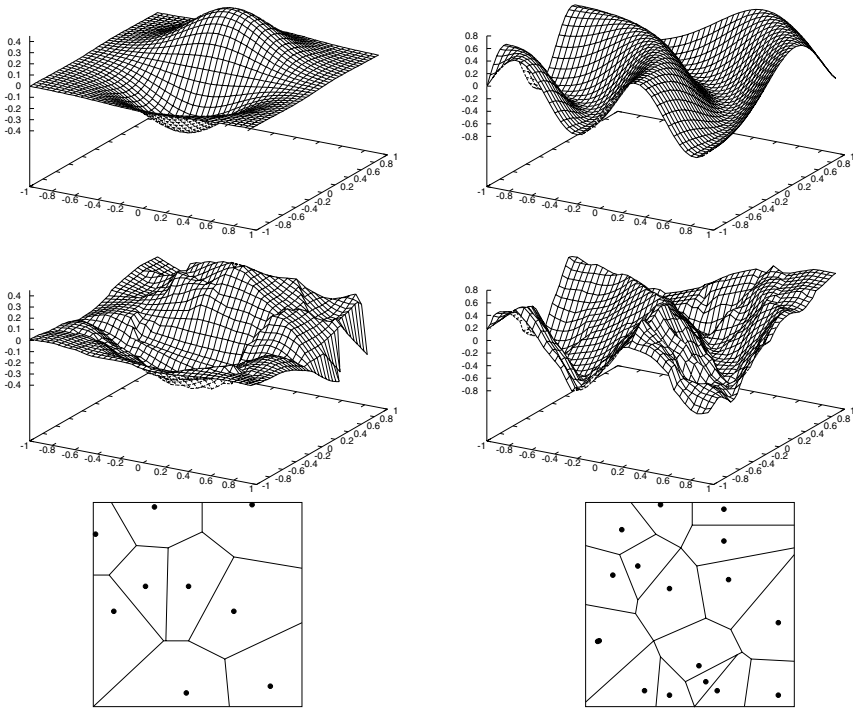


Fig. 5. Target functions, examples of approximations and Voronoi diagram partitions

superior performance when compared to standard one point crossover. In particular, it was not necessary to add a regularization term in order to avoid the solutions to be composed of a very large number of local approximators.

The evolution of individuals with RBF approximators produces an approximation $\mathcal{F}$ that is equivalent to an RBF neural network, since the computation performed by the local approximators correspond to what the nodes of the first layer do, and the combination strategy to the computation done at the output node. The usual training strategy of an RBF neural network consists in two steps [7]: a selection for the centers of the nodes in the first layer, and training in order to compute the weights that connect the two layers. Usually the first step is done through a non supervised technique (or by randomly selecting the centers), and the second step with supervised training. With the evolutionary algorithm, both steps are performed at the same time. An advantage of this approach is that it is not necessary to define in advance the number of centers (or regions) since the algorithm does the partition by itself.

It can be noted from the experiments that the evolutionary algorithm assigns more local approximators in areas that are more difficult to approximate. The concept of being difficult depends on the kind of local approximator. For example, a linear segment is easy to approximate for a linear local approximator but

more difficult for an RBF approximator. A representation that allows different kinds of local approximators to be evolved at the same time in different regions of the space is under analysis now.

Future experiments will include problems defined in terms of ordinary differential equations, like for example, problems from control, or models of some physical or chemical processes [6]. The main characteristic is the fact that these problems cannot be defined just in terms of a set of input-output patterns. In the experiments performed till now, even if it could have been possible to define the problems in terms of a fixed data set, it was preferred to follow this approach as much as possible, generating new random patterns for each evaluation of the individuals. It is expected that the local approximation properties of the method proposed can demonstrate itself to be useful in the identification of complex functions that do not pertain to data fitting.

References

1. Ahmed, M. and De Jong, K.: Function Approximator Design using Genetic Algorithms. Proceedings of the 1997 IEEE Int. Conference on Evolutionary Computation, Indianapolis, IN, pp. 519–523 (1997)
2. Anton, F., Mioc D. and Gold C.: Line Voronoi Diagram based Interpolation and Application to Digital Terrain Modeling. Proceedings of the 13th Canadian Conference on Computational Geometry, University of Waterloo, (2001)
3. Back, T. Fogel, D. and Michalewicz, Z.(Eds.): Handbook of Evolutionary Computation. IOP Publishing Ltd and Oxford University Press. (1997)
4. Boissonnat, J. and Cazals, F.: Smooth Surface Reconstruction via Natural Neighbour Interpolation of Distance Functions. Rapport de recherche de l'INRIA – Sophia Antipolis (2000)
5. de Berg, M., van Kreveld, M., Overmars, M., Schwarzkopf, O.: Computational Geometry, Algorithms and Applications. Second Edition. Springer-Verlag. (1998)
6. Fadda, A. and Schoenauer, M., Evolutionary Chromatographic Law Identification by Recurrent Neural Nets. In J. R. McDonnell and R. G. Reynolds and D. B. Fogel, Eds, Proc. 4th Annual Conference on Evolutionary Programming, pp 219–235. MIT Press, 1995.
7. Fiesler, E. and Beale R. (Eds.): Handbook of Neural Computation. Institute of Physics Publishing. (1997)
8. Koza, J.: Genetic Programming: On the Programming of Computers by means of Natural Evolution. MIT Press. (1992)
9. Michalewicz, Z.: Genetic Algorithms + Data Structures = Evolution Programs. Third, Revised and Extended Edition. Springer-Verlag Berlin Heidelberg. (1999)
10. Schoenauer, M., Jouve, F. and Kallel, L. Identification of Mechanical Inclusions. Dasgupta and Z. Michalewicz Ed. Identification of Mechanical Inclusions. Evolutionary Algorithms in Engineering Applications. (1997)
11. Wilson, S.: Classifiers that approximate functions. Natural Computing, 1(2–3), 211–234 (2002)
12. Zienkiewicz, O. C. and Taylor R. L.: Finite Element Method: Volume 1, The Basis. Fifth edition. Butterworth-Heinemann.(2000)

New Usage of SOM for Genetic Algorithms

Jung-Hwan Kim and Byung-Ro Moon

School of Computer Science and Engineering
Seoul National University
Shilim-dong, Kwanak-gu, Seoul, 151-742 Korea
{[@soar.snu.ac.kr](mailto:aram,moon)}

Abstract. Self-Organizing Map (SOM) is an unsupervised learning neural network and it is used for preserving the structural relationships in the data without prior knowledge. SOM has been applied in the study of complex problems such as vector quantizations, combinatorial optimization, and pattern recognition. This paper proposes a new usage of SOM as a tool for schema transformation hoping to achieve more efficient genetic process. Every offspring is transformed into an isomorphic neural network with more desirable shape for genetic search. This helps genes with strong epistasis to stay close together in the chromosome. Experimental results showed considerable improvement over previous results.

1 Introduction

There have been a great many genetic algorithms (GAs) for neural network (NN) optimization [9][11][18]. In order to represent solutions for NN optimization, most GAs used linear encodings following the convention of the GA community [11][15]. Mostly, every weight in an NN takes a position in a linear chromosome. However, linear encodings have limited capability in reflecting the geographic linkages of genes [1][7]. A number of 2D encodings were used to better reflect the geographic linkages of genes [1][4][7][12].

In this paper, we use a two-dimensional encoding for the genetic representation and employ 2D *geographic crossover* which demonstrated good performance for the neural network optimization problem [13] and the graph partitioning problem [12][16][17]. The weights of an NN can be represented by a 2D matrix and thus they are intrinsically suitable for 2D encoding.

A typical NN consists of input, output, and hidden layers. The neurons in the hidden layer enable the network to learn complex tasks by extracting progressively more meaningful features from the input patterns and to form the rules classifying the input patterns. Once an NN is represented by a two-dimensional encoding and the structure of the NN is fixed, the genotype of an NN depends on the order of neurons in the structure. All the hidden neurons are identical when only the connections are considered. They become different only after they start having weights in the connections. Some hidden neurons have stronger relation than between ordinary pairs of hidden neurons. However, the relationship among the hidden neurons is not revealed from neurons' placements. Since the

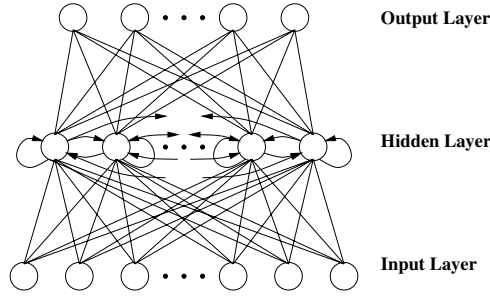


Fig. 1. The recurrent neural network architecture

indices of hidden nodes are assigned before they start having weights, there is no guarantee that strongly related neurons are assigned close indices to one another. This causes high-quality schemata not to survive well. In this paper, we transform each neural network to an isomorphic neural network with different indices of neurons and we aim to develop a better genetic algorithm by achieving a better representation of neural networks. We use self-organizing map (SOM) for the transformation.

Self-Organizing Map (SOM) is an unsupervised learning neural network and it is used for preserving the structural relationships in data without prior knowledge. SOM has been applied in the study of complex problems such as vector quantizations, combinatorial optimization, and pattern recognition. This paper proposes a new usage of SOM as a tool for schema transformation hoping to achieve more efficient genetic process.

The rest of this paper is organized as follows. In the next section, we describe the necessity of clustering strongly connected neurons and the property of SOM. In Section 3, we describe the transformation that enables useful building blocks to survive well. Section 4 shows our hybrid neuro-genetic framework for improving the performance of the neural network. We present our experimental results in Section 5 and state our conclusions in Section 6.

2 Preliminaries

2.1 Network Architecture

We use a recurrent neural network (RNN) architecture based on Elman's recurrent neural network [8]. It consists of two layers, excluding the input layer, as shown in Figure 1. Each hidden unit is connected to itself and also fully connected to all the other units. These connections are updated in the propagation phase at every Δt time interval. The network is usually trained by a backpropagation-based algorithm.

2.2 Topological Ordering Property of SOM

SOM is an effective software tool for the visualization of high-dimensional data. It converts complex, nonlinear statistical relationships among high-dimensional data items into simple geometric relationships on a low-dimensional display [14]. As it thereby compresses information while preserving the most important topological and metric relationships of the primary data items on the display, it may also be thought to produce some kind of abstractions. These two aspects, visualization and abstraction, can be utilized in a number of ways in complex tasks such as process analysis, machine perception, control, and communication. The data items are organized into a meaningful two-dimensional order in which similar items are close to one another. In this sense, SOM is a similarity graph and also a clustering diagram [14].

2.3 Gene Reordering

Each gene location has an explicit meaning in most genetic algorithms. This type encoding is called locus-based encoding. Many researchers have studied the gene reordering. Bagley [2] tried to change the gene ordering at random. Bui and Moon [3][6] tried to exploit the useful clustering information of graphs and showed that the performance of a GA can be improved by reordering genes in the graph bipartitioning problem with a linear encoding. Kim and Moon [13] proposed a reordering heuristic for neural-network representation. They believed that edges connecting units with strong relationships would have stronger patterns than a random set of edges and used the relative strengths of the connection weights to measure the relationships among genes. This approach showed considerable improvement of GA performance.

3 Transforming into Isomorphic Structures

Since the hidden neurons of the network are fully connected to one another, there are great many hidden node orderings with equivalent function. All these structures represent the same network but have different genotypes. Some of them are more advantageous than the others in genetic search. Figure 2 shows an example with matrix encoding. In the figure, the three neurons j , i , and $i + 1$ are assumed to have strong relationships. The positions of corresponding weights are marked with rectangles. The set of rectangles constructs a schema of perhaps high quality. Although the geographic crossover has the new-schema creation power for 2D encoding, if useful features on the encoding are not clustered (see the left weight matrix in Figure 2), corresponding schemata are not easy to survive in most crossovers. It is important that the given structure must be transformed to the functionally equivalent structure containing the useful clustered blocks. When we exchange the positions of neurons j and $i - 1$ in the left weight matrix, the schema is transformed to another one with better shape for survival (See the right weight matrix in Figure 2). We aim to transform high

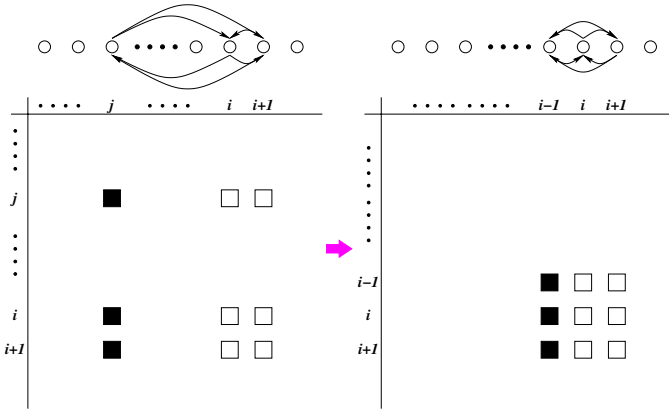


Fig. 2. Change of schema after transformation

quality schemata into better shapes with equivalent function. The more clusters of the high correlated neurons, the finer approximation of the complex system to be modeled can be eventually obtained. The idea takes advantage of the ability of SOM for clustering neurons.

3.1 SOM-Based Transformation

Each hidden neuron has a vector of weights and affects other neurons according to the weights. A large weight corresponds to a strong activation of another neuron. On the other hand, a significantly small negative weight corresponds to strong inhibition. If a neuron strongly activates or inhibits another neuron, we consider that these neurons are highly correlated. This information is utilized in the transformation. However, it is difficult to analyze the relative relationship among neurons and the spatial ordering of the corresponding neurons. In this work, we transform the topology of neural networks using SOM. This approach is motivated by a desire to better preserve the relationships of hidden neurons in the genetic process.

We describe the details of the SOM-based transformation in the following:

Input : an RNN with weights between neurons.

Output : a one-dimensional order of hidden neurons

$$\Pi = \langle \pi_1, \dots, \pi_p \rangle, \quad \pi_i \in \{1, 2, \dots, p\}.$$

1. The hidden layer is mapped to the competitive layer of SOM. That is, the number of hidden neurons is the same as the number of competitive neurons in SOM. The synaptic weights of the RNN are used as the input vectors for the SOM. Let the number of input neurons, hidden neurons, and output neurons of RNN be n , p , and q , respectively. Then the SOM has the $(n+p+q)$ input neurons and the p competitive neurons.

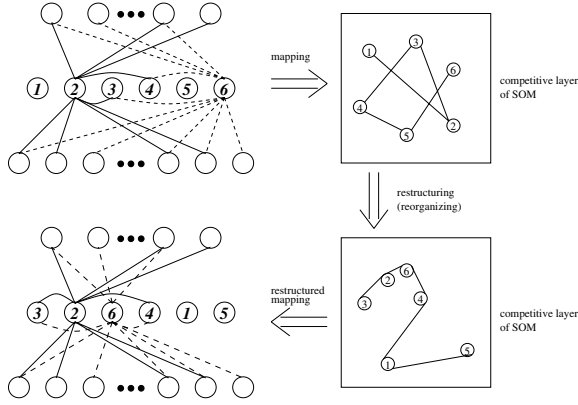


Fig. 3. Mapping between RNN and SOM

- SOM is trained with the synaptic weights related to the hidden neurons of the RNN. SOM produces a similarity graph reflecting the relationship among these synaptic weights. The number of input data pairs for SOM is p . That is, SOM trains the data pairs $\langle \mathbf{x}_j, d_j \rangle$ where $\mathbf{x}_j$ is an input vector for the SOM and d_j is the label corresponding to $\mathbf{x}_j$ as follows:

$$\begin{aligned} \mathbf{x}_j &= \mathbf{w}_j, & d_j &= i \quad (1 \leq i \leq p) \\ \mathbf{w}_j &= \langle w_{j1}^{(0)}, \dots, w_{jn}^{(0)}, w_{j1}^{(1)}, \dots, w_{jp}^{(1)}, w_{j1}^{(2)}, \dots, w_{jq}^{(2)} \rangle \end{aligned}$$

where

- $w_{jk}^{(0)}$: weight from input unit k to the hidden unit j of the RNN
- $w_{jk}^{(1)}$: weight from hidden unit k to the hidden unit j of the RNN
- $w_{jk}^{(2)}$: weight from hidden unit j to the output unit k of the RNN

- The SOM produces a linear order of the competitive neurons. The hidden neurons are rearranged according to the order produced by SOM. Sometimes several competitive neurons may have the same index. It occurs when the synaptic weights for different hidden neurons without being transformed represent a similar distribution. In this case, we assign the same index to multiple competitive neurons. This attempts to make the neural network obtain the effect of the weight-sharing [19].

Figure 3 shows an example SOM-based transformation. In the figure, the solid and dashed lines indicate the weights of high magnitudes. SOM trains the synaptic weights of the given network (the left-top neural network in Figure 3) and produces an ordering $\Pi = \langle 3, 2, 6, 4, 1, 5 \rangle$. The location of each node on the competitive layer of SOM reflects the relative relationship among hidden neurons. The high correlated neurons 2 and 6 locate distant to each other before the transformation; they are clustered after the transformation.

```

Create initial population of fixed size;
do {
    choose parent1 and parent2 from population;
    offspring = g2d_xover(parent1, parent2);
    mutation(offspring);
    backpropagation(offspring);
    SOM-transformation(offspring);
    if suited(offspring)
        then replace(population, offspring);
} until (stopping condition);
Return the best solution;

```

Fig. 4. The template of the hybrid GA

4 GA for Optimizing the Neural Network

4.1 The GA Structure

The template of the proposed genetic algorithm is shown in Figure 4. It is a typical steady-state hybrid genetic algorithm except for the SOM-based transformation. In the following, we describe the genetic framework.

- *Initialization*: The initial population is generated at random. We set the population size to be 50.
- *Selection*: In each iteration, two parents are selected according to probabilities that are proportional to their fitness values. The probability that the best solution is chosen was given four times that of the worst solution. This is a typical normalized roulette-wheel proportional selection.
- *Crossover*: The offspring is produced through geographic 2D crossover. The “*g2d_xover*” of Figure 4 indicates the geographic 2D crossover. The geographic 2D crossover is described in Section 4.3.
- *Mutation*: With a low probability, this offspring is then modified by a *mutation* operator as follows: $\omega_i = \omega_i + \lambda_{0.5}$ where ω_i is the i^{th} gene value and λ_γ returns a random value between $-\gamma$ and γ .
- *Local Optimization*: After an *offspring* is modified by a mutation operator, it is locally optimized by backpropagation. The backpropagation process helps the GA fine-tune around local optima. From another perspective, the GA provides diverse initial solutions to the backpropagation routine. After the backpropagation, the neural network has a mean-square error and a discrimination ratio on the training data. The fitness of the offspring is determined with these two values¹.
- *Transformation*: After local optimization, the genes in the offspring are re-ordered. This causes a structural change of the corresponding neural network

¹ $f = \varepsilon + \eta(100 - \mu)$,

where ε is the mean-square error, μ is the discrimination ratio, and η is a constant.

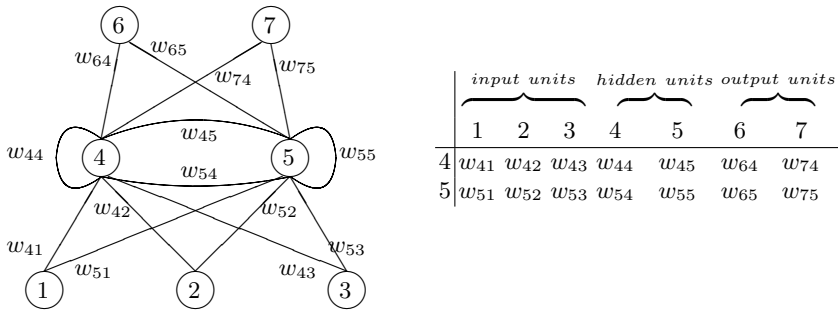


Fig. 5. Example of 2D encoding

without affecting the quality. The reorganizing approach was described in Section 3.

- *Replacement*: Then the offspring replaces a solution in the population by the following rule: the more similar parent to the offspring is replaced if the offspring is better; otherwise, the other parent is replaced if the offspring is better; if not again, the worst chromosome in the population is replaced. The rational behind this is to maintain the population diversity to the extent that not too much time is wasted [5].
- *Stopping Condition*: For stopping, we use a fixed number of generations.

4.2 Problem Encoding

Most GAs for neural network optimization encode a solution (a set of weights) with a linear string following the convention [9][11][15]. Instead of transforming into a linear string, we represent a solution by a weight matrix. In the matrix, each row corresponds to a hidden unit and each column corresponds to an input unit, a hidden unit, or an output unit. Figure 5 shows an example of such encoding. In this figure, the number within a circle (neuron) indicates the index of the neuron and w_{ij} represents the weight of the edge from neuron j to neuron i . The relationships among the edges in the network can be better reflected in this 2D representation, which was experimentally supported in [13]. In addition, we suspect that some neurons have stronger relationships with one another than with the others. We further modify the encoding by transforming the given network into an isomorphic network (See Section 3). In summary, a chromosome is represented by a $p \times (p + n + q)$ matrix with columns and rows rearranged, where p , n , and q are the numbers of hidden units, input units, and output units, respectively.

4.3 Geographic 2D Crossover

Two-dimensional encoding can preserve more geographical relationships among the genes [4][12][13]. However, when traditional straight-line-based cutting strategies are used, the power of new-schema creation is far below that of

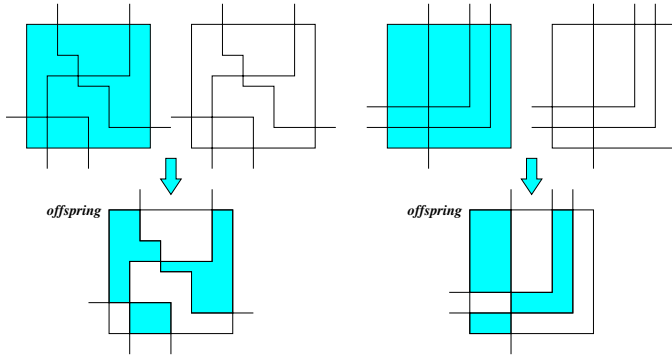


Fig. 6. Example of 2D geographic crossover

crossovers on linear encodings [12]. Geographic crossover was suggested to resolve this problem [12]. In the case of a 2D encoding, it chooses a number of monotonic lines, divides the chromosomal domain into two equivalence classes, and alternately copies the genes from the two parent chromosomes. Figure 6 shows two example geographic crossover operators. We used geographic crossover in this work. By combining two-dimensional representation, isomorphic transformation, and 2D geographic crossover, we are pursuing both reduced information loss in the stage of encoding and the power of new-schema creation.

5 Experimental Results

5.1 Database

We selected three well known datasets to measure the performance of the proposed approach. First, we used the MNIST handwritten character recognition database² for input patterns of 28×28 grids. The samples are evenly classified to ten digits. The digit images were written by 500 different writers. Recognition of handwritten character is an important task in the automated document analysis. This application is used to read postal addresses and bank checks, etc. We used 800 samples for training, 200 samples for the validation set, and 2000 samples for the test set. We used 45 input units and 20 hidden units by using a vertical density distribution and a horizontal density distribution as in [13].

Next, we experiment with Wisconsin breast cancer data (WBCD) and Cleveland heart disease data (CHDD) obtained from the UCI repository³. The WBCD contains 699 instances with 458 benign and 241 malignant cases. Each instance is described by a case number, nine attributes with integer values in the range [1, 10] and a binary class label. In the experiment using WBCD database, we used nine input units, nine hidden units, and two output units in the neural

² <http://yann.lecun.com/exdb/mnist>
³ <http://www.ics.uci.edu/~mllearn/MLRepository.html>

Table 1. Result for three testbeds

	BP	GA		
		Original	Heuristic	SOM
MNIST	90.28 (93.45)	94.08 (95.20)	96.52 (97.10)	96.85 (99.60)
WBCD	95.88 (96.37)	96.87 (97.21)	96.91 (97.43)	97.19 (97.79)
CHDD	80.56 (83.25)	82.22 (84.26)	82.75 (86.84)	82.82 (87.02)

network. We divided the data into 80 training samples, 20 validation samples, and 599 test samples. The CHDD contains 303 instances with 164 healthy instances. The rest are heart disease instances of various degrees of severity (4 classes). Each instance is described with 13 attributes. We used 80 samples for the training set, 20 samples for the validation set, and 203 samples for the test set. We used 13-21-4 network structure for CHDD data.

In these experiment, for robust check for the effect of the proposed strategy, we experimented with the data following the 5-fold *hold out method* cross-validation approach [10]. In such a situation, we divide the available set of data into 5 subsets. The model is trained on all the subsets except for one, and the validation error is measured by testing it on the subset left out. This procedure is repeated for a total of 5 trials, each time using a different subset for validation.

5.2 Experimental Results

Now the general effect of the proposed SOM-based transformation is illustrated with the above databases. The weights of a neural network were initialized at random between -0.5 and 0.5 . The learning rate was set to 0.8 and the momentum factor was used. The weights were tuned locally by backpropagation.

Table 1 shows the classification result. In the table, “BP” indicates the neural networks that are trained by backpropagation only. “GA” represents the hybrid neuro-genetic algorithm and is classified into three types. “Original” represents the version without any transformation in the genetic process. The other cases contain transformation in the genetic framework. “Heuristic” indicates the version with a heuristic transformation by [13]. “SOM” represents the version with SOM-based transformation. The two values in each experiment show the mean and the best recognition result, respectively, from 500 trials. The hybrid approach showed improvement over the non-hybrid approach (BP) and the transformation approaches turned out to be useful. The SOM-based transformation showed considerable improvement over the heuristic-based transformation.

6 Concluding Remarks

We devised a structural transformation algorithm for neuro-genetic hybrids to effectively reflect geographic correlations or relative contributions among neurons

in the genetic search. The topological ordering property of SOM was used to cluster neurons with high correlations. The proposed SOM-based transformation dynamically alters the geographical shapes of fitness landscapes; consequently, the shapes of schemata are altered as well.

To the best of our knowledge, this is the first usage of SOM in the context of genetic search. We believe that there is room for further improvement in this direction. Candidates of future studies include the enhancement of SOM-based transformation and the design of other types of transformation.

Acknowledgments. This research was supported in part by KOSEF through Statistical Research Center for Complex Systems at Seoul National University and Brain Korea 21 Project. The RIACT at Seoul National University provided research facilities for this study.

References

1. C. A. Anderson, K. F. Jones, and J. Ryan. A two-dimensional genetic algorithm for the Ising problem. *Complex Systems*, 5:327–333, 1991.
2. J. Bagley. *The Behavior of Adaptive Systems Which Employ Genetic and Correlation Algorithms*. PhD thesis, University of Michigan, Ann Arbor, MI, 1967.
3. T. N. Bui and B. R. Moon. Hyperplane synthesis for genetic algorithms. In *International Conference on Genetic Algorithms*, pages 102–109, 1993.
4. T. N. Bui and B. R. Moon. On multi-dimensional encoding/crossover. In *International Conference on Genetic Algorithms*, pages 49–55, 1995.
5. T. N. Bui and B. R. Moon. Genetic algorithm and graph partitioning. *IEEE Trans. on Computers*, 45(7):841–855, 1996.
6. T. N. Bui and B. R. Moon. GRCA: A hybrid genetic algorithm for circuit ratio-cut partitioning. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 17(3):193–204, 1998.
7. J. P. Cohoon and W. Paris. Genetic placement. In *IEEE International Conference on Computer-Aided Design*, pages 422–425, 1986.
8. J. L. Elman. Finding structure in time. *Cognitive Science*, 14:179–211, 1990.
9. A. Grauel and F. Berk. Mapping of dynamical systems by recurrent neural networks in an evolutionary algorithm approach. In *European Congress on Intelligent Techniques and Soft Computing*, volume 1, pages 470–476, 1998.
10. S. Haykin. *Neural Networks: A Comprehensive Foundation*. Prentice Hall, 1999.
11. R. Jeff and V. B. Ciesielski. An evolutionary approach to training feed-forward and recurrent neural networks. In *International Conference on Knowledge-Based Intelligent Electronics Systems*, pages 596–602, 1998.
12. A. B. Kahng and B. R. Moon. Toward more powerful recombinations. In *International Conference on Genetic Algorithms*, pages 96–103, 1995.
13. J. H. Kim and B. R. Moon. Neuron reordering for better neuro-genetic hybrids. In *Genetic and Evolutionary Computation Conference*, pages 407–414, New York, 9–13 July 2002. Morgan Kaufmann Publishers.
14. T. Kohonen, S. Kaski, K. Lagus, J. Salojärvi, J. Honkela, V. Paatero, and A. Saarela. Self organization of a massive document collection. *IEEE Transactions on Neural Networks*, 11(3):574–585, 2000.

15. C. T. Lin and C. P. Jou. Controlling chaos by GA-based reinforcement learning neural network. *IEEE Transactions on Neural Networks*, 10(4):846–869, 1999.
16. B. R. Moon and H. N. Kim. Effective genetic encoding with a two-dimensional embedding heuristic. *International Journal of Knowledge-Based Intelligent Engineering Systems*, 3(2):113–120, 1999.
17. B. R. Moon, Y. S. Lee, and C. K. Kim. GEORG: VLSI circuit partitioner with a new genetic algorithm framework. *Journal of Intelligent Manufacturing*, 9(5):401–412, 1998.
18. V. Patrdis, E. Paterakis, and A. Kehagias. A hybrid neural-genetic multimodel parameter estimation algorithm. *IEEE Transactions on Neural Networks*, 9(5):862–876, 1998.
19. D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. In D. E. Rumelhart and J. L. McClelland, editors, *Parallel Distributed Processing*, volume 1, chapter 8. MIT Press, Cambridge, MA, 1986.

Problem-Independent Schema Synthesis for Genetic Algorithms

Yong-Hyuk Kim, Yung-Keun Kwon, and Byung-Ro Moon

School of Computer Science & Engineering, Seoul National University
Shilim-dong, Kwanak-gu, Seoul, 151-742 Korea
{yhdfly, kwon, moon}@soar.snu.ac.kr

Abstract. As a preprocessing for genetic algorithms, static reordering helps genetic algorithms effectively create and preserve high-quality schemata, and consequently improves the performance of genetic algorithms. In this paper, we propose a static reordering method independent of problem-specific knowledge. One of the novel features of our reordering method is that it is applicable to any problem with no information about the problem. The proposed method constructs a weighted complete graph from the gene distances calculated from solutions with relatively high fitnesses, transforms them into a gene-interaction graph, and finds a gene rearrangement. Extensive experimental results showed significant improvement for a number of applications.

1 Introduction

By the schema theorem, Holland showed that highly fit schemata of short defining lengths and low orders have high probabilities of survival in the traditional genetic framework [19]. High-quality schemata with the above features are called *building blocks*. Building blocks are gene groups with high contribution to the fitnesses that have mutually strong interactions. The performance of a genetic algorithm highly depends on the survival environment and reproducibility of building blocks.

The survival probability of a gene group through crossovers is strongly affected by the positions of genes in the chromosome. Schemata consisting of widely scattered specific positions have poor survival probabilities through crossovers due to their long defining lengths. Thus, the strategy of locating genes significantly affects the performance of genetic algorithms. Inversion is a genetic operator devised for changing the loci of genes dynamically [3]. The efforts to exploit the loci of genes dynamically are called *linkage learning* [18]. Messy genetic algorithm is an example that implicitly pursues dynamic gene repositioning [16].

It has been observed that the performance of genetic algorithms on problems with locus-based encoding can be improved by statically reordering the indices of the genes. The technique of static reordering for genetic algorithms was first suggested in [6] [10], whose basic idea is to reassign the loci of genes in chromosomal representation to help genetic algorithms effectively preserve good schemata. A good reordering also leads to better creation of high-quality schemata than in

the original ordering. A number of studies on static reordering of gene positions in locus-based encodings showed performance improvement [6] [8] [10] [29].

However, previous reorderings depend on the specific information of their application [6] [8] [10]. Hence, a new heuristic has to be derived for the reordering of each new problem. In this paper, we describe a static reordering method which is free from problem-specific knowledge. The method requires locus-based encodings for chromosomal representation. We perform experiments on three representative combinatorial optimization problems that are NP-hard [15]: graph bisection, linear arrangement, and traveling salesman problem. Our experiments showed notable improvement when compared against the cases without reordering. In this paper, we use rearrangement, reordering, and preprocessing interchangeably.

The remainder of this paper is organized as follows. In Section 2, we summarize three testbed problems. The genetic framework that we used in this work is described in Section 3. In Section 4, we describe the problem-independent schema preprocessing for general purpose. We present experimental results in Section 5. Finally, we make our conclusions in Section 6.

2 Preliminaries

2.1 Graph Bisection

Let $G = (V, E)$ be an unweighted undirected graph, where V is the set of n vertices and E is the set of e edges. A bisection $\{C_1, C_2\}$ of the graph G satisfies $C_1, C_2 \subset V$, $C_1 \cup C_2 = V$, $C_1 \cap C_2 = \phi$, and $||C_1| - |C_2|| \leq 1$. The cut size of $\{C_1, C_2\}$ is $|\{(v, w) \in E : v \in C_1, w \in C_2\}|$. The graph bisection problem is the problem of finding a bisection with the minimum cut size. The problem has been extensively studied in the past [10] [25] [4] [23]. It is known to be NP-hard [15].

2.2 Linear Arrangement

Let $G = (V, E)$ be an unweighted undirected graph. The linear arrangement problem is the problem of finding a permutation $\sigma : V \rightarrow V$ of vertices with the minimum value of $\sum_{(u,v) \in E} |\sigma(u) - \sigma(v)|$. There have been a number of studies for the problem [1] [12] [37]. It is also NP-hard [15].

2.3 Traveling Salesman Problem (TSP)

Let $G = (V, E)$ be a complete graph with weights on the edges. A Hamiltonian cycle of G is a cycle that visits every vertex of the graph exactly once. The traveling salesman problem (TSP) is the problem of finding a Hamiltonian cycle with the minimum weight. TSP is well known to be NP-hard [15]. It has been extensively studied in the past due to its wide applications as well as for its complexity. Genetic algorithms have been applied to TSP with varying degrees of success [24] [32] [21] [38].

```

Preprocess;
Create an initial population;
repeat {
    choose parent1 and parent2 from population;
    offspring = crossover(parent1, parent2);
    local-improvement(offspring);
    replacement(population, offspring);
} until (stopping condition);
return the best solution;

```

Fig. 1. The framework of our hybrid genetic algorithm

3 A Hybrid Genetic Algorithm

A hybrid genetic algorithm is a genetic algorithm (GA) combined with a local improvement heuristic. Some people call it a memetic genetic algorithm [34] [11] [27] [28] [26] [5]. The general framework of hybrid steady-state genetic algorithm is used in our GA as shown in Figure 1. In the following, we describe each part of the GA that we used for this work.

- *Locus-based encoding*: Each solution in the population is represented by a chromosome. A binary encoding is used for the graph bisection problem. A gene has a value ‘0’ or ‘1’ depending on the side that the corresponding vertex belongs to. We use a permutation encoding for the linear arrangement problem. Each gene corresponds to a vertex in the graph and its value means the position in the arrangement. We also use a permutation encoding for the TSP. A gene corresponding a vertex v represents another vertex following vertex v in the Hamiltonian cycle. These encodings, where each gene location has an explicit meaning, are called locus-based encoding. It is necessary to use the locus-based encoding since the preprocessing heuristic presented in Section 4 is applicable only to locus-based encodings.
- *Selection and crossover*: To select two parents, we use a proportional selection scheme where the probability for the best solution to be chosen is four times higher than that for the worst solution. A crossover operator creates a new offspring by combining parts of parents. In the graph bisection problem, we use five-point crossover. After the crossover, an offspring may not satisfy the balance. It selects a random point on the chromosome and changes the required number of 1’s to 0’s (or 0’s to 1’s) from that point on. In the permutation encoding, we use the partially matched crossover [17]. There is no duplicated gene value in the offspring and it need not be repaired in case of the linear arrangement problem. However, since it may consist of more than one mutually disconnected subcycle, it may not be a proper Hamiltonian cycle in case of TSP. To resolve this problem, we used the repair algorithm introduced in [8].
- *Local improvement*: Hybrid genetic algorithms have been considered natural in solving a difficult problem to get desirable performance since genetic

1. Generate M solutions with relatively high fitness;
2. Compute distance for each gene pair;
3. Make a gene-interaction graph;
4. Find a gene arrangement;

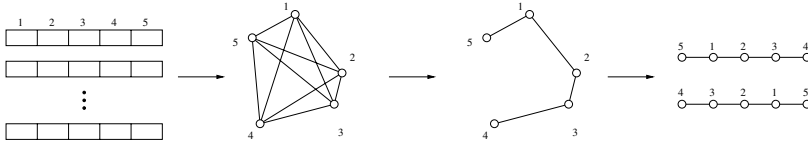


Fig. 2. The structure of problem-independent schema preprocessing

algorithms are not so good at fine tuning near local optima. In this study, we use one of the most basic local improvement heuristic, 2-Opt, which has 2-exchange as its neighbor structure. It is applied to the offspring after crossover in the GA.

- *Replacement and stop condition:* After generating an offspring, the GA replaces the worse of the two parents with the offspring. It is called *preselection* replacement. It stops after a fixed number of generations.

4 Problem-Independent Gene Rearrangement

As mentioned in Section 3, we use locus-based encodings for GA and rearrange the genes. Figure 2 shows the framework of the proposed schema preprocessing. It does not depend on any problem-specific knowledge.

- *Generating high quality solutions:* First, it generates M solutions with relatively high fitness. In this study, we generated 100 solutions using 2-Opt heuristics.
- *Computing the distance for each gene pair:* From the generated solution set, it computes the gene distance between each pair of genes according to its encoding type. Figure 3 describes how to measure the distance $D(g_i, g_j)$ between two genes g_i and g_j . In the figure, $f_l(g_i)$ means the value of gene g_i in the l^{th} solution. As explained in Section 3, binary encoding is used for graph bipartition problem. Sequential permutation encoding and cyclic permutation encoding are used for linear arrangement problem and TSP, respectively. Thus, we get a weighted complete graph with vertices and edges corresponding to genes and gene distances, respectively.
- *Making a gene-interaction graph:* It transforms the obtained weighted graph into unweighted sparse graph called *gene-interaction graph*. We assume that the edge weights in the weighted graph has the Gaussian distribution. To get the gene-interaction graph, it chooses only the heavy-weighted edges with 95% confidence level.

Binary encoding

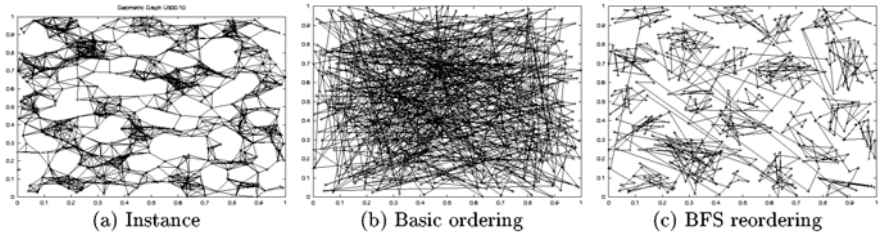
$$D(g_i, g_j) = \frac{1}{M} \sum_{l=1}^M I(f_l(g_i) \neq f_l(g_j))$$

Sequential permutation encoding

$$D(g_i, g_j) = \frac{1}{M} \sum_{l=1}^M |f_l(g_i) - f_l(g_j)|$$

Cyclic permutation encoding

$$D(g_i, g_j) = \frac{1}{M} \sum_{l=1}^M \operatorname{argmin}_k (g_i = f_l^k(g_j) \text{ or } g_j = f_l^k(g_i))$$

Fig. 3. Gene-distance measure in locus-based encoding**Fig. 4.** Reordering in graph bisection: instance U500.10

- *Finding a gene arrangement:* From the gene-interaction graph, it performs gene rearrangement. Given the set of genes $\{g_1, g_2, \dots, g_n\}$, a gene rearrangement $\{g_{\sigma(1)}, g_{\sigma(2)}, \dots, g_{\sigma(n)}\}$ is represented by a bijective map $\sigma : \{1, 2, \dots, n\} \rightarrow \{1, 2, \dots, n\}$. Gene v_i is the j^{th} gene in the gene rearrangement if $\sigma(j) = i$. In general, the objective of gene rearrangement is to preserve the clustering structure of the gene-interaction graph. In this paper, we use three general graph-search methods: BFS, DFS, and Max-Adjacency [2]. BFS and DFS reordering performs a breadth first search and a depth first search, respectively, on the input graph starting at a random vertex. The order in which the vertices are visited by the BFS or DFS is used to reorder the vertices. In Max-Adjacency reordering [31], starting at a random vertex, the vertex with the most edges incident to previously ordered vertices is iteratively added to the ordering.

5 Experimental Results

5.1 Graph Bisection

We tested our approach on a total of 21 graphs which consist of three groups of graphs: random graphs ($Gn.d$), random geometric graphs ($Un.d$), and caterpillar graphs ($\text{cat}.n$ and $\text{rcat}.n$). They have been used in a number of other studies

Table 1. Experimental results in graph bisection problem

Graph	Basic ordering	BFS reordering	DFS reordering	Max-Adj reordering
G500.2.5	52.48	52.38	51.50	51.94
G500.05	220.96	220.58	221.12	220.84
G500.10	630.32	629.88	630.34	629.64
G500.20	1751.48	1749.12	1750.14	1748.84
G1000.2.5	101.08	99.96	101.64	100.02
G1000.05	455.28	454.72	456.20	454.90
G1000.10	1374.04	1374.62	1372.88	1372.18
U500.05	8.66	5.88	4.66	4.72
U500.10	34.24	28.06	26.74	26.38
U500.20	178.28	178.12	178.18	178.36
U500.40	412.00	412.00	412.00	412.00
U1000.05	26.18	15.00	16.24	12.80
U1000.10	72.26	56.82	44.98	48.44
U1000.20	239.50	231.62	228.16	230.32
U1000.40	737.00	737.00	737.00	737.00
cat.352	3.56	1.04	1.28	1.76
cat.702	7.64	8.60	4.36	5.60
cat.1052	12.32	15.84	9.76	8.40
rcat.134	1.00	1.00	1.00	1.00
rcat.554	2.48	1.92	1.52	1.32
rcat.994	3.64	2.44	2.48	2.68

Average over 100 runs.

[20] [7] [10] [4]. Table 1 shows the experimental results. In the table, “BFS,” “DFS,” and “Max-Adj” represent the gene preprocessing methods. We should note again that this preprocessing is performed on the gene-interaction graphs which are independent of problems differently from previous static reordering methods such as [6] and [10]. We arrange the genes randomly in “Basic ordering.” The GAs have the same framework except the preprocessing. Preprocessed GAs significantly outperformed the GA in which genes are arranged randomly. In particular, the preprocessing showed larger performance improvement on geometric graphs and caterpillar graphs. We should also note that we can get improved solutions by using a stronger local optimization heuristic than 2-Opt. Here, we fixed the local optimization with 2-Opt since our major concern is the effect of the suggested reordering method.

To get visual insight into the reordering, we drew a problem instance and the preprocessed order in Figure 4. Figure 4(a) shows the original graph. Figure 4(b) and 4(c) were acquired by drawing segments between all consecutive vertices in each ordering. Of course, randomly ordered genes do not reflect any relation between most pair of vertices. We can observe that BFS helps highly related genes to stay close in the chromosome.

Table 2. Experimental results in linear arrangement problem

Graph	Basic ordering	BFS reordering	DFS reordering	Max-Adj reordering
U500.05	162852.40	158473.18	159004.50	158562.42
U500.10	317022.28	306477.68	308955.18	307840.62
U1000.05	658743.66	643714.92	647381.98	643003.50
U1000.10	1355660.12	1333089.58	1339380.60	1335310.82

Average over 100 runs.

Table 3. Experimental results in TSP

Instance	Basic ordering	BFS reordering	DFS reordering	Max-Adj reordering
lin318	42499.67	42451.82	42404.98	42406.36
pcb442	51886.53	51476.48	51510.38	51489.06
att532	28629.40	28173.80	28209.30	28199.80
rat783	9304.47	9104.72	9111.20	9095.08

Average over 100 runs.

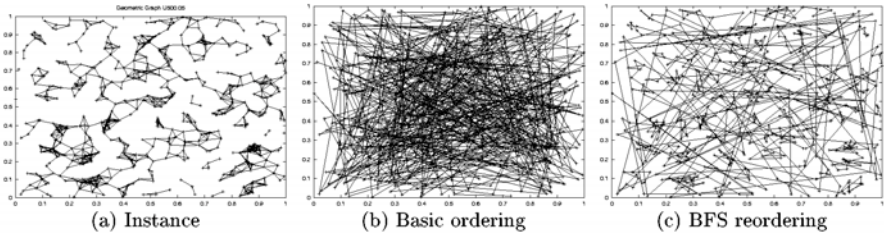


Fig. 5. Reordering in linear arrangement: instance U500.05

5.2 Linear Arrangement

To test our approach on the linear arrangement problem, we used sparse geometric graphs out of the graphs used in Table 1. Table 2 shows the experimental results. The three methods outperformed “Basic ordering” in all instances. In particular, “BFS” showed the best performance on the average. We also drew in Figure 5 the order of genes after a reordering on the linear arrangement problem. We can also observe that highly related genes stay close in the chromosome.

5.3 Traveling Salesman Problem

Table 3 shows the experimental result on four instances of TSPLIB¹. The results are consistent with the two previous experiments. GAs preprocessed by BFS, DFS, and Max-Adj have more chances to find good solutions than the basic ordering. Figure 6 shows the drawing of the gene orders on an instance att532. We can observe that closely located cities tend to locate closely in the

¹ <http://www.iwr.uniheidelberg.de/iwr/comopt/soft/TSPLIB95/TSPLIB.html>

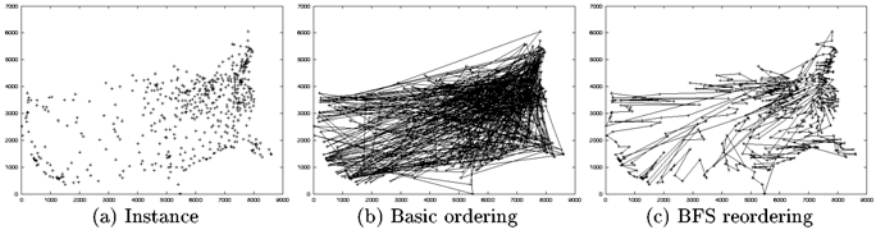


Fig. 6. Reordering in TSP: instance att532

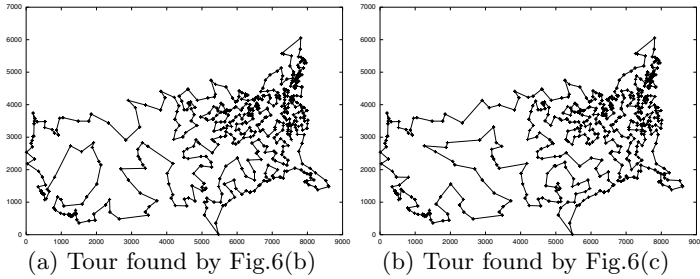


Fig. 7. TSP solutions: instance att532

chromosome with the BFS reordering. We can also observe the effect of reordering by visualizing the TSP tour. Figure 7 shows representative tours of TSP by GAs with the orderings of Figure 6. The GAs found considerably different tours according to their orderings.

6 Conclusions

In this paper, we proposed a static reordering framework of genes in locus-based encodings. It showed consistent performance improvement over genetic algorithms without reordering. One may be able to devise a better reordering, as a result of exploiting problem-specific knowledge, as far as each problem is concerned. The most notable feature of the suggested method is that it does not need any problem-specific information during the reordering process. When a new problem is given for GA, we do not have to devise a new preprocessing heuristic. The only thing we need is a measure of gene interaction for each problem. However, it may not be a big burden since most problem encodings can be classified into a number of representative encodings. Moreover, there exist useful studies on gene interactions [13] [35] [36] [14] [30] [33].

We considered only the linear encoding in this study. Although it is traditional and the most popular encoding, multi-dimensional encodings are also becoming common in the GA community [9] [22]. The proposed reordering framework has a limitation that it can be just applied to the linear encoding. Extending the reordering to multi-dimensional encodings seems to be a topic worth trying.

Acknowledgment. This work was partly supported by Optus Inc. and Brain Korea 21 Project. The RIACT at Seoul National University provided research facilities for this study.

References

1. D. Adolphson and T. Hu. Optimal linear ordering. *SIAM J. Appl. Math.*, 25(3):403–423, 1973.
2. C. Alpert and A. B. Kahng. A general framework for vertex orderings, with applications to netlist clustering. In *IEEE/ACM International Conference on Computer-Aided Design*, pages 63–67, 1994.
3. J. Bagley. *The Behavior of Adaptive Systems Which Employ Genetic and Correlation Algorithms*. PhD thesis, University of Michigan, Ann Arbor, MI, 1967.
4. R. Battiti and A. Bertossi. Greedy, prohibition, and reactive heuristics for graph partitioning. *IEEE Trans. on Computers*, 48(4):361–385, 1999.
5. M.J. Blesa, P. Moscato, and F. Xhafa. A Memetic Algorithm for the Minimum Weighted k -Cardinality Tree Subgraph Problem. In *4th Metaheuristics International Conference*, volume 1, pages 85–90, 2001.
6. T. N. Bui and B. R. Moon. Hyperplane synthesis for genetic algorithms. In *Fifth International Conference on Genetic Algorithms*, pages 102–109, July 1993.
7. T. N. Bui and B. R. Moon. A genetic algorithm for a special class of the quadratic assignment problem. *The Quadratic Assignment and Related problems, DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, 16:99–116, 1994.
8. T. N. Bui and B. R. Moon. A new genetic approach for the traveling salesman problem. In *IEEE Conference on Evolutionary Computation*, pages 7–12, June 1994.
9. T. N. Bui and B. R. Moon. On multi-dimensional encoding/crossover. In *Sixth International Conference on genetic Algorithms*, pages 49–56, 1995.
10. T. N. Bui and B. R. Moon. Genetic algorithm and graph partitioning. *IEEE Trans. on Computers*, 45(7):841–855, 1996.
11. E. K. Burke, J. P. Newall, and R. F. Weare. A memetic algorithm for university exam timetabling. In *1st International Conference on the Practice and Theory of Automated Timetabling (ICPTAT'95, Napier University, Edinburgh, UK, 30th Aug – 1st Sept 1995)*, pages 496–503, 1995.
12. C. Cheng. Linear placement algorithms and applications to VLSI design. *Networks*, 17:439–464, 1987.
13. Y. Davidor. Epistasis variance: A viewpoint on ga-hardness. In *Foundations of Genetic Algorithms 3*, pages 23–35. Morgan Kaufmann, 1991.
14. Cyril Fonlupt, Denis Robilliard, and Philippe Preux. A bit-wise epistasis measure for binary search spaces. *Lecture Notes in Computer Science*, 1498:47ff., 1998.
15. M. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman, San Francisco, 1979.
16. D. Goldberg, B. Korb, and K. Deb. Messy genetic algorithms: Motivation, analysis, and first results. *Complex System*, 3:493–530, 1989.
17. D. Goldberg and R. Lingle. Alleles, loci, and the traveling salesman problem. In *First International Conference on Genetic Algorithms and Their Applications*, pages 154–159, 1985.

18. G. R. Harik and D. E. Goldberg. Learning linkage. In *Foundations of Genetic Algorithms 4*, pages 247–262. 1996.
19. J. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, 1975.
20. D. S. Johnson, C. Aragon, L. McGeoch, and C. Schevon. Optimization by simulated annealing: An experimental evaluation, Part 1, graph partitioning. *Operations Research*, 37:865–892, 1989.
21. S. Jung and B. R. Moon. Toward minimal restriction of genetic encoding and crossovers for the 2D Euclidean TSP. *IEEE Transactions on Evolutionary Computation*, 6(6), 2002.
22. A. B. Kahng and B. R. Moon. Toward more powerful recombinations. In *Sixth International Conference on genetic Algorithms*, pages 96–103, 1995.
23. Y. H. Kim and B. R. Moon. A hybrid genetic search for graph partitioning based on lock gain. In *Genetic and Evolutionary Computation Conference*, pages 167–174, 2000.
24. P. Merz and B. Freisleben. Genetic local search for the TSP: New results. In *IEEE Conference on Evolutionary Computation*, pages 159–164, 1997.
25. P. Merz and B. Freisleben. Memetic algorithms and the fitness landscape of the graph bi-partitioning problem. In *Proceedings of the 5th International Conference on Parallel Problem Solving From Nature*, 1998. *Lecture Notes in Computer Science*, 1498:765–774, Springer-Verlag.
26. P. Merz and B. Freisleben. Fitness landscape analysis and memetic algorithms for the quadratic assignment problem. *IEEE-EC*, 4(4):337, November 2000.
27. Peter Merz and Bernd Freisleben. A comparison of memetic algorithms, tabu search, and ant colonies for the quadratic assignment problem. In *Proceedings of the Congress on Evolutionary Computation*, volume 3, pages 2063–2070. IEEE Press, 6-9 1999.
28. Peter Merz and Bernd Freisleben. Fitness landscapes and memetic algorithm design. In David Corne, Marco Dorigo, and Fred Glover, editors, *New Ideas in Optimization*, pages 245–260. McGraw-Hill, 1999.
29. B. R. Moon and C. K. Kim. A two-dimensional embedding of graphs for genetic algorithms. In *International Conference on Genetic Algorithms*, pages 204–211, 1997.
30. M. Munetomo and D. Goldberg. Identifying linkage by nonlinearity check, 1998.
31. H. Nagamochi and T. Ibaraki. Computing edge-connectivity in multigraphs and capacitated graphs. *Siam J. of Disc. Math*, 5(1):54–66, Feb 1992.
32. Y. Nagata and S. Kobayashi. Edge assembly crossover: A high-power genetic algorithm for the traveling salesman problem. In *7th International Conference on Genetic Algorithms*, pages 450–457, 1997.
33. Martin Pelikan, David Goldberg, and Fernando Lobo. A survey of optimization by building and using probabilistic model. Technical Report 99018, IlliGAL, September 1999.
34. Nicholas J. Radcliffe and Patrick D. Surry. Formal memetic algorithms. In *Evolutionary Computing, AISB Workshop*, pages 1–16, 1994.
35. Colin Reeves and Christine Wright. An experimental design perspective on genetic algorithms. In *Foundations of Genetic Algorithms 3*, pages 7–22. Morgan Kaufmann, 1995.
36. Colin Reeves and Christine C. Wright. Epistasis in genetic algorithms: An experimental design perspective. In *Proceedings of the Sixth International Conference on Genetic Algorithms*, pages 217–224. Morgan Kaufmann, 1995.

37. Y. Saab and C. Chen. An effective solution to the linear placement problem. *VLSI Design Journal*, 2(2):117–129, 1994.
38. D. I. Seo and B. R. Moon. Voronoi quantized crossover for traveling salesman problem. In *Genetic and Evolutionary Computation Conference*, pages 544–552, 2002.

Investigation of the Fitness Landscapes and Multi-parent Crossover for Graph Bipartitioning

Yong-Hyuk Kim and Byung-Ro Moon

School of Computer Science & Engineering, Seoul National University
Shillim-dong, Kwanak-gu, Seoul, 151-742 Korea
{yhdfly, moon}@soar.snu.ac.kr

Abstract. An empirical study is performed on the local-optimum space of graph bipartitioning. We examine some statistical features of the fitness landscape. They include the cost-distance correlations and the properties around the central area of local optima. The study revealed some new notable results about the properties of the fitness landscape; e.g., the central area yielded fairly good quality in the local-optimum space. We performed an experiment on a spectrum of different exploitation strengths of the central areas. From the results, it seems attractive to exploit the central area, but excessive or insufficient exploitation is not desirable.

1 Introduction

An NP-hard problem such as graph partitioning problem or traveling salesman problem (TSP) has a finite solution set and each solution has a cost. Although finite, the problem space is intractably large even for a small but nontrivial problem. It is almost impossible to find an optimal solution for those problems by exhaustive or simple search methods. Thus, in case of NP-hard problems, heuristic algorithms are being used. Heuristic algorithms provide reasonable solutions in acceptable computing time but have no performance guarantee.

Consider a combinatorial problem $C = (\Omega, f)$ and a local optimization algorithm $L_c : \Omega \rightarrow \Omega$, where Ω is the solution space and f is the cost function. If a solution $s^* \in \Omega$ is in $L_c(\Omega)$, then s^* is called a local optimum with respect to the algorithm L_c . For each local optimum $s^* \in L_c(\Omega)$, we define the neighborhood set of s^* to be a set $N(s^*) \subset \Omega$ such that, for every s in $N(s^*)$, $L_c(s)$ is equal to s^* . That is, s^* is the attractor of the solutions in $N(s^*)$. We examine the space $L_c(\Omega)$ and hope to get some insight into the problem space. This is an alternative for examining the intractably huge whole problem space. Good insight into the problem space can provide a motivation for a good search algorithm.

A number of studies about the ruggedness and the properties of problem search spaces have been conducted. Sorkin [21] defined the fractalness of a solution space and proposed that simulated annealing [18] is efficient when the space is fractal. Jones and Forrest [14] introduced the fitness-distance correlation as a measure of search difficulty. Manderick *et al.* [19] measured the ruggedness of a

problem space by autocorrelation function and correlation length obtained from a time series of solutions. Weinberger [23] conjectured that, if all points on a fitness landscape are correlated relatively highly, the landscape is bowl shaped. Boese *et al.* [3] suggested that, through measuring cost-distance correlations for the TSP and the graph bisection problem, the cost surfaces are globally convex; from these results they proposed an adaptive multi-start heuristic and showed that the heuristic is efficient [3]. Kauffman [15] proposed the NK-landscape model that can control the ruggedness of a problem space.

In this paper, we present a number of experiments to analyze problem spaces more elaborately. We examine the cost-distance correlations and the properties around the central areas of local optima. Based on the empirical study, we perform an experiment on a spectrum of different exploitation strengths of the central areas under a genetic algorithm (GA) framework. We perform these experiments on the graph bipartitioning problem.

The remainder of this paper is organized as follows. In Section 2, we summarize the graph bipartitioning problem, the Fiduccia-Mattheyses algorithm (FM) which is used as a major local optimization algorithm in this paper, and test graphs. We perform various experiments and analyze fitness landscapes in Section 3. In Section 4, we propose a multi-parent crossover for graph bipartitioning. Finally, we make our conclusions in Section 5.

2 Preliminaries

2.1 Graph Bipartitioning

Let $G = (V, E)$ be an unweighted undirected graph, where V is the set of vertices and E is the set of edges. A bipartition (A, B) consists of two subsets A and B of V such that $A \cup B = V$ and $A \cap B = \phi$. The *cut size* of a bipartition is defined to be the number of edges whose endpoints are in different subsets of the bipartition. The bipartitioning problem is the problem of finding a bipartition with minimum cut size. If the difference of cardinalities between two subsets is at most one, the problem is called *graph bisection* problem and if the difference does not exceed the fixed ratio of $|V|$, the problem is called *roughly balanced bipartitioning* problem. Without balance criterion, we can find the optimal solution in polynomial time by maxflow-mincut algorithm [10]. In a roughly balanced bipartitioning problem, 10% of skewness is usually allowed [20]. Since it is NP-hard for general graphs [11], heuristic algorithms are used practically. These include FM algorithm [9], a representative linear time heuristic, PROP [5] based on probabilistic notion, LG [17] based on lock gain, etc. In this paper, we consider only roughly balanced bipartitioning problem allowing 10% of skewness.

2.2 Fiduccia-Mattheyses Algorithm (FM)

Fiduccia and Mattheyses [9] introduced a heuristic for roughly balanced bipartitioning problem. The FM algorithm as well as the Kernighan-Lin algorithm

```

do {
    Compute gain  $g_v$  for each  $v \in V$ ;
    Make gain lists of  $g_v$ s;
     $Q = \phi$ ;
    for  $i = 1$  to  $|V| - 1$  {
        Choose  $v_i \in V - Q$  such that  $g_{v_i}$  is maximal and
            the move of  $v_i$  does not violate the balance criterion;
         $Q = Q \cup \{v_i\}$ ;
        for each  $v \in V - Q$  adjacent to  $v_i$ 
            Update its gain  $g_v$  and adjust the gain list;
    }
    Choose  $k \in \{1, \dots, |V| - 1\}$  that maximizes  $\sum_{i=1}^k g_{v_i}$ ;
    Move all the vertices in the subset  $\{v_1, \dots, v_k\}$  to their opposite sides;
} until (there is no improvement)

```

Fig. 1. The Fiduccia-Mattheyses algorithm (FM)

(KL) [16] is a traditional iterative improvement algorithm. The algorithm improves on an initial solution by single-node moves. The main difference between KL and FM lies in that a new partition in FM is derived by moving a single vertex, instead of KL's pair swap. The structure of the FM algorithm is given in Figure 1. FM proceeds in a series of *passes*. In each pass, all vertices are moved in chain and then the best bipartition during the pass is returned as a new solution. The algorithm terminates when one or a few passes fail to find a better solution. With an efficient data structure, each pass of FM runs in $\Theta(|E|)$ time.

2.3 Test Beds

We tested on a total of 17 graphs which consist of two groups of graphs. They are composed of 17 graphs from [13] (9 random graphs and 8 geometric graphs). The two classes were used in a number of other studies [20] [4] [1] [17]. The classes are briefly described below.

1. *Gn.d*: A random graph on n vertices, where an edge is placed between any two vertices with probability p independent of all other edges. The probability p is chosen so that the expected vertex degree, $p(n - 1)$, is d .
2. *Un.d*: A random geometric graph on n vertices that lie in the unit square and whose coordinates are chosen uniformly from the unit interval. There is an edge between two vertices if their Euclidean distance is t or less, where $d = n\pi t^2$ is the expected vertex degree.

3 Investigation of the Problem Space

In this section, we first extend the experimentation of Boese *et al.* [3] to examine the local-optimum space. We denote by local-optimum space the space consisting of all local optima with respect to a local optimization algorithm. Next, we examine the area around the “central point” of local optima. In our experiments, we use a sufficiently large number of local optima. We do not care

about solutions other than local optima. The local optimizer in our experiments is the FM algorithm.

In the graph bipartitioning problem for a graph $G = (V, E)$, each solution (A, B) is represented by a $|V|$ -bits code. Each bit corresponds to a vertex in the graph. A bit has value zero if the vertex is in the set A , and has value one otherwise. In this encoding, a vertex move in the FM algorithm changes the solution by one bit. Thus, it is natural to define the distance between two solutions by the Hamming distance. However, if the Hamming distance between two solutions is $|V|$, they are symmetric and equal. We hence define the *distance* between two solutions as follows.

Definition 1 Let the universal set U be $\{0, 1\}^{|V|}$. For $\mathbf{a}, \mathbf{b} \in U$, we define the distance between $\mathbf{a}$ and $\mathbf{b}$ as follows:¹

$$d(\mathbf{a}, \mathbf{b}) = \min(\mathfrak{H}(\mathbf{a}, \mathbf{b}), |V| - \mathfrak{H}(\mathbf{a}, \mathbf{b}))$$

where $\mathfrak{H}$ is the Hamming distance.

By the definition, $0 \leq d(\mathbf{a}, \mathbf{b}) \leq \lfloor |V|/2 \rfloor$ while $0 \leq \mathfrak{H}(\mathbf{a}, \mathbf{b}) \leq |V|$.

3.1 Cost-Distance Correlation

Given a set of local minima, Boese *et al.* [3] plotted, for each local minimum, i) the relationship between the cost and the average distance from all the other local minima, and ii) the relationship between the cost and the distance to the best local minimum. They performed experiments for the graph bisection and the traveling salesman problem, and showed that both problems have strong positive correlations for both i) and ii) in the above. This fact hints that the best local optimum is located near the center of the local-optimum space. From their experiments, they conjectured that cost surfaces of both problems are globally convex. In this subsection, we repeat their experiments for other graphs and extend their study to get more insight.

The solution space for the experiment is selected as follows. First, we choose thousands of random solutions and obtain the corresponding set of local optima by locally optimizing them. Next, we remove the duplicated solutions in the set if any. Figure 2 shows the plotting results for the graph U500.10. It is consistent with Boese *et al.*'s results with strong cost-distance correlation. More statistics for a number of graphs are given in Table 1. The meaning of each item in the table is as follows. "Population size" means the number of local optima we used for each graph. "Best cut" is the cost of the best local optimum. "Average cut" is the average cost of the local optima. "Cost-distance correlation" is the correlation

¹ Given an element $\mathbf{a} \in U$, there is only one element such that it is different from $\mathbf{a}$ and the distance d to $\mathbf{a}$ is zero. If the distance between two elements is equal to zero, we define them to be in relation R . Then, the relation R is an equivalence relation. Suppose Q is the quotient set of U by relation R , it is easily verified that (Q, d) is a metric space.

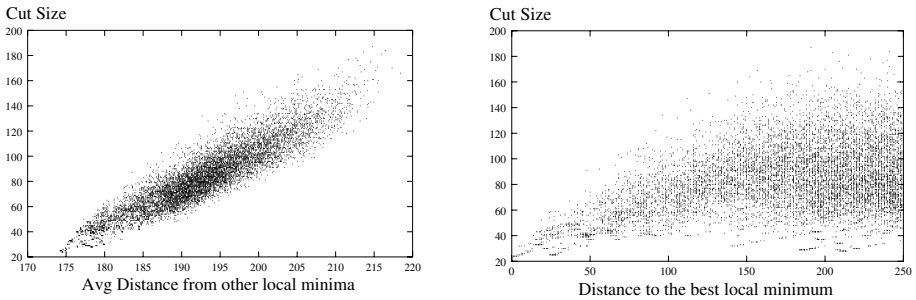


Fig. 2. Relationship between cost and distance: U500.10 (see Table 1)

Table 1. The results for each graph

Items	G250.10	G500.2.5	G1000.2.5	U500.05	U500.10	U1000.05
Population size	9877	10000	10000	10000	9302	10000
Best cut	352	52	103	5	24	16
Average cut	367.65	64.58	128.17	35.62	83.58	70.76
Cost-distance correlation	0.77	0.78	0.83	0.89	0.91	0.88
Central point cut (CP)	380	60	118	5	24	17
CP + FM	352	51	99	5	24	16
Average distance	102.94	217.11	453.44	215.63	192.83	448.09

Population size : the number of local optima
Best cut : the minimum cost
Average cut : the average cost
Cost-distance correlation : correlation coefficient between cost and average distance from each local optimum to others
Central point cut (CP) : the cost of the approximate central point in solution space
CP + local opt : the cost after local optimization on the approximate central point
Average distance : the average value of distances between a pair of local optima

coefficient between the costs of local optima and the average distances from the other local optima. “Central point cut (CP)” is the cost of the approximate central point of the local-optimum space (see Section 3.2 for the approximate central point). “CP + local opt” is the cost after local optimization on the approximate central point. Finally, “Average distance” means the average distance between a pair of local optima. Overall, each graph showed strong positive correlation. Depending on graphs, correlation coefficients were a bit different. Geometric graphs showed larger correlation coefficients than random graphs. In the statistical data of Table 1, each population was obtained from 10,000 random initial solutions. Among the six graphs, four graphs had no duplications and the other two graphs had 123 and 698 duplications, respectively. It is surprising that there were no duplications in the first 10,000 attractors for four of them. It seems to suggest that the number of all possible local optima with respect to FM is immeasurably large.

Figure 3, Table 2, and Table 3 compare the data with different local optimizers. A greedy local optimizer which moves only vertices with positive gain was named GREEDY. Its principle is the same as that of the steepest descent algorithm in the differentiable cases. NONE means a set of random solutions without any local optimization. From the cut sizes in Table 2 and Table 3, FM is clearly stronger than the GREEDY algorithm. The stronger the local opti-

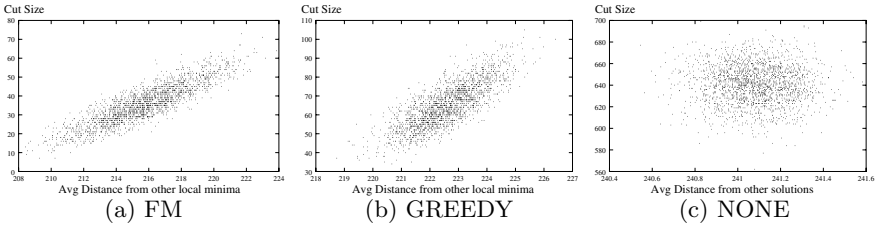


Fig. 3. Relationship between cost and distance with different local optimizer in the graph U500.05 (see Table 3)

Table 2. The data comparison with different local optimizer in the graph G500.10

Local opt	FM	GREEDY	NONE
Population size	2000	2000	2000
Best cut	623	666	1101
Average cut	648.60	706.26	1178.00
Cost-distance correlation	0.77	0.81	−0.02
Central point cut (CP)	659	670	1138
CP + local opt	623	643	—
Average distance	218.58	229.71	241.09

Table 3. The data comparison with different local optimizer in the graph U500.05

Local opt	FM	GREEDY	NONE
Population size	2000	2000	2000
Best cut	7	34	562
Average cut	35.86	65.16	640.89
Cost-distance correlation	0.88	0.79	−0.02
Central point cut (CP)	5	33	581
CP + local opt	5	30	—
Average distance	215.71	222.58	241.08

mizer, the smaller the average distance between two local optima and the more sharing among local optima. However, from Tables 1–3, it is surprising that, differently from our expectation, the average distance between two arbitrary local optima is nearly 80% ~ 90% of the possible maximum distance $\lfloor |V|/2 \rfloor$. This is an evidence of the huge diversity of local optima. In Figure 3, a stronger local optimization shows stronger cost-distance correlation. Since the average distances in graphs are various, these values may have some potential to be used as measures of the problem difficulty with respect to a local optimizer.²

3.2 Approximate Central Point

The results of Boese *et al.* [3] for the TSP and the graph bisection problem suggest that the best solution is located near the center of the local-optimum space. As a result of this, given a subspace of local optima for a problem, the “central

² This is not a simple issue, though.

point”³ of the subspace may be near the optimal solution. Hence, computing the “central point” not only supports the results of Boese *et al.* but may also be helpful for obtaining the optimal solution.

Given a subspace Ω' of the whole solution space in the graph bipartitioning problem, the “approximate central point”⁴ is computed as follows. Let one of the two encodings of the best solution in Ω' be p_{best} . First, since each solution has a pair of encodings, we make a set $S_{\Omega'}$ that contains only one encoding e for each solution in Ω' so that the Hamming distance between e and p_{best} is not greater than $\lfloor |V|/2 \rfloor$. Next, for each position, count the number of 0's and that of 1's for all elements of $S_{\Omega'}$. Make the approximate central point c so that each position of c has the more-frequently-appeared bit. Then, the approximate central point c is closer to the center⁵ of Ω' than p_{best} .⁶ That is, we have the following proposition.

Proposition 1 $\forall p_{best} \in S_{\Omega'}$, let $S_{\Omega'} = \{s_1, s_2, \dots, s_n\}$. Then,

$$\sum_{i=1}^n d(p_{best}, s_i) \geq \sum_{i=1}^n d(c, s_i).$$

Proof: Let $B_j(x)$ be the j^{th} value of x .

$$\begin{aligned} \sum_{i=1}^n d(p_{best}, s_i) &= \sum_{i=1}^n \mathfrak{H}(p_{best}, s_i) \\ &= \sum_{i=1}^n \sum_{j=1}^{|V|} |B_j(p_{best}) - B_j(s_i)| \\ &= \sum_{j=1}^{|V|} \sum_{i=1}^n |B_j(p_{best}) - B_j(s_i)| \\ &= \sum_{j=1}^{|V|} |\{s \in S_{\Omega'} : B_j(s) \neq B_j(p_{best})\}| \end{aligned}$$

³ We define “central point” to be the nearest solution to the center of local-optimum space.

⁴ In this problem, it is not easy to find the exact central point by a simple computation. Each solution has two different encodings. In order to get the distance to other solution, we select one to which the Hamming distance is smaller than the other. The more the solutions, the more complex the whole phase about which encoding is used to calculate the distance.

⁵ Here, the center of Ω' is defined to be the point that has the minimum average distance from the other solutions in Ω' .

⁶ Since the approximate central point obtained in this way can violate balance criterion, adjustment is required. Although not mentioned, the experimental data showed that most of adjusted approximate central points were closer to the center of Ω' than p_{best} .

$$\begin{aligned}
&\geq \sum_{j=1}^{|V|} |\{s \in S_{\Omega'} : B_j(s) \neq B_j(c)\}| \\
&= \sum_{i=1}^n \mathfrak{H}(c, s_i) \\
&\geq \sum_{i=1}^n d(c, s_i).
\end{aligned}$$

Q.E.D.

Although the approximate central points are calculated through a simple computation, it turned out that the costs of the approximate central points are quite attractive (see Tables 1–3). It is amazing that the cut size of the approximate central point without any fine-tuning was sometimes equal to or better than that of the best solution (see the cases of U500.05 and U500.10 in Table 1). In order to check the local optimum near the center, we applied local optimization to the approximate central point. The results are in the row “CP + local opt” of Tables 1–3. In all of the ten cases, the costs of the local optima near the approximate central points were at least as good as those of the best solutions; surprisingly enough, they were better than those of the best solutions in five cases of them. This shows the attractiveness of the central area of the local-optimum space, and provides a motivation for intensive search around the central area.

4 Exploiting Approximate Central Points

We observed in Section 3.2 that the approximate central points obtained by simple computation are quite attractive. In this section, we propose a pseudo-GA that exploits the areas around the approximate central points. Based on the GA, we perform an experiment on how strong exploitation of the central area is desirable.

4.1 A Pseudo-GA That Exploits the Central Areas

Multi-parent crossover is a generalization of the traditional two parent recombination. It was first introduced by Eiben *et al.* [8] and has been extensively studied in the past [7] [6] [22]. But, it is not adequate for problems with multiple representations for a solution like the graph partitioning problem. Unlike previous works, our multi-parent crossover is designed based on the statistical features of problem spaces. The offspring of our multi-parent crossover is exactly the approximate central point of the solutions in the population. Formally, the process of our multi-parent crossover is as follows. Consider the distance measure d defined in Section 3, the parent set P , and a parent p_k in P .⁷ For each parent $a \in P$, if

⁷ Assume that the distance d between any two elements in P is larger than zero.

$\mathfrak{H}(p_k, a) > \lfloor |V|/2 \rfloor$, make a transition that interchanges 0 and 1 at every gene position of a . Let the new set resulted from the transitions be $P' = \{p_1, p_2, \dots, p_n\}$. Then, for each $i = 1, 2, \dots, n$ ($p_i \neq p_k$), $0 < \mathfrak{H}(p_k, p_i) \leq \lfloor |V|/2 \rfloor$. Now, generate an offspring c such that for each $j = 1, 2, \dots, |V|$

$$B_j(c) = \begin{cases} 1, & \text{if } |\{p \in P' : B_j(p) = 1\}| > \lfloor n/2 \rfloor \\ 0, & \text{otherwise} \end{cases}$$

where $B_j(x)$ is the j^{th} gene value of x .

Figure 4 shows the template of pseudo-GA that is designed to exploit the central areas. It is a type of hybrid steady-state genetic algorithm using the multi-parent crossover described above.

- *Initialization and genetic operators:* The GA first creates K local optima at random. We set the population size K to be from 10 to 100. Simply, the total population is selected as the parent set and the GA performs K -parent crossover on them. Then the mutation perturbs the offspring by R percent. Mutation is important in this model since the crossover strongly drives the offspring to the central area; thus an appropriate degree of perturbation is needed.
- *Local optimization:* One of the most common local optimization heuristic for graph partitioning is the FM algorithm. We apply it to the offspring after mutation.
- *Replacement and stopping condition:* After generating an offspring and applying a local optimization on it, the GA replaces the most similar member of the population with the offspring. Maintaining population diversity, a randomly generated local optimum replaces the most similar member of the population per generation. It stops after a fixed number, $(M - K)/2$, of generations.

4.2 Experimental Results

We tested the GA with a number of different population sizes. The population size is denoted by K . K is also the number of parents for crossover; in other words, it is the degree of exploitation around the central area. The values of K represent a spectrum of the exploitation strengths of the central area. If K is equal to M , since it just generates initial population without genetic search, the heuristic equals the multi-start heuristic. The multi-start heuristic returns the best local optimum among a considerable number of local optima fine-tuned from random initial solutions. Although the multi-start heuristic is simple, it has been useful in a number of studies [12] [2].

The experimental results are given in Table 4. We used the FM algorithm as the local optimizer. We set M and R to 1,000 and 20 respectively in all cases and performed 100 runs for each case.

Overall, one can observe that it is helpful to exploit the central areas to some extent. Figure 5 shows two sample plottings, which have roughly bitonic

```

MPGA( $M, K, R$ )
//  $M$ : running time budget,  $K$ : population size, and  $R$ : perturbation rate
{
  for each  $i = 1, 2, \dots, K$  // Generate initial population with size  $K$ 
  {
    Generate a random individual  $P_i$ ;
     $P_i \leftarrow \text{local\_opt}(P_i)$ ;
  }
   $B \leftarrow$  the best among population;
  do {
    Make an offspring  $C$  using  $K$ -parent crossover from population;
     $C \leftarrow R\%\_random\_mutation(C)$ ;
     $C^* \leftarrow \text{local\_opt}(C)$ ;
    Replace the most similar individual from population with  $C^*$ ;
    Generate a random individual  $T$ ;
     $T \leftarrow \text{local\_opt}(T)$ ;
    Replace the most similar individual from population with  $T$ ;
     $B \leftarrow$  the best among  $B, C^*$ , and  $T$ ;
  } until (the number of generations is  $(M - K)/2$ )
  return  $B$ ;
}

```

Fig. 4. A simple genetic algorithm using multi-parent crossover

spectra of performance. The results of Table 4 and Figure 5 show that it is useful to exploit the central area, but that excessive or insufficient exploitation is not desirable.

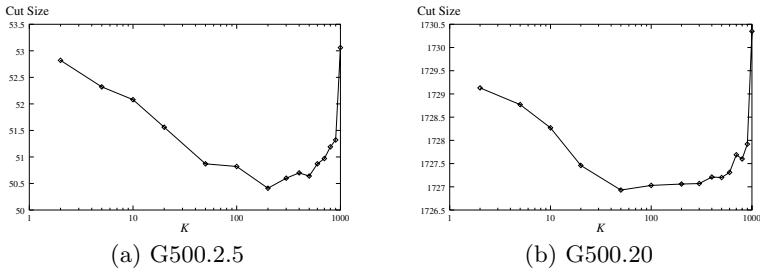


Fig. 5. Two sample spectra extended from Table 4

5 Conclusions

The fitness landscape of the problem space is an important factor to indicate the problem difficulty, and the analysis of the fitness landscape helps efficient search in the problem space. In this paper, we made a number of experiments and got some new insights into the global structure of the graph-partitioning problem space. We extended previous works and observed that the central area of multiple local optima is quite attractive.

Table 4. The comparison of cut sizes

Graphs	$K = 2$		$K = 5$		$K = 10$		$K = 20$	
	Ave [†]	CPU [‡]	Ave [†]	CPU [‡]	Ave [†]	CPU [‡]	Ave [†]	CPU [‡]
G500.2.5	52.82	1.56	52.32	1.51	52.08	1.56	51.56	1.63
G500.05	216.09	2.13	215.10	2.09	214.54	2.13	214.70	2.19
G500.10	622.28	3.41	621.46	3.36	620.64	3.41	619.96	3.45
G500.20	1729.13	6.77	1728.77	7.53	1728.27	7.62	1727.46	7.20
G1000.2.5	105.51	3.51	104.28	3.46	103.79	3.53	102.72	3.66
G1000.05	445.68	4.81	454.79	4.76	452.82	4.87	451.53	5.02
G1000.10	1363.76	8.63	1361.27	10.02	1359.44	9.65	1358.09	8.59
G1000.20	3357.62	27.04	3353.72	28.13	3350.87	26.45	3349.23	28.37
U500.05	7.64	1.95	7.22	1.83	7.06	1.92	6.67	1.98
U1000.05	24.60	4.84	25.73	4.51	24.35	5.30	22.81	4.87
U1000.10	41.84	7.18	41.68	6.78	40.64	9.84	40.31	7.35

Graphs	$K = 50$		$K = 100$		$K = 200$		$K = 500$		Multi-Start [§]	
	Ave [†]	CPU [‡]	Ave [†]	CPU [‡]	Ave [†]	CPU [‡]	Ave [†]	CPU [‡]	Ave [†]	CPU [‡]
G500.2.5	50.87	1.92	50.82	2.40	50.41	3.22	50.64	4.64	53.06	1.63
G500.05	214.12	2.45	213.95	2.92	213.92	3.75	213.78	5.19	217.23	2.20
G500.10	619.55	3.71	619.38	4.24	619.38	5.06	619.94	6.58	623.07	3.54
G500.20	1726.93	9.11	1727.03	8.38	1727.06	8.45	1727.20	11.60	1730.35	8.46
G1000.2.5	100.19	4.25	99.20	5.19	98.38	7.04	97.95	10.30	105.99	3.64
G1000.05	449.27	5.69	448.86	6.59	447.58	8.45	448.15	11.62	457.34	4.95
G1000.10	1356.30	10.10	1354.35	11.20	1354.69	12.17	1355.46	18.19	1365.14	10.21
G1000.20	3346.51	27.57	3345.77	27.56	3344.67	30.15	3345.18	33.71	3363.30	26.80
U500.05	5.60	2.24	5.04	2.71	4.98	3.55	4.87	5.06	7.98	2.07
U1000.05	20.12	5.40	17.28	6.38	14.72	8.28	12.74	11.50	24.09	5.16
U1000.10	40.13	7.74	40.55	8.48	40.86	10.51	40.97	13.58	42.89	8.43

§ $K = M (= 1000)$.

† Average over 100 runs.

‡ CPU seconds on Pentium III 750 MHz.

For the other geometric graphs (U500.10, U500.20, U500.40, U1000.20, and U1000.40) not shown here, all the methods always found the best known.

It seems clear that there are high-quality solutions clustered near the central area of local optima. Hence, it is attractive to exploit the central area. Too much exploitation of the central area perhaps makes the search diversity low. It seems desirable to exploit the central area avoiding excessive or insufficient exploitation. We showed that the performance of search could be improved by a multi-parent crossover based on the exploitation of the central area. The results presented in this paper can also be good supporting data for the previous studies on multi-parent crossover [8] [7] [6] [22]. More theoretical arguments for our empirical results are left for future study.

Our results were achieved in a specific problem, the graph partitioning problem. However, we expect that many other hard combinatorial optimization problems have similar properties. For example, in case of cost-distance correlation, TSP showed similar property to the graph partitioning problem [3]. We hope this study provides a good motivation for the investigation of problem spaces and the design of more effective search algorithms.

Acknowledgments. This work was partly supported by Optus Inc. and Brain Korea 21 Project. The RIAC at Seoul National University provided research facilities for this study.

References

1. R. Battiti and A. Bertossi. Greedy, prohibition, and reactive heuristics for graph partitioning. *IEEE Trans. on Computers*, 48(4):361–385, 1999.
2. K. D. Boese and A. B. Kahng. Best-so-far vs. where-you-are: Implications for optimal finite-time annealing. *Systems and Control Letters*, 22(1):71–78, January 1994.
3. K. D. Boese, A. B. Kahng, and S. Muddu. A new adaptive multi-start technique for combinatorial global optimizations. *Operations Research Letters*, 15:101–113, 1994.
4. T. N. Bui and B. R. Moon. Genetic algorithm and graph partitioning. *IEEE Trans. on Computers*, 45(7):841–855, 1996.
5. S. Dutt and W. Deng. A probability-based approach to VLSI circuit partitioning. In *Design Automation Conference*, pages 100–105, June 1996.
6. A. E. Eiben. Multi-parent recombination, 1997. Handbook of Evolutionary Algorithms (pp. 25–33). IOP Publishing Ltd. and Oxford University Press.
7. A. E. Eiben and T. Back. An empirical investigation of multiparent recombination operators in evolution strategies. *Evolutionary Computation*, 5(3):347–365, 1997.
8. A. E. Eiben, P.-E. Raué, and Zs. Ruttkay. Genetic algorithms with multi-parent recombination. In *Parallel Problem Solving from Nature – PPSN III*, pages 78–87, 1994.
9. C. Fiduccia and R. Mattheyses. A linear time heuristics for improving network partitions. In *19th ACM/IEEE Design Automation Conference*, pages 175–181, 1982.
10. L. R. Jr. Ford and D. R. Fulkerson. *Flows in Networks*. Princeton University Press, 1962.
11. M. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman, San Francisco, 1979.
12. D. S. Johnson. Local optimization and the traveling salesman problem. In *17th Colloquium on Automata, Languages, and Programming*, pages 446–461. Springer-Verlag, 1990.
13. D. S. Johnson, C. Aragon, L. McGeoch, and C. Schevon. Optimization by simulated annealing: An experimental evaluation, Part 1, graph partitioning. *Operations Research*, 37:865–892, 1989.
14. T. Jones and S. Forrest. Fitness distance correlation as a measure of problem difficulty for genetic algorithms. In *Sixth International Conference on Genetic Algorithms*, pages 184–192, 1995.
15. S. Kauffman. Adaptation on rugged fitness landscapes. *Lectures in the Science of Complexity*, pages 527–618, 1989.
16. B. Kernighan and S. Lin. An efficient heuristic procedure for partitioning graphs. *Bell Systems Technical Journal*, 49:291–307, Feb. 1970.
17. Y. H. Kim and B. R. Moon. A hybrid genetic search for graph partitioning based on lock gain. In *Genetic and Evolutionary Computation Conference*, pages 167–174, 2000.
18. S. Kirkpatrick, C. D. Jr. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, May 1983.
19. B. Manderick, M. de Weger, and P. Spiessens. The genetic algorithm and the structure of the fitness landscape. In *International Conference on Genetic Algorithms*, pages 143–150, 1991.

20. Y. G. Saab. A fast and robust network bisection algorithm. *IEEE Trans. on Computers*, 44(7):903–913, 1995.
21. G. B. Sorkin. Efficient simulated annealing on fractal landscapes. *Algorithmica*, 6:367–418, 1991.
22. I. G. Sprinkhuizen-Kuyper, C. A. Schippers, and A. E. Eiben. On the real arity of multiparent recombination. In *Proceedings of the Congress on Evolutionary Computation*, volume 1, pages 680–686, 6-9 1999.
23. E. D. Weinberger. Fourier and Taylor series on fitness landscapes. *Biological Cybernetics*, 65:321–330, 1991.

New Usage of Sammon's Mapping for Genetic Visualization

Yong-Hyuk Kim and Byung-Ro Moon

School of Computer Science & Engineering, Seoul National University
Shillim-dong, Kwanak-gu, Seoul, 151-742 Korea
{yhdfly, moon}@soar.snu.ac.kr

Abstract. It is a hard problem to understand the fitness landscape of a problem as well as the evolution of genetic algorithms. For the purpose, we adopt Sammon's mapping for the investigation. We demonstrate its usefulness by applying it to the graph partitioning problem which is a well-known NP-hard problem. Also, through the investigation of schema traces, we explain the genetic process and the reordering effect in the genetic algorithm.

1 Introduction

An NP-hard problem such as graph partitioning problem or traveling salesman problem (TSP) has a finite solution set and each solution has a cost. Although finite, the problem space is intractably large even for a small but nontrivial problem. A number of studies about the ruggedness and the properties of problem search spaces were done. Weinberger [17] conjectured that, if all points on a fitness landscape are correlated relatively highly, the landscape is bowl shaped. Boese *et al.* [1] suggested that, through measuring cost-distance correlation for the TSP and the graph partitioning problem, the cost surfaces are globally convex. Jones and Forrest [11] introduced fitness-distance correlation as a measure of search difficulty. Good insight into the problem space can provide a motivation for a good search algorithm [1]. We examine the problem space and hope to get some insight into the problem space.

For NP-hard problems with intractably large problem space, it is almost impossible to find an optimal solution by exhaustive or simple search methods. Thus, in case of NP-hard problems, heuristic algorithms or meta-heuristics are used. The genetic algorithm (GA) is one of the most powerful search methods among them. A number of studies for understanding GA's working mechanism were done. These include schema theorem [10], Royal Road function [13], etc.

Visualization is one of the most basic tools for studies of search spaces. A notable method for fitness landscapes is the plotting of fitness-distance correlation [1]. For GA visualization, the most popular method is the fitness flow over time as in many GA papers. Another wholesale method is the population data matrix¹ for identifying population features. In this paper, we propose new visualization

¹ In the matrix, the entire population is displayed in textual form.

techniques primarily using Sammon's mapping. We analyze the problem space for graph partitioning more elaborately. We visualize the solutions associated with the genetic search. We also trace schemata and analyze them.

The remainder of this paper is organized as follows. In Section 2, we summarize graph partitioning problem and Sammon's mapping which is used as a major tool for visualization in this paper. We analyze fitness landscapes for graph partitioning in Section 3. In Section 4, we provide some visualization for genetic algorithms. In Section 5, schema traces are visualized and analyzed. Finally, we make our conclusions in Section 6.

2 Preliminaries

2.1 Graph Partitioning

Let $G = (V, E)$ be an unweighted undirected graph, where V is the set of vertices and E is the set of edges. A bipartition (A, B) consists of two subsets A and B of V such that $A \cup B = V$ and $A \cap B = \phi$. The *cut size* of a bipartition is defined to be the number of edges whose endpoints are in different subsets of the bipartition. The bipartitioning problem is the problem of finding a bipartition with minimum cut size. If the difference of cardinalities between two subsets is at most one, the problem is called *graph bisection* problem. It is a representative NP-hard problem [8]. In this paper, we use three graphs from [3] (one geometric graph and two caterpillar graphs)² as test beds.

2.2 Sammon's Mapping

Sammon's mapping [16] is a mapping technique for transforming a dataset from a high-dimensional (say, m -dimensional) input space onto a low-dimensional (say, d -dimensional) output space (with $d < m$). The basic idea is to arrange all the data points on a d -dimensional output space in such a way that minimizes the distortion of the relationship among data points.

Sammon's mapping tries to preserve distances. This is achieved by minimizing an error criterion which penalizes the differences of distances between points in the input space and the output space. Consider a dataset of n objects. If we denote the distance between two points x_i and x_j in the input space by δ_{ij}

² The classes are briefly described below.

- i) *Un.d*: A random geometric graph on n vertices that lie in the unit square and whose coordinates are chosen uniformly from the unit interval. There is an edge between two vertices if their Euclidean distance is t or less, where $d = n\pi t^2$ is the expected vertex degree.
- ii) *rcat.n*: A caterpillar graph on n vertices. It is constructed out of a straight line (called the spine), where all the vertices on this line have degree two except the outermost two vertices. Each vertex on the spine is then connected to $\sqrt{n}$ new vertices, the legs of the caterpillar.

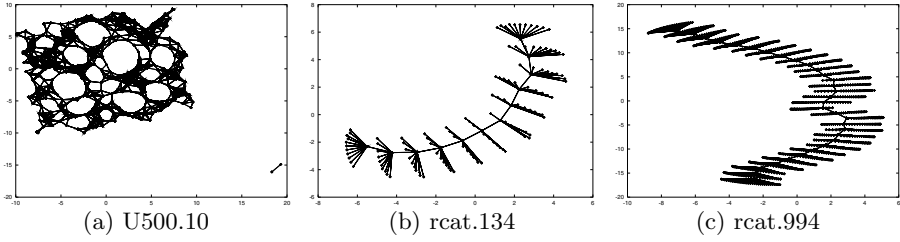


Fig. 1. Examples of Sammon's mapping

and the distances between x'_i and x'_j in the output space by d_{ij} , then Sammon's stress measure E is defined as follows:

$$E = \frac{1}{\sum_{i=1}^{n-1} \sum_{j=i+1}^n \delta_{ij}} \sum_{i=1}^{n-1} \sum_{j=i+1}^n \frac{(\delta_{ij} - d_{ij})^2}{\delta_{ij}}.$$

The stress range is $[0,1]$ with 0 indicating a lossless mapping. This stress measure can be minimized using any minimization technique. Sammon [16] proposed a technique called pseudo-Newton minimization, a steepest-descent method. The complexity of Sammon's mapping is $O(n^2m)$. There were several studies about Sammon's mapping [7] [5] [14].

The resulting output space depicts clusters of the input space as groups of data points mapped close to each other in the output space. Figure 1 shows Sammon's mapping of three graphs into 2-dimensional space. In the graphs, we defined the distance between two vertices to be the shortest path length between each other. We can observe that the mapping well accords with the characteristics of the graphs.

3 Fitness Landscapes

In this section, we first extend the experimentation of Boese *et al.* [1] to examine the local-optimum space. We mean by local-optimum space the space consisting of all local optima with respect to a local optimization algorithm. We then examine the distribution of local optima. In our experiments, we used a sufficiently large number of local optima. We do not care about solutions other than local optima. The Kernighan-Lin algorithm (KL) [12] was used for local optimization.

In the graph bisection problem for a graph $G = (V, E)$, each solution (A, B) is represented by a $|V|$ -bits code. Each bit corresponds to a vertex in the graph. A bit has value 0 if the vertex is in the set A , and has value 1 otherwise. In this encoding, a vertex move in the solution changes the solution by one bit. Thus, it is natural to define the distance between two solutions by the Hamming distance. Formally, we define the *genotype distance* between two solutions as follows.

Definition 1 Let the universal set U be $\{0, 1\}^{|V|}$. For $\mathbf{a}, \mathbf{b} \in U$, we define the genotype distance between $\mathbf{a}$ and $\mathbf{b}$ as follows:

$$d_g(\mathbf{a}, \mathbf{b}) = \mathfrak{H}(\mathbf{a}, \mathbf{b})$$

where $\mathfrak{H}$ is the Hamming distance.

However, if the genotype distance between two solutions is $|V|$, they are equal. We hence define the *phenotype distance* between two solutions as follows.

Definition 2 Let the universal set U be $\{0, 1\}^{|V|}$. For $\mathbf{a}, \mathbf{b} \in U$, we define the phenotype distance between $\mathbf{a}$ and $\mathbf{b}$ as follows:³

$$d_p(\mathbf{a}, \mathbf{b}) = \min(d_g(\mathbf{a}, \mathbf{b}), |V| - d_g(\mathbf{a}, \mathbf{b}))$$

where d_g is the genotype distance.

By the definition, $0 \leq d_p(\mathbf{a}, \mathbf{b}) \leq \lfloor |V|/2 \rfloor$ while $0 \leq d_g(\mathbf{a}, \mathbf{b}) \leq |V|$. In this paper, we use the phenotype distance d_p for the distance between two solutions unless otherwise noted.

3.1 Cost-Distance Correlation

Given a set of local minima, Boese *et al.* [1] plotted, for each local minimum, i) the relationship between the cost and the average distance to all the other local minima, and ii) the relationship between the cost and the distance to the best local minimum. They performed experiments for the graph bisection and the traveling salesman problem, and showed that both problems have strong positive correlations for both i) and ii) in the above. This fact hints that the best local optimum is located near the center of local-optimum space. From their experiments, they conjectured that the cost surfaces of both problems are globally convex. In this subsection, we repeat their experiments for another graph.

The solution space for the experiment was selected as follows. First, we chose a large number of random solutions and obtained the corresponding set of local optima by locally optimizing them. Then, we removed the duplicated solutions in the set if any. Figures 2(a) and (b) show the plotting results with 9,302 local minima⁴ for the graph U500.10. The correlation coefficient for the experiment i) was 0.91. It is consistent with Boese *et al.*'s results with strong cost-distance correlation.

³ Given an element $\mathbf{a} \in U$, there is only one element such that it is different from $\mathbf{a}$ and the distance d_p to $\mathbf{a}$ is zero. If the distance between two elements is equal to zero, we define them to be in relation R . Then, the relation R is an *equivalence relation*. Suppose Q is the quotient set of U by relation R ($Q = U/R$), it is easily verified that (Q, d_p) is a *metric space*.

⁴ There were 698 duplications among 10,000 local minima.

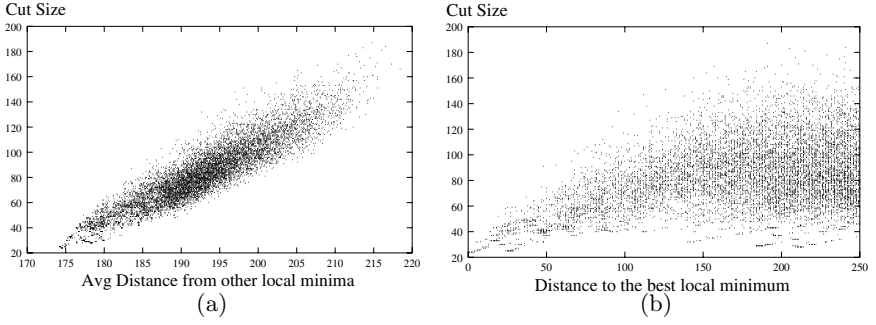


Fig. 2. Fitness-distance correlation (U500.10)

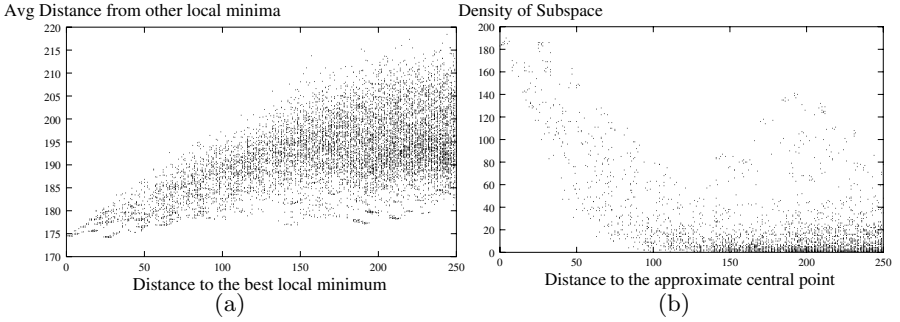


Fig. 3. Distribution of local minima (U500.10)

3.2 Distribution of Local Optima

As a result of the experiments of Boese *et al.* [1], we agree with the conjecture about the global convexity of local-optimum space but it is difficult to obtain further deduction. Figure 3(a) shows the relationship between the distance to the best local minimum and the average distance to the other local minima for each local minimum in the local-optimum space. In the figure, there are considerably many solutions such that they are far from the best solution but their average distances are small. This fact suggests that solutions may be clustered in more than one place.

We devised another way to examine the distribution of local optima. For each solution $\mathfrak{s}$ in the problem space, we chose a ball centered at $\mathfrak{s}$ with radius r (here we set r to be $|V|/8$) and counted the number of local optima inside the ball. Figure 3(b) plots the densities of the balls. It shows that the density of local optima near the center of the problem space is remarkably high. Interestingly enough, one can also observe fairly high-density areas far from the center. It suggests the existence of “medium valleys”⁵ or “small valleys.” It can not be explained by the experimental methods such as [1].

⁵ The relative notion to big valleys mentioned by Boese *et al.* [1].

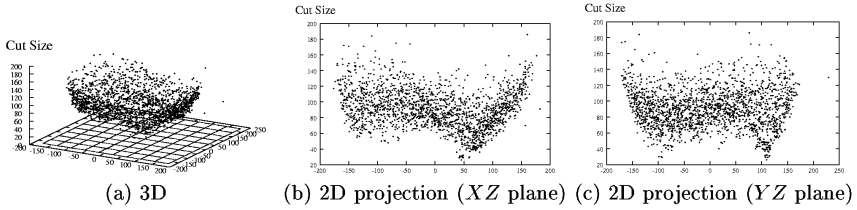


Fig. 4. Fitness landscape with Sammon's mapping (U500.10)

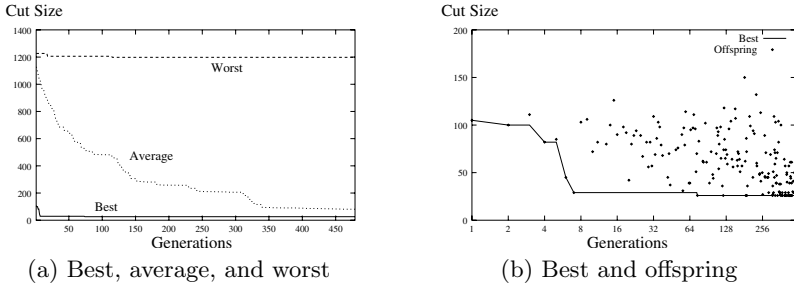


Fig. 5. Traditional plotting (a hybrid GA on U500.10)

3.3 Visualization by Sammon's Mapping

Sammon's mapping is a good visualization tool for multi-dimensional datasets. Local-optimum spaces are also good candidates for Sammon's mapping. Sammon's mapping of the local-optimum space helps visually understanding the problem space. Figure 4(a) shows Sammon's mapping of the local-optimum space for the graph U500.10. The local optima were Sammon-mapped on the XY plane. The Z -axis means the cut size. Figures 4(b) and (c) indicate the projected spaces of Figure 4(a) into XZ plane and YZ plane, respectively. We can observe the fitness landscape with respect to Sammon's mapping. In this case, the result suggests the existence of valleys in more than one place.

4 Visualization of a Steady-State Genetic Search

4.1 Previous Studies

Traditionally, the fitness-generation plotting has been popular for the visualization of genetic process. A great number of papers include these plottings to visualize their genetic search process. Figure 5 shows examples of the traditional plotting.

Recently, Dybowski *et al.* [6] proposed a GA visualization method using Sammon's mapping. There have been a number of studies about GA visualization using Sammon's mapping [4] [15]. They presented initial studies about small problems. An extensive survey of GA visualization techniques appeared in [9]. In this paper, we focus only on the visualization by Sammon's mapping.

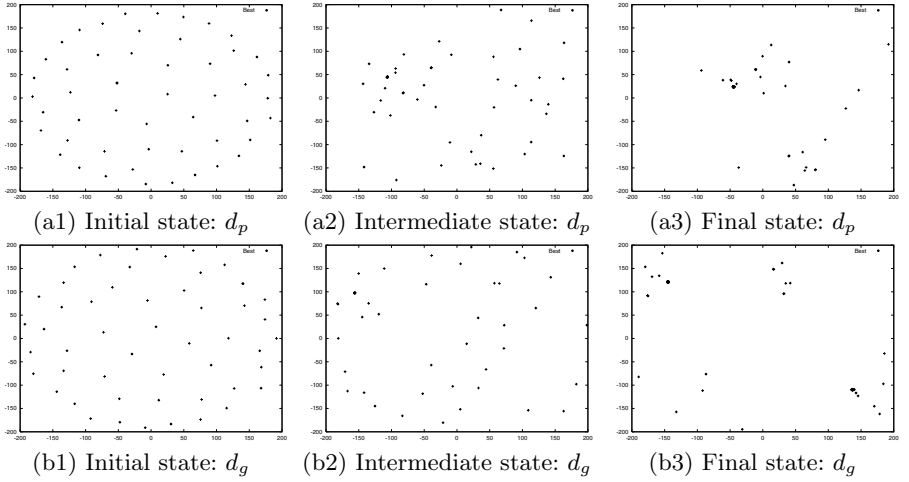


Fig. 6. 2D mapping with different distances (hybrid GBA on U500.10)

4.2 Extended Experiments

We extend the works of Dybowski *et al.* [6]. Using Sammon's mapping, they showed that population converges into one or more clusters. They used the Hamming distance (called the genotype distance in this paper) for the distance in the input space of binary chromosomes.

In this subsection, we provide two experiments for visualization with a GA. First, we make experiments with different distance measures. Then, we provide a new technique for a steady-state GA to visualize the change of population in the genetic search process.

We used the Genetic Bisection Algorithm (GBA) [3] for graph bisection problem. It is a steady-state GA with population size 50, 5-point crossover, adjacent repair⁶, and GENITOR-style replacement [18]. If GBA is hybridized with KL local optimization, it is denoted by KL-GBA. We use KL-GBA on the graph U500.10 for the experiments in this subsection.

Given a space, various distance measures can be defined. The properties of the space are largely dependent on the distance measure. Particularly, in Sammon's mapping, if the output space is a metric space, the input space need to be a metric space to minimize the stress measure. We used two distance measures: the genotype distance d_g and the phenotype distance d_p defined in Section 3. With d_p as the distance measure, Figures 6(a1), (a2), and (a3) show 2-dimensional mapped population spaces at initial, intermediate, and final generation, respectively. Figures 6(b1), (b2), and (b3) show the results with d_g as the distance

⁶ After the crossover, an offspring may not satisfy the balance requirement. It then selects a random point on the chromosome and changes the required number of 1's to 0's (or 0's to 1's) starting at that point on to the right. This adjustment produces some mutation effect.

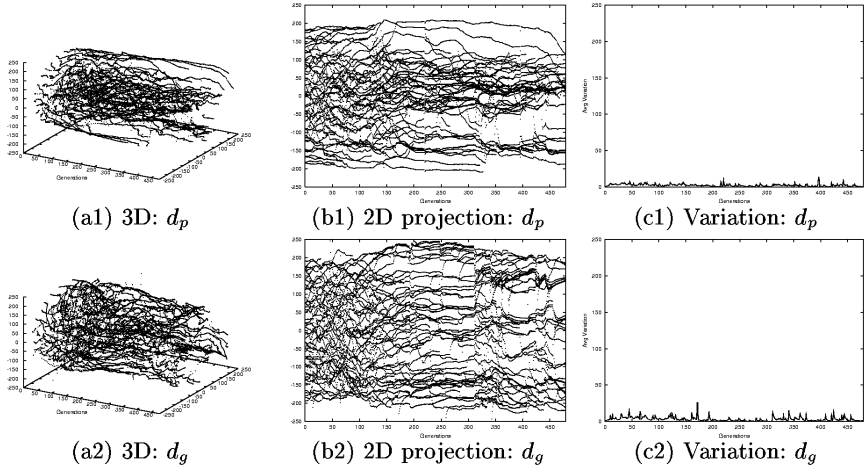


Fig. 7. Visualization of genetic search process with different distances (hybrid GBA on U500.10)

measure. When we used the genotype distance d_g , the population converged into four clusters. However, with the phenotype distance d_p , the population converged into roughly two clusters. From the fact that one phenotype matches two genotypes in graph partitioning, it seems to be reasonable. We observed two notable valleys in this problem space in Section 3.2 and Section 3.3. It is interesting that the population of GA converged into two clusters. The plotting by Sammon's mapping can be extended to three dimensions. We omit the results here.

In the next experiment, we visualize the change of population in the process of a steady-state GA. Generally, Sammon's mapping starts with random initial positions of n objects. Iteratively, it optimizes the stress measure E . A steady-state GA typically generates only one offspring per iteration. It does not make a rapid change per population. Hence, if the positions of the previous generation are used for the initial positions of the next-generation Sammon's mapping, the positions would change steadily over the generations. This makes it possible to visualize solutions over the genetic search process. Figure 7 shows the visualization of a genetic process. Figure 7(a) visualizes the time-varying dataset such that X -axis is the time and YZ plane represents the 2-dimensional Sammon's mapping. Figure 7(b) shows its projection into XZ plane. Figure 7(c) gives the average variation between the previous population positions and the current population positions. More formally, in the i^{th} generation, average variation V_i is defined to be $V_i = \frac{1}{K} \sum_k \|s_k(i) - s_k(i-1)\|$, where $s_k(i)$ is the mapped vector of the k^{th} chromosome in the i^{th} generation and K is the population size. At a generation with large variation, which probably suggests the occurrence of an important solution, the continuity gets broken. From this visualization, we can

also observe the process of population convergence. It helps us to understand the genetic search process more elaborately. It is notable that this visualization is related to average cost plotting of Figure 5(a). In the range with stable average costs, the mapped data also shows minor changes (e.g., see the range [150, 310] in Figure 5(a) and Figure 7). The phenomenon of punctuated equilibria may also be observable by this plotting.

5 Schema Traces

A schema is a pattern of bit strings consisting of specific symbols and asterisks; here, specific symbols represent the pattern and the asterisks represent “don’t care” positions. A genetic algorithm starts with a group of random initial solutions. Of course, the quality of the solutions is low in the early stages of the genetic algorithm. However, most low-quality solutions contain some schemata common to high-quality solutions. The crossover operators of genetic algorithms generate larger schemata by juxtaposition of smaller schemata. It is important to preserve valuable schemata. A schema is prone to be destroyed by crossover operators if the positions forming the schema are scattered.

Generally speaking, it is not easy to know high-quality schemata in a problem. However, for some problems, it is possible to find high-quality schemata. Specially, in Royal Road function [13], all of the desired schemata are given in its description. We can also find high-quality schemata in some instances of graph partitioning. Caterpillar graphs are good examples. It is clear from their Sammon’s mapping (see Figures 1(b) and (c)). In this subsection, we provide the visualization of high-quality schema traces for a graph partitioning problem and a Royal Road function.

First, we compare KL-GBA with BFS-KL-GBA on the graph rcat.994 (see Figure 1(c)). KL-GBA was introduced in Section 4.2. BFS-KL-GBA is an approach proposed in [2] for the purpose of transforming the shapes of valuable schemata to those advantageous for survival by using Breadth-First Search (BFS) reordering. We selected a schema consists of 156 vertices. Figure 8 shows the schema traces in the genetic search process (upper row with KL-GBA and lower row with BFS-KL-GBA). A bold dot represents the presence of the schema in a solution. A bold line mostly means the continual presence of the schema. One can observe remarkable difference between the two algorithms. Not only did KL-GBA show low frequency of schema creation, it also showed a low rate of schema survival. On the contrary, BFS-KL-GBA showed a high rate of schema survival as well as high frequency of schema creation. Since BFS reordering tends to shorten the defining lengths of high-quality schemata, the survival probabilities of those schemata become high through crossovers [3]. Without reordering, despite its early appearance, the schema did not spread all over the population as steadily as the reordered version. One can observe a high rate of schema distinction. On the contrary, the reordered version showed fairly stable preservation of the schema.

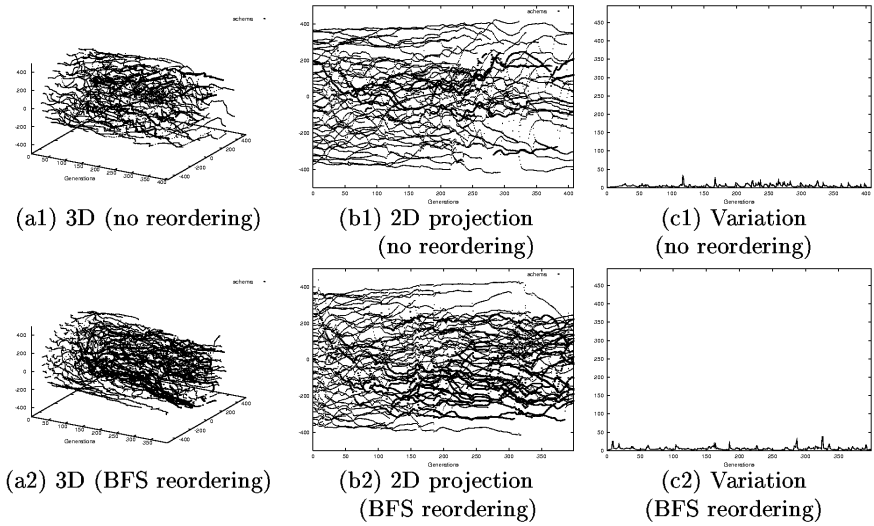


Fig. 8. Schema and reordering (hybrid GBA on rcatt.994)

In the next experiment, we observe the schema traces with a 64-bit Royal Road function. The fitness of a chromosome is determined by the presence of predefined 8th order schemata. It defines a tailor-made fitness landscape for GA's search and provides an ideal laboratory for studying GA's behavior. Dynamics of the search process can be studied by tracing individual schemata. We used a steady-state GA with population size 50, 1-point crossover, 0.5% mutation probability per bit, and GENITOR-style replacement. The Hamming distance is used as the distance measure. Figure 9 shows Sammon's mappings over the generations ((a) and (b)), a traditional plotting (c), and the schema traces ((d), (e), and (f)). Figures 9(a) and (b) clearly reflect strong convergence. It is surprising that the average fitness value nearly reflects the distribution of the population (compare figure (b) with the average line in figure (c)). The second and third rows of Figure 9 show the traces of two low-order high-quality schemata ((d) and (e)) and the high-order schema merging them (f). It visualizes only the individuals containing the schema. Here, a dotted line does not mean the discontinuity of schema presence but usually corresponds to a new appearance of an individual containing the schema. One can observe that schema2 and schema3 first appeared at around 2500th and 1000th generation, respectively, and they were successfully combined to a large schema at around 4500th generation. Although schema3 is appeared earlier than schema2, schema2 spreads over the population faster than schema3. Figures (d), (e), and (f) visualize a process of GA by tracing the lives of particular schemata. Figures (d'), (e'), and (f') are the 2D projections of figures (d), (e), and (f), respectively.

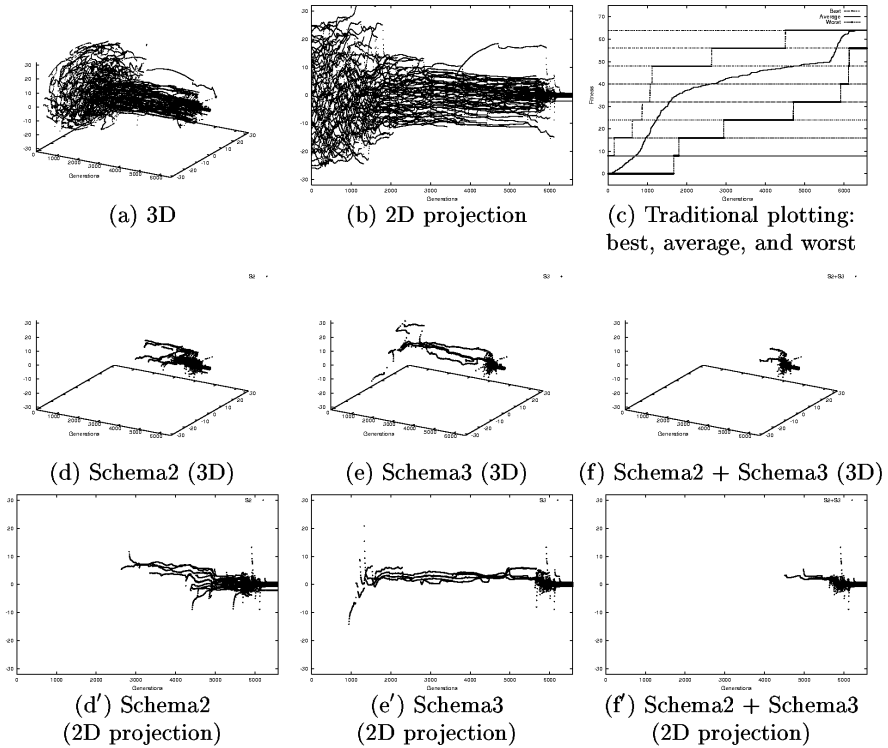


Fig. 9. Royal Road function with 8 schemata

6 Conclusions

Our approach goes beyond those of Boese *et al.* [1] and Dybowski *et al.* [6]. To get insights into fitness landscapes and GA's working mechanism, we introduced visualization techniques using Sammon's mapping and analyzed various experimental results. A steady-state GA for graph partitioning was mainly used in this paper. We could obtain some useful insights from the visualization. They could not be explained by previous visualization experiments. Our approach will be also useful for other optimization problems.

Sammon's mapping is one of the possible mapping methods. We may consider other mapping methods. It is of particular interest as well to investigate the visualization with respect to genetic operators.

Acknowledgments. This work was partly supported by KOSEF through Statistical Research Center for Complex Systems at Seoul National University (SNU) and Brain Korea 21 Project. The RIACT at SNU provided research facilities for this study.

References

1. K. D. Boese, A. B. Kahng, and S. Muddu. A new adaptive multi-start technique for combinatorial global optimizations. *Operations Research Letters*, 15:101–113, 1994.
2. T. N. Bui and B. R. Moon. Hyperplane synthesis for genetic algorithms. In *Fifth International Conference on Genetic Algorithms*, pages 102–109, July 1993.
3. T. N. Bui and B. R. Moon. Genetic algorithm and graph partitioning. *IEEE Trans. on Computers*, 45(7):841–855, 1996.
4. T. D. Collins. Genotypic-space mapping: Population visualization for genetic algorithms. The Knowledge Media Institute, The Open University, Milton Keynes, UK, Technical Report KMI-TR-39, 30th September 1996.
5. D. De Ridder and R.P.W. Duin. Sammon's mapping using neural networks: a comparison. *Pattern Recognition Letters*, 18(11–13):1307–1316, 1997.
6. R. Dybowski, T.D. Collins, and P. Weller. Visualization of binary string convergence by Sammon mapping. In *Fifth Annual Conference on Evolutionary Programming*, pages 377–383, 1996.
7. W. Dzwiniel. How to make Sammon mapping useful for multidimensional data structures analysis. *Pattern Recognition*, 27(7):949–959, 1994.
8. M. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman, San Francisco, 1979.
9. E. Hart and P. Ross. GAVEL – a new tool for genetic algorithm visualization. *IEEE Transactions on Evolutionary Computation*, 5(4):335–348, 2001.
10. J. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, 1975.
11. T. Jones and S. Forrest. Fitness distance correlation as a measure of problem difficulty for genetic algorithms. In *Sixth International Conference on Genetic Algorithms*, pages 184–192, 1995.
12. B. Kernighan and S. Lin. An efficient heuristic procedure for partitioning graphs. *Bell Systems Technical Journal*, 49:291–307, Feb. 1970.
13. M. Mitchell, S. Forrest, and J. H. Holland. The royal road for genetic algorithms: Fitness landscapes and GA performance. In *Toward a Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life*, pages 245–254, Cambridge, MA, 1992. MIT Press.
14. E. Pekalska, D. De Ridder, R.P.W. Duin, and M.A. Kraaijveld. A new method of generalizing Sammon mapping with application to algorithm speed-up. In *Fifth Annual Conference of the Advanced School for Computing and Imaging*, pages 221–228, 1999.
15. H. Pohlheim. Visualization of evolutionary algorithms – set of standard techniques and multidimensional visualization. In *Genetic and Evolutionary Computation Conference*, pages 533–540, 1999.
16. J. W. Sammon, Jr. A non-linear mapping for data structure analysis. *IEEE Transactions on Computers*, 18:401–409, 1969.
17. E. D. Weinberger. Fourier and Taylor series on fitness landscapes. *Biological Cybernetics*, 65:321–330, 1991.
18. D. Whitley and J. Kauth. GENITOR: A different genetic algorithm. In *Rocky Mountain Conference on Artificial Intelligence*, pages 118–130, 1988.

Exploring a Two-Population Genetic Algorithm

Steven Orla Kimbrough¹, Ming Lu¹, David Harlan Wood², and D.J. Wu³

¹ University of Pennsylvania, 3730 Walnut Street, Philadelphia, PA 19104-6340
{kimbrough,milu}@wharton.upenn.edu

² University of Delaware, CIS Dept., Newark, DE 19716
wood@cis.udel.edu

³ DuPree College of Management, Georgia Institute of Technology, Atlanta, GA 30332
wudj@drexel.edu

Abstract. In a two-market genetic algorithm applied to a constrained optimization problem, two ‘markets’ are maintained. One market establishes fitness in terms of the objective function only; the other market measures fitness in terms of the problem constraints only. Previous work on knapsack problems has shown promise for the two-market approach. In this paper we: (1) extend the investigation of two-market GAs to nonlinear optimization, (2) introduce a new, two-population variant on the two-market idea, and (3) report on experiments with the two-population, two-market GA that help explain how and why it works.

1 Introduction

Evolution programs (EPs), and more specifically genetic algorithms (GAs), are optimizing or optimum-seeking procedures. They are used routinely to maximize or minimize an objective function of a number of decision variables. Because the EPs/GAs are general-purpose procedures (aka: weak methods) they can be used in the computationally more challenging cases in which the objective function is nonlinear and/or the decision variables are integers or mixtures of integers and reals. However, when the objective function is constrained by one or more constraints use of GAs (and EPs) is problematic. Considerable attention has been paid to the subject (see [4,9,10,12,16] for excellent reviews and treatments), but no consensus approach has emerged for encoding constrained optimization problems as GAs.

A natural—and the most often used—approach is to penalize the objective function in proportion to the size of the constraint violations. A feasible solution will impose no penalty on the objective function, an infeasible solution will impose an objective function penalty (negative if maximizing, positive if minimizing) as a function of the magnitude of the violation(s) of the constraint(s). Putting this more formally, here is the general form of a maximization constrained optimization problem.

$$\max_{x_i} z(\mathbf{x}), \text{ subject to } E(\mathbf{x}) \geq \mathbf{a}, F(\mathbf{x}) \leq \mathbf{b}, G(\mathbf{x}) = \mathbf{c}, x_i \in \mathcal{S}_i \quad (1)$$

Here, $z(\mathbf{x})$ is the objective function value produced by the candidate solution $\mathbf{x}$.¹ E , F and G each yield zero or more constraint inequalities or equalities. Taken as functions, in the general case z, E, F, G can be any functions at all (on $\mathbf{x}$) and in particular need not be linear. $\mathcal{S}_i$ is the set of permitted values for the $\mathbf{x}_i$ (the components of the vector $\mathbf{x}$), and are often called the *decision variables* for the problem. $\mathcal{S}_i$ may include reals, integers, or mixtures. Problems of the form of expression (1) are not directly translatable into the linear encodings normally used for GA solutions. The purpose of a penalty function formulation is to produce a representation of the problem that can be directly and naturally encoded as a GA. To indicate a penalty function representation, let $\mathbf{x}$ be a (candidate) solution to a maximization constrained optimization problem. Its absolute fitness, $W(\mathbf{x})$, in the presence of penalties for constraint violation is measured as:

$$\max_{x_i} W(\mathbf{x}) = z(\mathbf{x}) - P(\mathbf{x}) \quad (2)$$

where $P(\mathbf{x})$ is the total penalty (if any) associated with constraint violations by $\mathbf{x}$. Problems representable as in expression (2) are directly and naturally encoded as GAs. If a GA finds a solution $\mathbf{x}$ that is feasible ($P(\mathbf{x}) = 0$) and has a high value for $W(\mathbf{x})$, then we may congratulate ourselves on the successful application of the GA heuristic.

Typically, and by design, the penalty imposed on an infeasible solution will severely reduce the net fitness of the solution in question, leading to quick elimination of the solution from the GA population. This may be undesirable and it may be responsible for the generally recognized weak performance of GAs for constrained optimization problems.

Kimbrough et al. [8] have noted that in constrained optimization a GA will drive the population of solutions to the neighborhood of the Pareto frontier.² Under a penalty regime, GA-driven probing of the Pareto frontier is inhibited because infeasible solutions are heavily penalized, leading to likely loss of useful aspects (properties, genetic material) that placed them near the frontier. Armed with this intuition, Kimbrough et al. [8] investigated a “two-market GA” in which a single population of solutions to a constrained optimization problem undergoes two distinct GA procedures. Recalling expression (2), in phase 1 (“optimality improvement”) the population is evaluated with the fitness function $\max_{x_i} z(\mathbf{x})$ and a new generation is created via the usual GA mechanisms. Then, in phase 2

¹ By *candidate solution* or just *solution* we mean any instance of $\mathbf{x}$. Some candidate solutions are feasible and some not. An optimal solution is a candidate solution, which need not be unique, satisfying the expression; that is, it is feasible and no feasible candidate solution yields a better value of $z(\mathbf{x})$.

² Or rather the effective Pareto frontier. While it is often true, it is not true in general that the solution to a constrained optimization problem lies on the Pareto frontier of the constraint set. In this paper when we speak of the Pareto frontier we mean the effective Pareto frontier, that is the set of solutions such that each is feasible (is within the constraint set) and such that no decision variable in the solution can be singly changed so as to improve the objective function and have the solution remain feasible.

Table 1. Summary of Group 1 Problems

Problem	Min/Max	Objective	# Variables	# Linear Inequalities	# Nonlinear Inequalities
1	min	Linear	8	3	3
2	min	Polynomial	7	0	4
3	min	Quadratic	10	3	5
4	min	Polynomial	2	0	2
5	min	Linear	2	0	2
6	min	Polynomial	2	0	2
7	min	Quadratic	2	0	2
8	min	Quadratic	2	1	1
9	max	Nonlinear	20	1	1
10	max	Nonlinear	50	1	1

(“feasibility improvement”) the population is evaluated with the fitness function $\min_{x_i} P(\mathbf{x})$ and a new generation is created in the usual way. The procedure continues, one phase following the other, until a stopping condition is met.

Kimbrough et al. [8] obtained encouraging results for a specialized class of constrained optimization problems (knapsack problems), demonstrating that (as measured by the number of fitness evaluations employed) their two-market GA outperformed generally accepted GA penalty function approaches. Knapsacks are simple, linear integer programming problems. They are known to be NP-hard, yet in practice they have proved tractable and there exist excellent polynomial time heuristics.

The investigations we describe in §2 examined the effectiveness of the two-market GA on a collection of standard nonlinear constrained optimization problems. This is a challenging class of problems, for which improved solution techniques are actively being sought. Further, we introduce a variation of the two-market GA, which we call a *two-population* (and two-market) GA. The two-population GA is essentially equivalent to the single-population, two-market version, but it affords insight into how the GA is exploring the solution space to these constrained optimization problems. We discuss this in some detail in §3.

2 Results

Genocop II/III [11] is a sophisticated, highly-developed product, and may be taken to represent the state of the art for constrained optimization using GAs. We use it as our benchmark for evaluating the performance of our two-market GAs. It is convenient to separate the test problems we studied, and our discussions of them, into distinct groups.

Table 2. Summary of Group 1 Results

Problem	Min/Max	Best Known or Optimal*	Genocop II/III	Two-Market GA		
				Best of 10	Median	Std.
1	min	7049.330923*	7268.650	∅	∅	∅
2	min	680.6300573*	680.640	680.6374	680.7566	0.085123
3	min	24.3062091*	25.883	25.18437	25.61935	0.825277
4	min	0.25*	close	0.25	0.25	0
5	min	-5.5079*	close	-5.50773	-5.50708	0.001042
6	min	-6961.81381*	close	-6960.95	-6957.44	1.908346
7	min	5*	close	5	5	0
8	min	1*	close	1	1	0
9	max	0.80351067	0.80351067	0.80288	0.7888	0.12853
10	max	0.8348	0.83319378	0.809211	0.743844	0.054728

2.1 Group 1

Group 1 contains 10 test problems described by Michalewicz in [10].³ Table 1 summarizes these problems.

Table 2 compares results reported by Michalewicz with results obtained by our two-market, single-population GA. In these experiments we used a floating point representation for real number variables, fitness-proportional selection, arithmetical crossover [10, page 112] with probability 0.4, non-uniform mutation with probability 0.4 [10, page 103].⁴ The results reported for Genocop are for the best solution found from 10 runs. In problems 1–3, the Genocop and the two-market GA runs are for 5,000 generations with a population of 70. (In the two-market GA case, this means 2,500 generations of phase 1 and 2,500 generations of phase 2, alternating). For problems 4–8 Michalewicz reports that Genocop achieved “convergence” to the known optimal solution within 1,000 generations for a population of 70. The two-market GA results reported are also for 1,000 generations—or rather half-generations (500 each, phase 1 and phase 2)—with a population of 70. The Genocop results for problems 9–10 are for 10,000 generations on a population of 70. In the two-market GA, the population size was 50 and the total number of half generations was 10,000 (5,000 phase 1; 5,000 phase 2). We report the best solution found, the median best solution, and the standard deviation of the best solution found over 10 runs. In problem #1, the two-market GA failed to find a feasible solution. We note that Genocop requires a feasible solution at inception; it is not clear how Michalewicz obtained the necessary solution(s) for Genocop. On the remaining problems, the performance of this simple, two-market, single-population GA compares well with Genocop.

³ The correspondence between our numbering and his is, using format (our-number, page-ref, his-identifier): (1, 145, test case #2), (2, 145, test case #3), (3, 146, test case #5), (4, 137, test case #1), (5, 137, test case #2), (6, 138, test case #3), (7, 138, test case #4), (8, 140, test case #5), (9, 156, Keane (n=20)), (10, 156, Keane (n=50)).

⁴ In all our experiments we employed elitest selection [10, page 59].

Table 3. Summary of Group 2 Problems

Problem	Min/Max	Objective	# Variables	# Linear Inequalities	# Nonlinear Inequalities
11 (test11)	max	Nonlinear	2	0	2
12 (chance)	min	Linear	3	2	1
13 (circle)	min	Linear	3	0	10
14 (ex3_1_4)	min	Linear	3	2	1
15 (ex7_3_1)	min	Linear	4	6	1
16 (ex7_3_2)	min	Linear	4	6	1
17 (ex14_1_1)	min	Linear	3	0	4
18 (st_e08)	min	Linear	2	0	2
19 (st_e12)	min	Nonlinear	3	3	0
20 (st_e19)	min	Polynomial	2	1	1
21 (st_e41)	min	Nonlinear	4	0	2

Problems 4–8 are relatively easy, while 1–3 and 9–10 are much more challenging. On problems 9–10, the Genocop results are superior to the two-market GA, while on problems 2–3 Genocop falls behind. We would also draw the reader’s attention to the reasonably low standard deviations for the two-market GA.

2.2 Group 2

Group 2 consists of 11 nonlinear programming problems. Ten are from the GLOBAL Library at GAMS World [6]. The Library is “a web site aiming to bridge the gap between academia and industry by providing highly focused forums and dissemination services in specialized areas of mathematical programming.” As such it is an important source of mathematical programming (constrained optimization) problems that have proved valuable and interesting to the operations research and management science community (cf., INFORMS [5]). Demonstrable success on these problems with evolution programming would cause the INFORMS community to take notice. An eleventh problem, “test11”, appeared in a paper by Schoenaur and Xanthakis [17]. Table 3 summarizes the Group 2 problems, all of which have only inequality constraints (rather than equality constraints), perhaps after algebraic transformation by us.⁵

We obtained the Genocop results by executing Genocop III [11] for 10,000 generations, using the default settings (as recommended), including a population of size 70. The two-market GA was run for 10,000 half generations (5,000 phase 1, 5,000 phase two, alternating), with a population size of 50. Again, we used a floating point representation for real number variables, fitness-proportional selection, arithmetical crossover with probability 0.4, non-uniform mutation with probability 0.4. Because we have not tuned the crossover or mutation rates, and we are using a smaller population, it has to be concluded that the two-market

⁵ We so transformed problems 12 and 19.

Table 4. Summary of Group 2 Results. † = 29.8943939275064, ‡ = **0.324275369**.

Problem	Best Known or Optimal*	Genocop III			Two-Market GA		
		Best of 10	Median	Std.	Best of 10	Median	Std.
11	?	0.115047	0.11504	4.87E-05	0.115047	0.115047	1.12E-07
12	29.8943781591	29.89549	29.94807	0.034976	†	29.90093	0.003931
13	4.57424778502	4.574318	4.575747	0.027615	4.574249	4.574257	9.01E-06
14	-4.0000	-4	-4	0.032482	-4	-4	3.01E-05
15	0.341739553124	0.3558	0.416292	0.141607	‡	0.328268	0.010168
16	1.08986397147	1.09145	1.113644	0.033588	1.089928	1.089985	0.000111
17	0.0000	1.44E-10	1.19E-06	2.85E-06	6.18E-06	0.000172	0.00031
18	?	0.741782	0.741782	9.93E-09	0.741782	0.741782	1.09E-07
19	?	-4.5099	-4.49564	0.02252	-4.50691	-4.49855	0.004872
20	?	-118.705	-118.704	0.021621	-118.705	-118.705	0.001853
21	?	645.626	648.9749	4.398663	641.8242	642.0479	1.827517

GA is not operating at any advantage over Genocop III in these experiments. Table 4 summarizes our results.

Note that for problem 15, the two-market GA has found a solution (shown in **bold**) that is superior to the best solution otherwise known to us.

2.3 Group 2 with a Two-Population GA

The two-market GA introduced and explored for knapsack problems by Kimbrough et al. [8], and further explored by us, is not the only possible form for a two-market GA. There is a *single-population* two-market GA. We introduce here a *two-population* two-market GA.

In the two-population GA, the *feasible population* contains only feasible solutions to the constrained optimization problem to hand, while the *infeasible population* contains only infeasible solutions. The process is simple. The *population-size* parameter is chosen in the customary manner (we use 50). The *initialization-pool* parameter is set (we used 100). At initialization we randomly generate at most *initialization-pool* × *population-size* solutions. When a solution is created we test it for feasibility. If it is feasible and the feasible population is not complete (has less than *population-size* solutions), we add the solution to it. Similarly, if the infeasible population is incomplete and the solution is infeasible, we add it to the infeasible population. After initialization, each population has at most *population-size* solutions or chromosomes. Either population may have fewer, if a sufficient number were not found.

In phase 1 of the two-population (+two-market) GA, the fitnesses of all solutions in the feasible population are calculated based only on the objective function, $z(\mathbf{x})$, and a new generation is created under the chosen genetic regime. The regime is any standard GA with the exceptions that (a) the highest fitness solution is placed directly into the next generation and (b) any infeasible daughter solutions created by the genetic operations are placed in the infeasible

population. A total of *population-size* daughter solutions are created each generation, but the size of the succeeding generation may be considerably smaller.

In phase 2 of the two-population GA, fitnesses of all the solutions in the infeasible population are calculated. (Because of contributions resulting from phase 1, more than *population-size* solutions may be present.) The fitness calculations are based only on the employed penalty function for the constraint set, $P(\mathbf{x})$, and not on the objective function.⁶ The best *population-size* infeasible solutions are candidates to be parents for the next generation. As in phase 1, (a) the highest fitness solution is placed directly into the next generation and (b) any feasible daughter solutions created by the genetic operations are placed in the feasible population.

Phase 1 and 2 are processed in alternation until a stopping condition is reached. Table 5 shows our results with Group 2 using the two-population GA for 10,000 half generations (5,000 phase 1, 5,000 phase 2). Otherwise, the GAs employed are exactly as in the single-population, two-market version discussed above. Note that for problem 15 the two-population GA has found solution

Table 5. Summary of Group 2 Results for a Two-Population GA. † = 29.8943786178232, ‡ = **0.319777729979837**.

Problem	Best Known or Optimal*	Genocop III			Two-Population GA		
		Best of 10	Median	Std.	Best of 10	Median	Std.
11	?	0.115047	0.11504	4.87E-05	0.115047	0.115047	4.16E-08
12	29.8943781591	29.89549	29.94807	0.034976	†	29.91486	0.032017
13	4.57424778502	4.574318	4.575747	0.027615	4.574248	4.574257	6.67E-05
14	-4.0000	-4	-4	0.032482	-4	-3.99991	0.000131
15	0.341739553124	0.3558	0.416292	0.141607	‡	0.320127	0.00033
16	1.08986397147	1.09145	1.113644	0.033588	1.089952	1.089994	2.56E-05
17	0.0000	1.44E-10	1.19E-06	2.85E-06	4.69E-05	6.63E-05	1.36E-05
18	?	0.741782	0.741782	9.93E-09	0.741782	0.741782	5.58E-08
19	?	-4.5099	-4.49564	0.02252	-4.51347	-4.51319	0.000995
20	?	-118.705	-118.704	0.021621	-118.705	-118.705	0.000187
21	?	645.626	648.9749	4.398663	641.8244	641.8283	2.663347

(shown in **bold**) that is superior to the best solution otherwise known to us. Note further that the two-population GA has found a solution⁷ to problem 15 that is superior to the best solution⁸ that the single-population GA found, which itself is better than the best known solution we are otherwise aware of.

⁶ Throughout our two-market runs we used the sum of absolute violations on constraints as our penalty function. We leave for future research, on tuning the two-market GAs, the problem of finding optimal penalty functions.

⁷ $x_1 = 1055.688727, x_2 = 3.360444871, x_3 = 5.040713598, x_4 = 0.319777719$.

⁸ $x_1 = 1053.464266, x_2 = 3.361225273, x_3 = 5.029994467, x_4 = 0.324275369$.

2.4 Summary and Comments on Results

We have investigated a number of constrained optimization problems that are nonlinear in their objective functions or in their constraints (or both). This class of problem is interesting because it arises often in practice and because standard OR/MS solution methods are often defeated by practical problems of this sort. The problems we discuss above are the problems we have investigated. Specifically, none have been left out, e.g., because results were disappointing.⁹

Although it is imperative to study many more problems in this class, these results are broadly quite encouraging. The simple, untuned, and in many ways naïve, single-population, two-market GA of Kimbrough et al. is apparently robust for nonlinear optimization. It compares favorably with the performance of Genocop, an excellent, well-tuned, state-of-the-art conventional GA package. In addition, we have introduced a two-population variant of a two-market GA, and shown that it too performs well (at least on the Group 1 and Group 2 problems we have investigated).

3 Discussion

We conclude with a discussion focusing on understanding why the two-market GAs should perform well for constrained optimization. This ought to be useful in designing future investigations.

Here is an easy question: Why, in a constrained optimization problem attacked with a GA (or EP) should we have *any* penalty associated with constraint violation? The answer, of course, is that the constraints are part of the problem and unless we provide information from them to the GA solving procedure there is no reason to think that the right problem will be addressed. Something has to be done with the constraints.

A more interesting question: Why not impose prohibitive penalties on constraint violation, why not simply eliminate infeasible solutions from consideration? Answer: it's been tried and doesn't work very well.¹⁰ Why not? Expanding on suggestions in Kimbrough et al. [8], we should think of the GA as creating a population of solutions that is driven towards and then explores the Pareto frontier¹¹ of the constrained optimization problem.

During this stochastic exploration process solutions will inevitably be created that “step over the line” from the feasible region to infeasibility. Are these

⁹ More carefully, we are not reporting experience with nonlinear problems having equality constraints for which it is not straightforward to eliminate a variable and thereby convert to an inequality constraint. We examined three such problems with mixed results.

¹⁰ Repair of infeasible solutions is a more modest variant of this approach and it sometimes does work reasonably well. We defer discussion of repair to future work. We note further that the “death penalty” method, which liquidates solutions found to be infeasible, has been reported to perform poorly for constrained optimization problems, e.g., [10].

¹¹ See earlier comments qualifying this terminology.

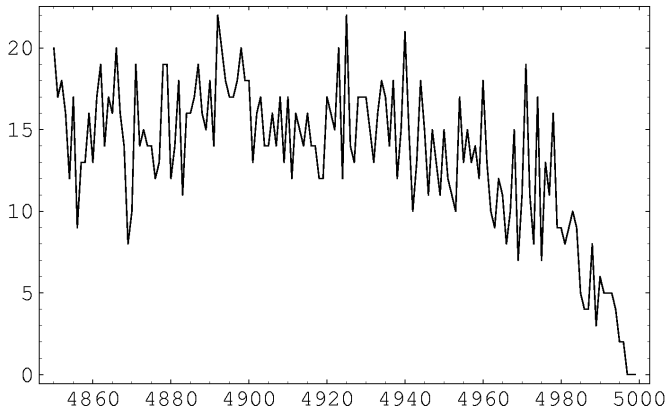


Fig. 1. For problem ex7_3_2 with population size 50: count of number of infeasible solutions created from the feasible population, by generation, 4850–4999.

solutions now worthless? Surely not, one would conjecture. Even a randomly generated pool of solutions will typically contain useful genetic material, which after recombination, mutation, etc. may prove valuable indeed. One would expect that infeasible solutions, descending from feasible parents and pressured by selection to minimize infeasibility, might have a fruitfully dense presence of useful genetic material. The two-market GAs, in both the single- and double-population forms, present ways of preserving genetic material in infeasible solutions so that it might be used by future generations.

The two-population GA is convenient for studying the exchange dynamics between feasible and infeasible solutions. For the two-population GA we described above and for the problems we applied it to, we found a robust pattern of behavior. First, of the 50 daughter solutions created from the feasible population each generation, roughly 10–20 are infeasible and get put into the infeasible population. This number tends to tail off to nearly zero as the population gets very close to finding an optimal solution. See Figure 1 for a representative example. Second, of the 50 daughter solutions created from the infeasible population each generation, roughly 0–2 are feasible and get put into the feasible population. This number typically increases, roughly to 0–4 then 2–8, as the number of generations increases beyond 5,000. See Figure 2 for a representative example. These two findings indicate that the infeasible population is steadily and increasingly sending genetic material to the feasible population. Our third finding confirms this. If we track for each feasible solution whether it has an infeasible ancestor we find that very quickly, before 20 generations have elapsed, *every* solution in the feasible population is descended from one infeasible solution or another! See Figure 3 for a representative example.

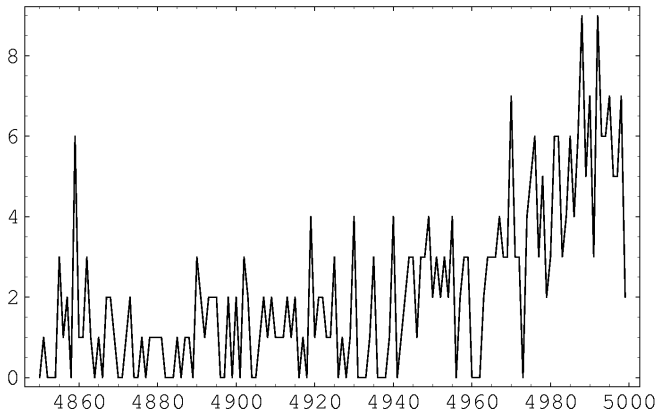


Fig. 2. For problem ex7.3.2 with population size 50: count of number of feasible solutions created from the infeasible population, by generation, 4850–4999.

These findings help us understand the dynamics of constrained optimization with GAs. Yet as good findings should, they raise more questions than they answer. How can these simple two-market GAs (both single- and double-population GAs) be improved and tuned? How broadly will they work? By characterizing the structure of the effective Pareto frontier can we gain useful information for configuring the GAs? How does the richness of the genetic material in the infeasible population vary over generations and how does it compare to that in the feasible population? What are the most effective forms of penalty functions? Should scaling be introduced on constraints? There will be an exchange rate in computational value between maintaining an incrementally larger feasible population and an incrementally larger infeasible population. At some point—it would appear from our results—an extra infeasible solution is worth more for optimization than an extra feasible solution. When does this happen? Specifically, we have maintained identical target sizes for both the feasible and the infeasible populations. Is this the right tradeoff or might it be improved? What can be done to maximize, or even diagnose, the value of the genetic material in the infeasible population? And so forth.

In addition, GAs and EPs are members of a larger family of *metaheuristics* for constrained optimization [13,15,18]. This larger family includes simulated annealing, neural networks, tabu search, extremal optimization [2], and other forms of local search heuristics. We are especially intrigued by parallels in the concepts that different heuristics attempt to instantiate. For example crossover in GAs and annealing in simulated annealing arguably are parallel approaches to avoid greedy hill-climbing in favor of a less myopic form of local search. The underlying concept for both forms of the two-market GA is to maintain a diversity of useful partial solutions (genetic material) and to facilitate subsequent use of this information. A tantalizing parallel concept, called *demon algorithms*,

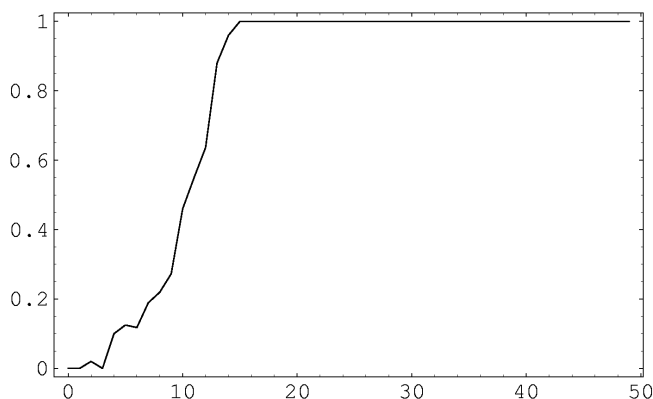


Fig. 3. For problem ex7_3.2 with population size 50: fraction, by generation, of feasible individuals with an infeasible ancestor.

has recently been brought to our attention. (There are various forms; see [3] for an overview.) Demons act rather like lines of credit for a search process. If the process is minimizing, it is always permitted to go down hill without drawing on its line of credit (its demon). Faced with a solution that would take the process up hill, however, it may draw on its available line of credit and undertake a local upwards climb. The draw may be deterministic or stochastic, depending on the variety of algorithm in play. Convergence may be realized by gradually reducing the available credit.

We see demon algorithms as motivated by an intuition or concept similar to that underlying the two-market GAs. Related ideas appear elsewhere, e.g. [14]. What other forms might this concept take? How is it best employed and when will work or not work well? Our two-population GA would seem to be a GA analog of primal-dual algorithms [1,7], with two bodies of solutions—one primal feasible, the other dual feasible—being driven towards each other. These and many other questions are now on the table, due in part to the encouraging results obtained, here and earlier, with two-market GAs.

References

1. Mokhtar S. Bazaraa, Hanif D. Sherali, and C. M. Shetty. *Nonlinear Programming: Theory and Algorithms*. John Wiley & Sons, Inc., New York, NY, 1993.
2. Stefan Boettcher and Allon G. Percus. Extremal optimization: An evolutionary local-search algorithm. In Hemant K. Bhargava and Nong Ye, editors, *Computational Modeling and Problem Solving in the Networked World: Interfaces in Computer Science and Operations Research*, Operations Research/Computer Science Interface Series, pages 61–77. Kluwer, Boston, MA, 2003.

3. Bala Chandran, Bruce Golden, and Edward Wasil. A computational study of three demon algorithm variants for solving the traveling salesman problem. In Hemant K. Bhargava and Nong Ye, editors, *Computational Modeling and Problem Solving in the Networked World: Interfaces in Computer Science and Operations Research*, Operations Research/Computer Science Interface Series, pages 155–76. Kluwer, Boston, MA, 2003.
4. Carlos Artemio Coello Coello. A survey of constraint handling techniques used with evolutionary algorithms. Technical report Lania-RI-99-04, Laboratorio Nacional de Informática Avanzada, Veracruz, México, 1999.
<http://www.lania.mx/~ccoello/constraint.html>.
5. Institute for Operations Research and Management Science. Informs online. Pages on the World Wide Web, Accessed 2003-01-19. URL: <http://www.informs.org>.
6. GAMS World. Global world. Pages on the World Wide Web, Accessed 2003-01-19. URLs: <http://www.gamsworld.org/global/index.htm>,
<http://www.gamsworld.org/global/globallib.htm>.
7. Arthur M. Geoffrion. Duality in nonlinear programming: A simplified applications-oriented development. *SIAM Review*, 13(1):1–37, January 1971.
8. Steven O. Kimbrough, Ming Lu, David Harlan Wood, and D. J. Wu. Exploring a two-market genetic algorithm. In W. B. Langdon, E. Cantú Paz, and et al., editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2002)*, pages 415–21, San Francisco, CA, 2002. Morgan Kaufmann Publishers.
9. Zbigniew Michalewicz. A survey of constraint handling techniques in evolutionary computation methods. In *Proceedings of the 4th Annual Conference on Evolutionary Programming*, pages 135–155, Cambridge, MA, 1995. MIT Press.
<http://www.coe.uncc.edu/~zbyszek/papers.html>.
10. Zbigniew Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer, Berlin, Germany, third edition, 1996.
11. Zbigniew Michalewicz. Genocop – optimization via genetic algorithms. World Wide Web, Accessed January 2003.
<http://www.cs.sunysb.edu/~algorithm/implement/genocop/implement.shtml>.
12. Zbigniew Michalewicz and David B. Fogel. *How to Solve It: Modern Heuristics*. Springer, Berlin, Germany, 2000.
13. I. H. Osman and J. P. Kelly, editors. *Meta-Heuristics: Theory and Application*. Kluwer, Boston, MA, 1996.
14. Carlos-Andres Pena-Reyes and Moshe Sipper. Fuzzy CoCo: Balancing accuracy and interpretability of fuzzy models by means of coevolution. *IEEE Transactions on Fuzzy Systems*, 9(5):727–737, October 2001.
15. Colin R. Reeves. *Modern Heuristic Techniques for Combinatorial Problems*. John Wiley & Sons, New York, NY, 1993.
16. Ruhul Sarker, Masoud Mohammadian, and Xin Yao, editors. *Evolutionary Optimization*. Kluwer, Boston, MA, 2002.
17. M. Schoenauer and S. Xanthakis. Constrained GA optimization. In *Proceedings of the 5th International Conference on Genetic Algorithms*, pages 573–80. Morgan Kaufmann, 1993.
18. S. Voß. Metaheuristics: The state of the art. In A. Nareyek, editor, *Local Search for Planning and Scheduling*, volume 2148 of *Lecture Notes in Computer Science*, pages 1–23. Springer, Heidelberg, Germany, 2001.

Adaptive Elitist-Population Based Genetic Algorithm for Multimodal Function Optimization

Kwong-Sak Leung and Yong Liang

Department of Computer Science & Engineering,
The Chinese University of Hong Kong, Shatin, N.T., Hong Kong
{ksleung, yliang}@cse.cuhk.edu.hk

Abstract. This paper introduces a new technique called adaptive elitist-population search method for allowing unimodal function optimization methods to be extended to efficiently locate all optima of multimodal problems. The technique is based on the concept of adaptively adjusting the population size according to the individuals' dissimilarity and the novel elitist genetic operators. Incorporation of the technique in any known evolutionary algorithm leads to a multimodal version of the algorithm. As a case study, genetic algorithms(GAs) have been endowed with the multimodal technique, yielding an adaptive elitist-population based genetic algorithm(AEGA). The AEGA has been shown to be very efficient and effective in finding multiple solutions of the benchmark multimodal optimization problems.

1 Introduction

Interest in the multimodal function optimization is expanding rapidly since real-world optimization problems often require the location of multiple optima in the search space. Then the designer can use other criteria and his experiences to select the best design among generated solutions. In this respect, evolutionary algorithms(EAs) demonstrate the best potential for finding more of the best solutions among the possible solutions because they are population-based search approach and have a strong optimization capability. However, in the classic EA search process, all individuals, which may locate on different peaks, eventually converge to one peak due to genetic drift. Thus, standard EAs generally only end up with one solution. The genetic drift phenomenon is even more serious in EAs with the elitist strategy, which is a widely adopted method to improve EAs' convergence to a global optimum of the problems.

Over the years, various population diversity mechanisms have been proposed that enable EAs to maintain a diverse population of individuals throughout its search, so as to avoid convergence of the population to a single peak and to allow EAs to identify multiple optima in a multimodal domain. However, various current population diversity mechanisms have not demonstrated themselves to be very efficient as expected. The efficiency problems, in essence, are related to

some fundamental dilemmas in EAs implementation. We believe any attempt of improving the efficiency of EAs has to compromise these dilemmas, which include:

- *The elitist search versus diversity maintenance dilemma:* EAs are also expected to be global optimizers with unique global search capability to guarantee exploration of the global optimum of a problem. So the elitist strategy is widely adopted in the EAs search process. Unfortunately, the elitist strategy concentrates on some “super” individuals, reduces the diversity of the population, and in turn leads to the premature convergence.
- *The algorithm effectiveness versus population redundancy dilemma:* For many EAs, we can use a large population size to improve their effectiveness including a better chance to obtain the global optimum and the multiple optima for a multimodal problem. However, the large population size will notably increase the computational complexity of the algorithms and generate a lot of redundant individuals in the population, thereby decrease the efficiency of the EAs.

Our idea in this study is to strike a tactical balance between the two contradictory issues of the two dilemmas. We propose a new adaptive elitist-population search technique to identify and search multiple peaks efficiently in multimodal problems. We incorporate the technique in genetic algorithms (GAs) as a case study, yielding an adaptive elitist-population based genetic algorithm (AEGA).

The next section describes the related work relevant to our proposed technique. Section 3 introduces the adaptive elitist-population search technique and describes the implementation of the algorithm. Section 4 presents the comparison of our results with other multimodal evolutionary algorithms. Section 5 draws some conclusion and proposes further directions of research.

2 Related Work

In this section we briefly review the existing methods developed to address the related issues: elitism, niche formation method, and clonal selection principle of an artificial immune network.

2.1 Elitism

It is important to prevent promising individuals from being eliminated from the population during the application of genetic operators. To ensure that the best chromosome is preserved, elitist methods copy the best individual found so far into the new population [4]. Different EAs variants achieve this goal of preserving the best solution in different ways, e.g. GENITOR [8] and CHC [2]. However, “elitist strategies tend to make the search more exploitative rather than explorative and may not work for problems in which one is required to find multiple optimal solutions” [6].

2.2 Evolving Parallel Subpopulations by Niching

Niching methods extend EAs to domains that require the location and maintenance of multiple optima. Goldberg and Richardson [1] used Holland's sharing concept [3] to divide the population into different subpopulations according to similarity of the individuals. They introduced a sharing function that defines the degradation of the fitness of an individual due to the presence of neighboring individuals. The sharing function is used during selection. Its effect is such that when many individuals are in the same neighborhood they degrade each other's fitness values, thus limiting the uncontrolled growth of a particular species.

Another way of inducing niching behavior in a EAs is to use crowding methods. Mahfoud [7] improved standard crowding of De Jong [4], namely deterministic crowding, by introducing competition between children and parents of identical niche. Deterministic crowding works as follows. First it groups all population elements into $n/2$ pairs. Then it crosses all pairs and mutates the offspring. Each offspring competes against one of the parents that produced it. For each pair of offspring, two sets of parent-child tournaments are possible. Deterministic crowding holds the set of tournaments that forces the most similar elements to compete. Similarity can be measured using either genotypic or phenotypic distances. But deterministic crowding fails to maintain diversity when most of the current populations have occupied a certain subgroup of the peaks in the search process.

2.3 Clonal Selection Principle

The clonal selection principle is used to explain the basic features of an adaptive immune response to an antigenic stimulus. This strategy suggests that the algorithm performs a greedy search, where single members will be optimized locally (exploitation of the surrounding space) and the newcomers yield a broader exploration of the search space. The population of clonal selection includes two parts. First one is the clonal part. Each individual will generate some clonal points and select best one to replace its parent, and the second part is the newcomer part, the function of which is to find new peaks. Clonal selection algorithm also incurs expensive computational complexity to get better results of the problems [5].

All the techniques found in the literature try to give all local or global optimal solutions an equal opportunity to survive. Sometimes, however, survival of low fitness but very different individuals may be as, if not more, important than that of some highly fit ones. The purpose of this paper is to present a new technique that addresses this problem. We show that using this technique, a simple GA will converge to multiple solutions of a multimodal optimization problem.

3 Adaptive Elitist-Population Search Technique

Our technique for the multimodal function maximization presented in this paper achieves adaptive elitist-population searching by exploiting the notion of the

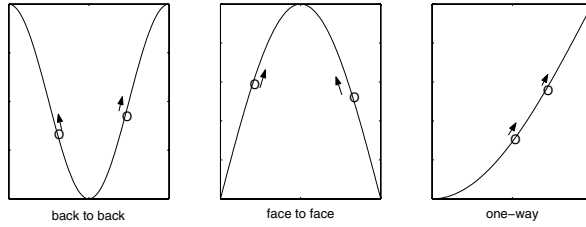


Fig. 1. The relative ascending direction of both individuals being considered: back to back, face to face and one-way.

relative ascending directions of both individuals (and for a minimization problem this direction is called relative descending direction).

For a high dimension maximization problem, every individual generally has many ascending directions. But along the line, which is uniquely defined by two individuals, each individual only has one ascending direction, called the relative ascending direction toward the other one. Moreover, the relative ascending directions of both individuals only have three probabilities: back to back, face to face and one-way (Fig.1). The individuals located in different peaks are called dissimilar individuals. We can measure the dissimilarity of the individuals according to the composition of their relative ascending directions and their distance. The distance between two individuals $x_i = (x_{i1}, x_{i2}, \dots, x_{in})$ and $x_j = (x_{j1}, x_{j2}, \dots, x_{jn})$ is defined by:

$$d(x_i, x_j) = \sqrt{\sum_{k=1}^n (x_{ik} - x_{jk})^2} \quad (1)$$

In this paper we use the above definition of distance, but the method we described will work for other distance definitions as well.

3.1 The Principle of the Individuals' Dissimilarity

Our definition of the principle of the individuals' dissimilarity, as well as the operation of the AEGA, depends on the relative ascending directions of both individuals and a parameter we call the distance threshold, which denoted by σ_s . The principle to measure the individuals' dissimilarity is demonstrated as follows:

- If the relative ascending directions of both individuals are back to back, these two individuals are dissimilar and located on different peaks;
- If the relative ascending directions of both individuals are face to face or one-way, and the distance between two individuals is smaller than σ_s , these two individuals are similar and located on the same peak.

In niching approach, the distance between two individuals is the only measurement to determine whether these two individuals are located on the same peak,

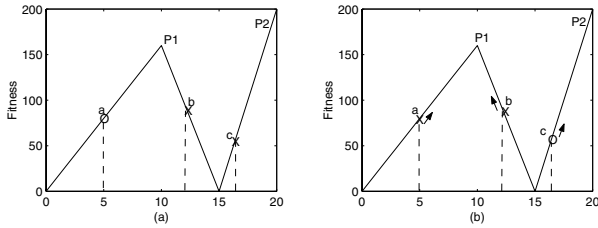


Fig. 2. Determining subpopulations by niching method and the relative ascending directions of the individuals.

but this is often not accurate. Suppose, for example, that our problem is to maximize the function shown in Fig.2. P_1 and P_2 are two maxima and assume that, in a particular generation, the population of the GA consists of the points shown. The individuals a and b are located on the same peak, and the individual c is on another peak. According to the distance between two individuals only, the individuals b and c will be put into the same subpopulation, and the individual a into another subpopulation (Fig.2-(a)). Since the fitness of c is smaller than that of b , the probability of c surviving to the next generation is low. This is true even for a GA using fitness sharing, unless a sharing function is specifically designed for this problem. However, the individual c is very important to the search, if the global optimum P_2 is to be found. Applying our principle, the relative ascending directions of both individuals b and c are back to back, and they will be considered to be located on different peaks (Fig.2-(b)). Identifying and preserving the “good quality” of individual c is the prerequisite for genetic operators to maintain the diversity of the population. We propose to solve the problems by using our new elitist genetic operators described below.

3.2 Adaptive Elitist-Population Search

The goal of the adaptive elitist-population search method is to adaptively adjust the population size according to the features of our technique to achieve:

- a single elitist individual searching for each peak; and
- all the individuals in the population searching for different peaks in parallel.

For satisfying multimodal optimization search, we define the elitist individuals in the population as the individuals with the best fitness on different peaks of the multiple domain. Then we design the elitist genetic operators that can maintain and even improve the diversity of the population through adaptively adjusting the population size. Eventually the population will exploit all optima of the multimodal problem in parallel based on elitism.

Elitist Crossover Operator: The elitist crossover operator is composed based on the individuals’ dissimilarity and the classical crossover operator. Here we

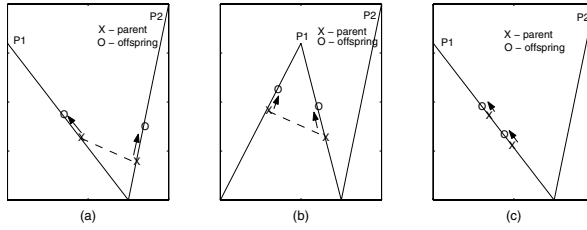


Fig. 3. A schematic illustration that the elitist crossover operation.

have chosen the random uniformly distributed variable to perform crossover (with probability p_c), so that the offspring c_i and c_j of randomly chosen parents p_i and p_j are:

$$\begin{aligned} c_i &= p_i \pm \mu_1 \times (p_i - p_j) \\ c_j &= p_j \pm \mu_2 \times (p_i - p_j) \end{aligned} \quad (2)$$

where μ_1, μ_2 are uniformly distributed random numbers over $[0, 1]$ and the signs of μ_1, μ_2 are determined by the relative directions of both p_i and p_j .

The algorithm of the elitist crossover operator is given as follows:

Input: g —number of generations to run, N —population size
Output: P_g —the population to next generation

```

for  $t \leftarrow 1$  to  $N/2$  do
   $p_i \leftarrow$  random select from population  $P_g(N)$ ;
   $p_j \leftarrow$  random select from population  $P_g(N - 1)$ ;
  determine the relative directions of both  $p_i$  and  $p_j$ ;
  if back to back then  $\mu_1 < 0$  and  $\mu_2 < 0$ ;
  if face to face then  $\mu_1 > 0$  and  $\mu_2 > 0$ ;
  if one-way then  $\mu_1 > 0, \mu_2 < 0$  or  $\mu_1 < 0, \mu_2 > 0$ ;
   $c_i \leftarrow p_i + \mu_1 \times (p_i - p_j)$ ;
   $c_j \leftarrow p_j + \mu_2 \times (p_i - p_j)$ ;
  if  $f(c_1) > f(p_1)$  then  $p_1 \leftarrow c_1$ ;
  if  $f(c_2) > f(p_2)$  then  $p_2 \leftarrow c_2$ ;
  if the relative directions of  $p_1$  and  $p_2$  are face to face or one-way, and
     $d(p_1, p_2) < \sigma_s$ , then
    if  $f(p_1) > f(p_2)$  then  $P_g \leftarrow P_g/p_2$  and  $N \leftarrow N - 1$ ;
    if  $f(p_2) > f(p_1)$  then  $P_g \leftarrow P_g/p_1$  and  $N \leftarrow N - 1$ ;
end for

```

As shown above, through determining the signs of the parameters μ_1 and μ_2 by the relative directions of both p_1 and p_2 , the elitist crossover operator generates the offspring along the relative ascending direction of its parents (Fig.3), thus the search successful rate can be increased and the diversity of the population be maintained. Conversely, if the parents and their offspring are determined to be on the same peak, the elitist crossover operator could select the elitist to be

retained by eliminating all the redundant individuals to increase the efficiency of the algorithm.

Elitist Mutation Operator: The main function of the mutation operator is finding a new peak to search. However, the classical mutation operator cannot satisfy this requirement well. As shown in Fig.4-(a), the offspring is located on a new peak, but since its fitness is not better than its parent, so it is difficult to be retained, and hence the new peak cannot be found by this mutation operation. We design our elitist mutation operator for solving this problem based on any mutation operator, but the important thing is to determine the relative directions of the parent and the child after the mutation operation. Here we use the uniform neighborhood mutation (with probability p_m):

$$c_i = p_i \pm \lambda \times r_m \quad (3)$$

where λ is a uniformly distributed random number over $[-1, 1]$, r_m defines the mutation range and it is normally set to $0.5 \times (b_i - a_i)$, and the $+$ and $-$ signs are chosen with a probability of 0.5 each.

The algorithm of the elitist mutation operator is given as follows:

Input: g —number of generations to run, N —population size
Output: P_g —the population to next generation

for $t \leftarrow 1$ **to** N **do**
 $c_t \leftarrow p_t + \lambda \times r_m$;
 determine the relative directions of both p_t and c_t ;
 if face to face or one-way and $f(c_t) > f(p_t)$ **then**
 $p_t \leftarrow c_t$ and **break** ;
 else if back to back, **then**
 for $s \leftarrow 1$ **to** $N - 1$ **do**
 if $[d(p_s, c_t) < \sigma_s]$ and $[f(p_s) > f(c_t)]$ **then break**;
 else $P_g \leftarrow P_g \cup \{c_t\}$ and $N \leftarrow N + 1$;
 end for
 end if
end for

As shown above, if the direction identification between the parent and offspring demonstrates that these two points are located on different peaks, the parent is passed on to the next generation and its offspring is taken as a new individual candidate. If the candidate is on the same peak with another individual, the distance threshold σ_s will be checked to see if they are close enough for fitness competition for survival. Accordingly, in Fig.4-(a), the offspring will be conserved in the next generation, and in Fig.4-(b), the offspring will be deleted. Thus, the elitist mutation operator can improve the diversity of the population to find more multiple optima.

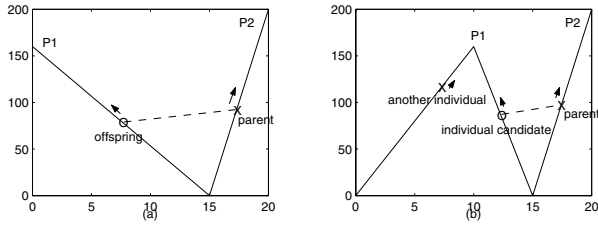


Fig. 4. A schematic illustration that the elitist mutation operation.

3.3 The AEGA

In this section, we will present the outline of the Adaptive Elitist-population based Genetic Algorithm(AEGA). Because our elitist crossover and mutation operators can adaptively adjust the population size, our technique completely simulates the “survival for the fittest” principle without any special selection operator. On the other hand, since the population of AEGA includes most of the elitist individuals, a classical selection operator could copy some individuals to the next generation and deleted others from the population, thus the selection operator will decrease the diversity of the population, increase the redundancy of the population, and reduce efficiency of the algorithm. Hence, we design the AEGA without any special selection operator. The pseudocode for AEGA is shown bellow. We can see that the AEGA is a single level parallel (individuals) search algorithm same as the classical GA, but the classical GA is to search for single optimum. The niching methods is a two-level parallel (individuals and subpopulations) search algorithms for multiple optima. So in terms of simplicity in the algorithm structure, the AEGA is better than the other EAs for multiple optima.

The structure of the AEGA:

```

begin
   $t \leftarrow 0$ ;
  Initialize  $P(t)$ ;
  Evaluate  $P(t)$ ;
  while (not termination condition) do
    Elitist crossover operation  $P(t+1)$ ;
    Elitist mutation operation  $P(t+1)$ ;
    Evaluate  $P(t+1)$ ;
     $t \leftarrow t+1$ ;
  end while
end

```

4 Experimental Results

The test suite used in our experiments include those multimodal maximization problems listed in Table 1. These types of functions are normally regarded as

difficult to be optimized, and they are particularly challenging to the applicability and efficiency of the multimodal evolution algorithms. Our experiments of multimodal problems were divided into two groups with different purpose. We report the results of each group below.

Table 1. The test suite of multimodal functions used in our experiments.

Deb's function(5 peaks):
$f_1(x) = \sin^6(5\pi x), x \in [0, 1];$
Deb's decreasing function(5 peaks):
$f_2(x) = 2^{-2((x-0.1)/0.9)^2} \sin^6(5\pi x), x \in [0, 1];$
Roots function(6 peaks):
$f_3(x) = \frac{1}{1+ x^6-1 },$ where $x \in C, x = x_1 + ix_2 \in [-2, 2];$
Multi function(64 peaks):
$f_4(x) = x_1 \sin(4\pi x_1) - x_2 \sin(4\pi x_2 + \pi) + 1; x_1, x_2 \in [-2, 2].$

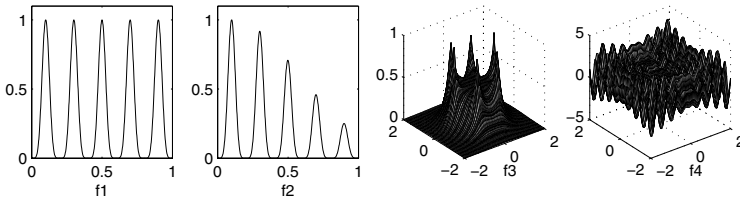


Fig. 5. The suite of the multimodal test function.

Explanatory Experiments: This group of experiments on $f_3(x)$ aims to exhibit the evolution details (particularly, the adaptive adjusting population size details) of the AEGA for 2-D case, and also to demonstrate the parameter control of the search process of the AEGA. In applying the AEGA to solve the 2-D problem $f_3(x)$, we set the initial population size $N = 2$ and the distance threshold $\sigma_s = 0.4$ or 2 .

Fig.6 demonstrate clearly how the new individual are generated on a newly discovered peak and how the elitist individuals reach each optimum in the multimodal domain. Fig.6-(a) shows the 2 initial individuals, when $\sigma_s = 0.4$, the population size is increased to 8 at the 50th generation(Fig.6-(b)), and Fig.6-(c) show the 4 individuals in the population at the 50th generation when $\sigma_s = 2$. When σ_s is smaller, new individuals are generated more easily. At the 50th generation, the result of $\sigma_s = 0.4$ seems to be better; but finally, both settings of the AEGA can find all the 6 optima within 200 generations (Fig.6-(d)). This means the change of the distance threshold does not necessarily influence the efficiency of the AEGA.

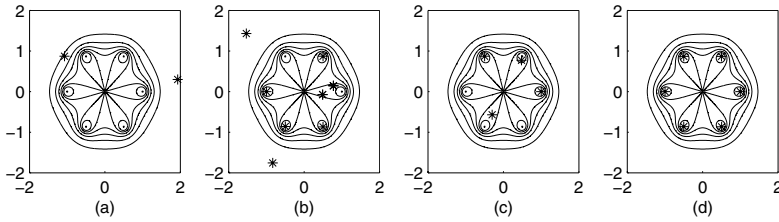


Fig. 6. A schematic illustration that the AEGA to search on the Roots function, (a) the initial population; (b) the population at 50th generation ($\sigma_s = 0.4$); (c) the population at 50th generation ($\sigma_s = 2$); (d) the final population at 200th generation ($\sigma_s = 0.4$ and 2).

Comparisons: To assess the effectiveness and efficiency of the AEGA, its performance is compared with the fitness sharing, determining crowding and clonal selection algorithms. The comparisons are made in terms of the solution quality and computational efficiency on the basis of applications of the algorithms to the functions $f_1(x) - f_4(x)$ in the test suite. As each algorithm has its associated overhead, a time measurement was taken as a fair indication of how effectively and efficiently each algorithm could solve the problems. The solution quality and computational efficiency are therefore respectively measured by the number of multiple optima maintained and the running time for attaining the best result by each algorithm. Unless mentioned otherwise, the time is measured in seconds as measured on the computer.

Tables 2 lists the solution quality comparison results in terms of the numbers of multiple optima maintained when the AEGA and other three multimodal algorithms are applied to the test functions, $f_1(x) - f_4(x)$. We have run each algorithms 10 times. We can see, each algorithm can find all optima of $f_1(x)$. In the AEGA, two initial individuals increase to 5 individuals and find the 5 multiple optima. For function $f_2(x)$, crowding algorithm cannot find all optima for each time. For function $f_3(x)$, crowding cannot get any better result. Sharing and clonal algorithms need to increase the population size for improving their performances. The AEGA still can use two initial individuals to find all multiple optima. For function $f_4(x)$, crowding, sharing and clonal algorithms cannot get any better results, but the successful rate of AEGA for finding all multiple optima is higher than 99%. Figs.7 and 8 show the comparison results of the AEGA and the other three multimodal algorithms for $f_1(x)$ and $f_2(x)$, respectively. The circles and stars represent the initial populations of AEGA and the final solutions respectively. In the AEGA process, we have only used 2 individuals in the initial population. In 200 generations, finally the 5 individuals in the population can find the 5 multiple optima. These clearly show why the AEGA is significantly more efficient than the other algorithms. On the other hand, the computational efficiency comparison results are also shown in Tables 2. It is clear from these results that the AEGA exhibits also a very significant outperformance of many orders compared to the three algorithms for all test

functions. All these comparisons show the superior performance of the AEGA in efficacy and efficiency.

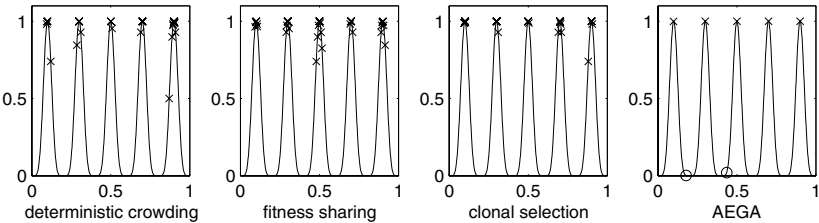


Fig. 7. A schematic illustration that the results of the algorithms for $f_1(x)$.

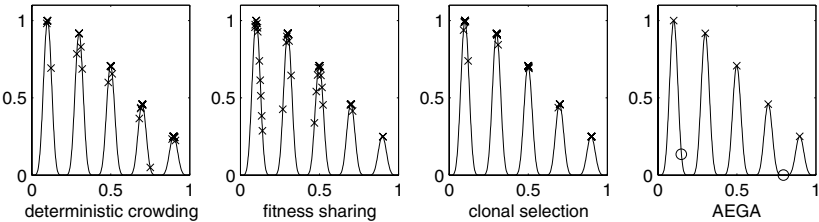


Fig. 8. A schematic illustration that the results of the algorithms for $f_2(x)$.

Table 2. Comparison of results of the algorithms for $f_1(x) - f_4(x)$.

Algorithms	Initial population				Generations				Nb of Peaks				Running time(s)			
	f_1	f_2	f_3	f_4	f_1	f_2	f_3	f_4	f_1	f_2	f_3	f_4	f_1	f_2	f_3	f_4
Deterministic Crowding	100	100	500	1000	20000	20000	20000	20000	5	4.8	3.2	13.2	6.09	6.14	51.5	121
Fitness Sharing	100	100	500	500	200	200	2000	5000	5	5	5.1	18.7	2.44	2.45	41.2	236
Clonal Selection	50	50	200	500	200	200	1000	5000	5	5	5.8	30.1	1.10	1.12	37.1	187
AEGA	2	2	2	50	200	200	200	5000	5	5	6	63.7	0.01	0.01	0.11	9.4

5 Conclusion and Future Work

In this paper we have presented the adaptive elitist-population search method, a new technique for evolving parallel elitist individuals for multimodal function optimization. The technique is based on the concept of adaptively adjusting the population size according to the individuals' dissimilarity and the elitist genetic operators.

The adaptive elitist-population search technique can be implemented with any combinations of standard genetic operators. To use it, we just need to introduce one additional control parameter, the distance threshold, and the population size is adaptively adjusted according to the number of multiple optima. As an example, we have endowed genetic algorithms with the new multiple technique, yielding an adaptive elitist-population based genetic algorithm(AEGA).

The AEGA then has been experimentally tested with a difficult test suite consisted of complex multimodal function optimization examples. The performance of the AEGA is compared against the fitness sharing, determining crowing and clonal selection algorithms. All experiments have demonstrated that the AEGA consistently and significantly outperforms the other three multimodal evolutionary algorithms in efficiency and solution quality, particularly with efficiency speed-up of many orders.

We plan to apply our technique to hard multimodal engineering design problems with the expectation of discovering novel solutions. We will also need to investigate the behavior of the AEGA on the more theoretical side.

Acknowledgment. This research was partially supported by RGC Earmarked Grant 4212/01E of Hong Kong SAR and RGC Research Grant Direct Allocation of the Chinese University of Hong Kong.

References

1. D.E. Goldberg and J.Richardson, Genetic algorithms with sharing for multimodal function optimization, *Proc. 2nd ICGA*, pp.41-49, 1987
2. Eshelman.L.J. The CHC adaptive search algorithm: how to have safe search when engaging in nontraditional genetic recombination. In *Rawlins, G.J.E., editor, Foundation of Genetic Algorithms* pp.265-283, Morgan Kaufmann, San Mateo, California, 1991
3. Holland and J.H. Adaptation in Natural and ArtificialSystem. *University of Michigan Press*, Ann Arbor, Michigan, 1975.
4. K.A. De Jong, An analysis of the behavior of a class of genetic adaptive systems, *Doctoral dissertation, Univ. of Michigan*, 1975
5. L.N.Castro and F.J.Zuben, Learning and Optimization Using the Clonal Selection Principle, *IEEE Transactions on EC*, vol .6 pp.239-251, 2002
6. Sarma.J. and De Jong, Generation gap methods. *Handbook of Evolutionary Computation*, pp.C2.7:1-C2.7:5, 1997
7. S.W.Mahfoud. Niching Methods for Genetic algorithms, *Doctoral Dissertation, IlliGAL Report 95001, University of Illinois at Urbana Champaign, Illinois Genetic Algorithm Laboratory*, 1995
8. Whitley.D., The GENITOR algorithm and selection pressure: why rank-based allocation of reproductive trials is best. *Proc. 3rd ICGA*, pp.116-121, 1989

Wise Breeding GA via Machine Learning Techniques for Function Optimization

Xavier Llorà and David E. Goldberg

Illinois Genetic Algorithms Laboratory (IlliGAL),
National Center for Supercomputing Applications,
University of Illinois at Urbana-Champaign,
104 S. Mathews Avenue, Urbana, IL 61801.
{xllora,deg}@illigal.ge.uiuc.edu

Abstract. This paper explores how inductive machine learning can guide the breeding process of evolutionary algorithms for black-box function optimization. In particular, decision trees are used to identify the underlying characteristics of good and bad individuals, using the mined knowledge for wise breeding purposes. Inductive learning is complemented with statistical learning in order to define the breeding process. The proposed evolutionary process optimizes the fitness function in a dual manner, both maximizing and minimizing it. The paper also summarizes some tuning and population sizing issues, as well as some preliminary results obtained using the proposed algorithm.

1 Introduction

Recently, a new interest in the genetic algorithms (GA) community has been growing. The work published by Baluja [1,2], Juels & Wattenberg [3], and Mühlenbein & Paaß [4]—among others—sparked a new way to approach to GA. Instead of recombining genes, as in a traditional GA, this new approach proposes the usage of explicit statistics as the main breeding force. These kind of GA are known as probabilistic model building GA (PMBGA), or estimation of distribution algorithms (EDAs) [5]. Instead of using crossover or mutation operators, these GA breed a new population of individuals sampling a learned probabilistic model that describes the good individuals in the population.

Some early efforts done in PMBGA assume that the probability distribution of each gene can be computed independently. This assumption, not true in many real-world problems, leads to some well-know algorithms. Some examples of algorithms that assume gene independence are: PBIL (*Population Based Incremental Learning*) [1], UMDA (*Univariate Marginal Distribution Algorithms*) [6], and the cGA (*compact Genetic Algorithm*) [7]. Dependences among genes, and what that implies to the distributions to be learned, have also been studied by several authors. Relevant work deals with gene dependence modeling and linkage learning. For an overview of these approaches please see [5,8]. An example of this kind of algorithms is BOA (*Bayesian Optimization Algorithm*) [9]. BOA bases its probability distribution on a Bayesian network. This network describes

the probability distribution of the good individuals in the population, as well as the dependences among genes.

There are other non-statistical approaches to model building GA for breeding purposes. Some of them are provided by inductive machine learning techniques. An example of this kind of approach is LEM (*Learnable Evolution Model*) proposed by Michalski [10,11]. LEM combines two different learning paradigms, evolutionary learning and inductive rule learning. On the one hand, the evolutionary learning uses a simple GA for function optimization. On the other hand, LEM1 and LEM2 uses AQ15 [12] and AQ18 [13] for rule learning. However, LEM is a hybrid approach that still relies on traditional GA mechanisms, although they can be turned off, moving it away from the ideas that inspired PMBGA and EDAs.

This paper presents an alternative approach. The work combines statistical models and inductive machine learning for guiding the breeding process. The goal is to create an algorithm that exploits the knowledge that can be inferred from the current population. In order to mine the population looking for useful breeding knowledge, we combine two different machine learning methods under a supervised learning paradigm. This is achieved casting the optimization task into a problem that can be solved using machine learning techniques. The goal is boosting the evolutionary search process using the mined knowledge. This population based learning algorithm, SI3E (*statistical and inductive tree based evolution*), is applied to some optimization problems, showing its competence, as well as some interesting evolutionary dynamics, optimizing the fitness function in a dual manner, both maximizing and minimizing it. Moreover, casting the optimization into an inductive learning problem provides an elegant way for population sizing of the proposed algorithm.

The paper is structured as follows. Section 2 reviews some background, needed for understanding SI3E. Then, section 3 describes the evolutionary model proposed by SI3E. This section also discusses the parameter tuning and population sizing for a competent usage of the algorithm. Section 4 summarizes the preliminary results obtained using the algorithm on two different well-known black-box optimization functions. Finally, section 5 presents some conclusions about the work presented in this paper.

2 Background

The goal of the work presented in this paper is to combine the ideas that inspired PMBGA, EDAs, and inductive machine learning. SI3E (*statistical and inductive tree based evolution*) is the result of mixing a PMBGA and inductive decision trees. Properly speaking, SI3E defines a common framework based on PBIL [1] and ID3 [14,15]. This section describes both algorithms briefly, PBIL and ID3, in order to explain how SI3E integrates both for obtaining a model-based breeding GA in the next section.

2.1 Population Based Incremental Learning (PBIL)

PBIL was introduced by Baluja [1], and was later improved in [2,16]. The goal of PBIL is to solve a black-box optimization problem. The target function is defined over a binary space $\Omega = \{0, 1\}^\ell$. Thus, each individual in the population can be represented by a binary string. However, PBIL does not maintain the population. Instead, PBIL models the population using a probability vector $p(x)$ of size ℓ :

$$p(x) = (p_t(x_1), p_t(x_2), \dots, p_t(x_\ell)) \quad (1)$$

where $p_t(x_i)$ refers to the probability of finding a 1 in the i^{th} position of D_t , the population of individuals in the t^{th} generation. Thus, PBIL models the population using the probability vector $p(x)$.

The algorithm samples the distribution defined by $p(x)$ generating m individuals. These individuals are evaluated using the fitness function. Then, the best n individuals ($n \leq m$) are selected. These individuals can be denoted, as shown in [5], by:

$$x_{1:m}^t, \dots, x_{i:m}^t, \dots, x_{n:m}^t \quad (2)$$

Then, $p(x)$ is updated using these selected individuals. This update is done using the following rule:

$$p_{t+1}(x) = (1 - \alpha) p_t(x) + \alpha \frac{1}{n} \sum_{k=1}^n x_{k:m}^t \quad (3)$$

where α is a parameter that controls the learning rate.

2.2 Induction of Decision Trees (ID3)

The induction of decision trees was born in the machine learning community. The goal is the learning of a concept from a set of illustrative examples, or supervised learning. The first well-known approach to the induction of decision trees, ID3, was proposed by Quinlan [14]—improved later as C4.5 [15]. ID3 is a tree inducer based on a divide and conquer strategy. Given a set of examples of the target concept, the algorithm picks an attribute for building the root node of the tree. Once the root node is chosen, the data set is divided according to the values of the selected attribute. For each of these divided data sets, the process is repeated recursively creating the branches of the root node. The process stops when all the examples in the data set belong to the same class of the target concept.

The selection of the splitting attribute becomes critical for the induced decision tree. ID3 measures the purity of the split introduced, by picking an attribute, using an *information gain* heuristic. The goal is to minimize the impurity that introduces the split, looking to build a compact decision tree. The *information gain* heuristic is based on the *entropy* measure. The *entropy* of a given set D [17],

relative to a classification task of c classes (generalized version of the concept learning problem from positive and negative examples) is:

$$Entropy(D) \equiv \sum_{i=1}^c -p_i \log_2 p_i \quad (4)$$

where p_i is the proportion of D belonging to class i . Using the *entropy* measure, the *information gain*, $Gain(D, A)$, of an attribute A is the expected reduction in *entropy* caused by partitioning the examples according to this attribute. The *information gain* is then defined as

$$Gain(D, A) \equiv Entropy(D) - \sum_{v \in Values(A)} \frac{|D_v|}{|D|} Entropy(D_v) \quad (5)$$

where $Values(A)$ is the set of all possible values for attribute A , and D_v is the subset of D for which attribute A has value v (i.e. $D_v = \{d \in D | (A(d) = v)\}$).

3 Statistical Learning + Inductive Learning = SI3E

The previous section presented some basic background for SI3E. This section explains how statistical learning and inductive learning can be mixed in order to provide a breeding model for GA. As explained in the introduction, the goal is to create an algorithm that exploits the knowledge that can be mined from the current population. Inductive learning is going to be in charge of this task, however, as we will show later, it has some limitations if it is to become the core of a breeding model for GA. Nevertheless, this limitations can be overcome by combining inductive learning and statistical learning.

3.1 Knowledge Extraction from a Population Using ID3

The work presented in this paper deals with black-box optimization functions defined on a binary space, $\Omega = \{0, 1\}^\ell$. Thus, an individual x of the population is defined by a binary string, $x \in \Omega$. The target function is used as a fitness function, $f : \Omega \mapsto \mathbb{R}$. Using the fitness function $f(x)$, we can sort the individuals in a population $\mathcal{P}$. Therefore, we know which are the best and worst individuals. As proposed in LEM [11], we mark the best individuals as *positive* examples, $\mathcal{P}^\oplus$. In the same way, we can mark the worst individuals as *negative* examples, $\mathcal{P}^\ominus$. These sets contain the best and worst individuals seen so far in the evolutionary process. This approach differentiates SI3E from PMBGA and EDAs. The main reason for this two sets is the result of casting the optimization problems into a supervised learning problem. Using this approach, we artificially create a data set, $D = \mathcal{P}^\oplus \cup \mathcal{P}^\ominus$, from which we can extract knowledge using any supervised learning algorithm from the *machine learning* literature. All the instances in the data set D are individuals of the population. Therefore, we deal with a binary classification task (*positive* and *negative* examples) where all ℓ attributes

are binary. Then, the goal is to find an explicit representation of the differences between *positive* and *negative* instances—individuals of the population. Since the classification problem to be solved is quite simple, ID3 is suited for inducing a compact tree, leaving more complex approaches (i.e C4.5 [15]) for further study.

Table 1. Turning the population into a data set D for supervised learning. Using D , the rules induced by ID3 are: $\mathcal{R}=\{0***:\ominus, 10**:\ominus, 11**:\oplus\}$.

Artificially created data set $\mathcal{D}$											
Rank	Genotype	$f(x)$	Class	Rank	Genotype	$f(x)$	Class	Rank	Genotype	$f(x)$	Class
0	1111	4	$\oplus$	4	1101	3	not used	8	0101	2	$\ominus$
1	1110	3	$\oplus$	5	1110	3	not used	9	1010	2	$\ominus$
2	1110	3	$\oplus$	6	0111	3	not used	10	0001	1	$\ominus$
3	1110	3	$\oplus$	7	1100	2	not used	11	0001	1	$\ominus$

Table 1 shows how a randomly generated population $\mathcal{P}$ can be turned into a data set $\mathcal{D}$ for supervised learning. In this example, the size of the individuals in the population is $\ell = 4$, whereas the population size $|\mathcal{P}| = 12$. The fitness $f(x)$ is computed using the OneMax function [18]. After sorting the population according to $f(x)$, we split the population into three subsets: the best individuals, the worst individuals, and the mediocre ones. We will discuss more about how this split is performed later on. Once we have built $\mathcal{D}$, we used ID3 for inducing a decision tree, extracting the set of equivalent rules. For further details, please see [14,15,17]. In the *condition* part of the rules, 0 or 1 denotes the allele to be used at the given gene, whereas * indicates that the gene can take any value (i.e. 0 or 1). The *class* part of the rules identifies whether the rule describes a characteristic of a good individual ($\oplus$) or a bad one ($\ominus$). We can take the interpretation of these rules one step further. They can be seen as a kind of notation of the underlying *schemas* [19] for good and bad individuals. This point can be assumed if we agree that the knowledge mined by ID3 is actually the *building blocks* of good and bad individuals.

The rules produced by ID3 are the base of the breeding model used by SI3E. However, there is a question that must be answered before using such a thing. The rules produced by ID3 split the binary space of the genotype in a non-overlapping and recursive way. This means that rules cannot be combined in any useful way. Moreover, there are a large number of genes marked as * in the rule. This means that if we pick a rule in order to generate a new good individual, the genotype will be under-specified, because the rule is giving no clue about the value to be substituted in place of the *. This is where PMBGA can help solve this problem.

3.2 Can PBIL Ideas Help?

The rules obtained using ID3 show the underlying patterns of good and bad individuals. Nevertheless, there are several approaches to fill the empty genes

described by the rules. PBIL provides a simple one using the *positive examples* ($\mathcal{P}^{\oplus}$) and the *negative examples* ($\mathcal{P}^{\ominus}$) sets. Using both data sets we can compute the appearance probability of the alleles, 0 and 1, for each gene. Then, given a rule produced by ID3, we can fill the unspecified positions probabilistically.

For choosing an allele for the empty genes we compute two probability vectors, $p(x|\mathcal{P}^{\oplus})$ and $p(x|\mathcal{P}^{\ominus})$, using the *positive* and *negative* examples data sets. Given an example data set $\mathcal{S}$, the probability vector $p(x|\mathcal{S})$ of size ℓ is

$$p(x|\mathcal{S}) = (p(x_1|\mathcal{S}), p(x_2|\mathcal{S}), \dots, p(x_\ell|\mathcal{S})) \quad (6)$$

where $p(x_i|\mathcal{S})$ refers to the probability of finding a 1 in the i^{th} variable of $\mathcal{S}$.

Thus, combining the rules with the statistical information, we can generate new good and bad individuals. However, there are still two issues to solve before we can use this approach for breeding new individuals. They are explained in the next subsection.

3.3 Fixed Loci and Example Set Formation

Before showing the algorithmic description of SI3E, we present the last two remaining issues to solve. The first one arises from the appearance of genes with fixed values across $\mathcal{P}^{\oplus}$ and $\mathcal{P}^{\ominus}$. The second is the formation of the data sets, $\mathcal{P}^{\oplus}$ and $\mathcal{P}^{\ominus}$, using the current population $\mathcal{P}$.

A fixed locus happens when for all the instances of a data set $\mathcal{S}$, the same allele appears on a given gene. This property can be expressed as

$$fixed(x_i, \mathcal{S}) \Leftrightarrow \mathcal{S}_{x_i}^j = \mathcal{S}_{x_i}^{j+1}, \forall j = 1, 2, \dots, |\mathcal{S}| - 1 \quad (7)$$

where x_i is the i^{th} gene, $\mathcal{S}$ the available data set, and $\mathcal{S}^j$ the j^{th} instance of the $\mathcal{S}$ data set. The set of fixed locus can be defined as

$$\phi(X, \mathcal{P}^{\oplus}, \mathcal{P}^{\ominus}) = \{x_i \in X | fixed(x_i, \mathcal{P}^{\oplus}) \wedge fixed(x_i, \mathcal{P}^{\ominus}) \wedge \mathcal{P}_{x_i}^{\oplus} \neq \mathcal{P}_{x_i}^{\ominus}\} \quad (8)$$

where $\mathcal{P}_{x_i}^{\oplus}$ is allele of the fixed position x_i in the $\mathcal{P}^{\oplus}$ data set, as well as $\mathcal{P}_{x_i}^{\ominus}$ is the allele of the fixed position x_i in the $\mathcal{P}^{\ominus}$ data set.

Fixed loci mislead ID3 and the *information gain* heuristic. The problem arises when the same gene appears as a fixed locus on both $\mathcal{P}^{\oplus}$ and $\mathcal{P}^{\ominus}$ sets, but the fixed locus represent different alleles. An example of a data set D with a fixed locus is: $\mathcal{R} = \{1100:\ominus, 0001:\ominus, 0110:\oplus, 1111:\oplus\}$. The gene x_3 contains the allele 0 for all the instances in $\mathcal{P}^{\ominus}$, whereas the allele 1 is fixed in the $\mathcal{P}^{\oplus}$ data set. If we compute the *gain* using the data set $D = \mathcal{P}^{\ominus} \cup \mathcal{P}^{\oplus}$ with the fixed gene x_3 , we obtain $Gain(D, x_3) \equiv 1$. $Gain(D, x_3)$ is the maximum gain using the four genes available. Moreover, if we select x_3 and split the data set D , we obtain a perfect description of good and bad individuals in data set $\mathcal{D}$. Therefore, ID3 would stop there and produce two different rules, $**0*:\ominus$ and $**1*:\oplus$.

Fixed loci turn the breeding process into a PBIL equivalent. Nevertheless, fixed loci do not provide much breeding information, since they stop ID3 from exploring linkages among genes. This situation can be avoided by removing the

fixed loci from the set of available genes to explore. This fact does not imply that these fixed genes are ignored, on the contrary, $p(x|\mathcal{P}^\oplus)$ and $p(x|\mathcal{P}^\ominus)$ contains enough breeding information for their correct usage.

Another critical point in SI3E is the formation of $p(x|\mathcal{P}^\oplus)$ and $p(x|\mathcal{P}^\ominus)$ data sets. In this paper we use a simple approach suggested by LEM [11]. As shown in table 1, we split the current population $\mathcal{P}$, ordered by fitness, into three equally sized disjoint subsets. The best individuals form $\mathcal{P}^\oplus$, the worst ones form $\mathcal{P}^\ominus$, and the rest are not used. This process looks for obtaining opposite instances of the concept to be learned (good or bad individuals), removing the regular ones in order to avoid adding extra noise.

However, if we use this splitting policy strictly, we may be introducing some noise. We can see in table 1, that the last best individual of $\mathcal{P}^\oplus$ has a fitness value of 3. There are three more individuals that also have this fitness value that were not chosen. The same happens with the last worst individual that form $\mathcal{P}^\ominus$ having a fitness value of 2, leaving 1 individuals with the same fitness out of the set. These random choices introduce noise in the learning process of ID3. We softened this splitting policy, adding these equivalent fitness individuals. However, we still maintain that the fitness of the worst individual in $\mathcal{P}^\oplus$ should be better than the fitness of the best individual in $\mathcal{P}^\ominus$.

3.4 Everything in Place: SI3E Algorithms and Their Tuning

The algorithms that implement the evolutionary process of SI3E are presented in figures 1 and 2. Inspecting the algorithms, it can be seen that there are two parameters for tuning SI3E. These parameters are α , the learning rate, and the population size. In this paper we tune these parameters using previous work. The first one, α , is usually set between $[0.1, 0.2]$. These values are common in the *reinforcement learning* community. Please refer to [17,20] for more information.

The other parameter to tune is the population size. Our population sizing model is based on guarantee successful breeding. That is, we size the population in terms of $\mathcal{P}^\oplus$ and $\mathcal{P}^\ominus$. Moreover, if we assume the policy presented in section 3.3, the population size $|\mathcal{P}|$ is approximated by $|\mathcal{P}| \approx 3|\mathcal{P}^\oplus|$. Thus, the sizing model for $\mathcal{P}$ is obtained determining the sizing model of $|\mathcal{P}^\oplus|$. The size of $\mathcal{P}^\oplus$, relies on the capabilities of the learning algorithms, ID3 and PBIL. In this paper, we will only focus on the worst case provided by ID3, trying to estimate the lower bound of $|\mathcal{P}^\oplus|$. In other words, we need to estimate the minimum size of $\mathcal{P}^\oplus$ that leads ID3 to learn a competent description of the target concept (characteristics of good and bad individuals).

Our population sizing problem can be translated into a well-known problem in the *machine learning* community. This problem is to determine the number of instances (samples of the hypotheses space) that a learning algorithm requires for learning the target concept. Haussler [21] compute this lower bound using the theoretical framework proposed by Valiant [22]. This computation relies on the *probably approximately correct* (PAC) model. PAC learners, like ID3, learn target concepts from some concept class C , using training examples drawn at random according to an unknown, but fixed, probability distribution. It requires


```

SI3E( $\mathcal{P}$ )
   $t \leftarrow 0$ 
  initialize  $\mathcal{P}(t)$ ,  $p(x|\mathcal{P}^\oplus)(t)$ , and  $p(x|\mathcal{P}^\ominus)(t)$ 
  evaluate and sort  $\mathcal{P}(t)$ 
  WHILE  $\neg$  end-criteria-satisfied
  DO
    split  $\mathcal{P}(t)$  into  $\mathcal{P}^\oplus$  and  $\mathcal{P}^\ominus$ 
    compute  $p(x|\mathcal{P}^\oplus)(t+1)$  and  $p(x|\mathcal{P}^\ominus)(t+1)$ 
    identify fixed locus  $\phi(X, \mathcal{P}^\oplus, \mathcal{P}^\ominus)$  using
       $p(x|\mathcal{P}^\oplus)(t+1)$  and  $p(x|\mathcal{P}^\ominus)(t+1)$ 
    obtain the rule set  $\mathcal{R}$  using ID3,  $\phi(X, \mathcal{P}^\oplus, \mathcal{P}^\ominus)$ ,  $\mathcal{P}^\oplus$ , and  $\mathcal{P}^\ominus$ 
     $p(x|\mathcal{P}^\ominus)(t+1) \leftarrow (1 - \alpha) \cdot p(x|\mathcal{P}^\ominus)(t) + \alpha \cdot p(x|\mathcal{P}^\ominus)(t+1)$ 
     $p(x|\mathcal{P}^\oplus)(t+1) \leftarrow (1 - \alpha) \cdot p(x|\mathcal{P}^\oplus)(t) + \alpha \cdot p(x|\mathcal{P}^\oplus)(t+1)$ 
    breed a new population  $\mathcal{P}(t+1)$  using
       $\mathcal{R}$ ,  $p(x|\mathcal{P}^\oplus)(t+1)$ ,  $p(x|\mathcal{P}^\ominus)(t+1)$ ,  $\mathcal{P}^\oplus$ , and  $\mathcal{P}^\ominus$ ,  $t$ 
     $t \leftarrow t+1$ 
    evaluate and sort  $\mathcal{P}(t)$ 
  DONE
  RETURN  $\mathcal{P}$ 

```

Fig. 1. Algorithm implemented by SI3E.

that the learner (with probability at least $[1-\delta]$) learns an hypothesis that is approximately (within error ϵ) correct.

Within the setting of the PAC learning model, any consistent learner using a finite hypothesis space H where $C \subseteq H$ (where $|H| = 2^\ell$ in SI3E), with a probability $(1-\delta)$, output a hypothesis within error ϵ of the target concept after observing m randomly drawn training examples, as long as sufficient examples are provided. This bound is computed [21] as

$$m \geq \frac{1}{\epsilon} \left(\ln \frac{1}{\delta} + \ell \ln 2 \right) \quad (9)$$

Equation 9 suggest an interesting result for SI3E, when analyzed from an asymptotically: m must grow at $O(\ell)$. Therefore, the number of new individuals generated, *MAX* (see figure 2), should grow linearly to the length ℓ of the individuals in the population; that is, logarithmically to the size of the search space. For further detail about the PAC model, please refer to [22,21,17].

4 Experiments

In this section we present some preliminary results obtained using SI3E. The conducted experiments involved SI3E and two different functions (OneMax, introduced in section 3, and the concatenation of 4-bit deceptive traps, see [18].)

```

Breed( $\mathcal{P}, \mathcal{R}, p(x|\mathcal{P}^{\oplus}), p(x|\mathcal{P}^{\ominus}), \mathcal{P}^{\oplus}, \mathcal{P}^{\ominus}, t$ )
   $\mathcal{P}(t+1) \leftarrow \mathcal{P}^{\oplus} \cup \mathcal{P}^{\ominus}$ 
   $i \leftarrow 0$ 
  WHILE ( $i < \text{MAX}$ )
    DO
      draw a rule  $r$  from  $\mathcal{R}$  at random
      IF ( $r \in \oplus$ )
        THEN
           $x \leftarrow$  fill the unspecified positions of  $r$  using  $p(x|\mathcal{P}^{\oplus})$ 
        ELSE
           $x \leftarrow$  fill the unspecified positions of  $r$  using  $p(x|\mathcal{P}^{\ominus})$ 
      FI
       $\mathcal{P}(t+1) \leftarrow \mathcal{P}(t+1) \cup \{x\}$ 
       $i \leftarrow i+1$ 
    DONE
  RETURN  $\mathcal{P}$ 

```

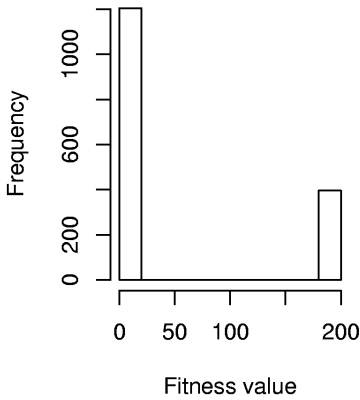
Fig. 2. Breeding algorithm used by SI3E.

These experiments were designed for showing the viability of the approach suggested by SI3E, as well as, for studying its scalability. SI3E was tuned as follows. α was set to 0.2 (please refer to [17,20]). The population size was set using the results presented in the previous section. The population was parameterized as follows: $\text{MAX}=4\ell$, $|\mathcal{P}^{\oplus}| = \ell$, and $|\mathcal{P}^{\ominus}| = 3\ell$, biasing the population towards the *negative* instances. The reason for this bias is that *negative* usually outnumber the *positive* ones (i.e 4-bit deceptive traps).

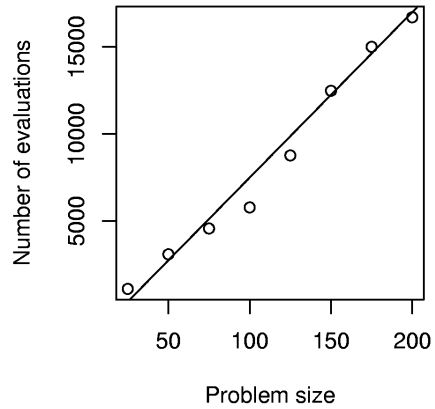
Figures 3(a) and 3(c) shows the distribution of the population evolved by SI3E solving OneMax and the concatenation of 4-bit deceptive traps. As explained in section 3, SI3E is optimizing the fitness function in a dual manner (both maximizing and minimizing it). Thus, the evolved population collapses on two different regions. On the right-hand side, the best ones (highest fitness set $\mathcal{P}^{\oplus}$), whereas on the left-hand side the worst evolved individuals (lowest fitness set $\mathcal{P}^{\ominus}$) can be found. The amount of individuals on each set is the result of the ratio of good and bad individuals (1:3).

We also conducted some preliminary analysis about the scalability of SI3E. Figures 3(b) and 3(d) summarizes the obtained results using SI3E solving the OneMax and the concatenation of 4-bit deceptive traps functions. Each result is the average of 50 independent runs. SI3E scales linearly in the OneMax problem as shown in figure 3(b). These results was not unexpected. PBIL performs in $O(\ell)$ in the OneMax problem. Hence, the building block identification introduced by ID3 do not degrade the overall performance.

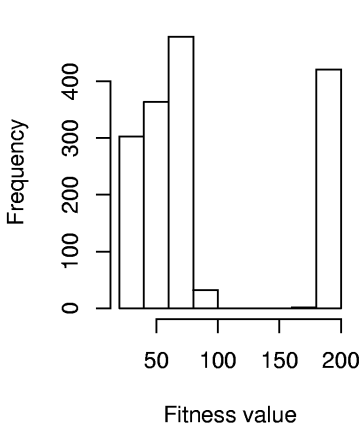
The building block identification, performed by ID3 in SI3E, proves its usefulness when solving the concatenation of 4-bit deceptive traps. As mentioned before, this deceptive function has a clear underlying patterns. Thus, SI3E can



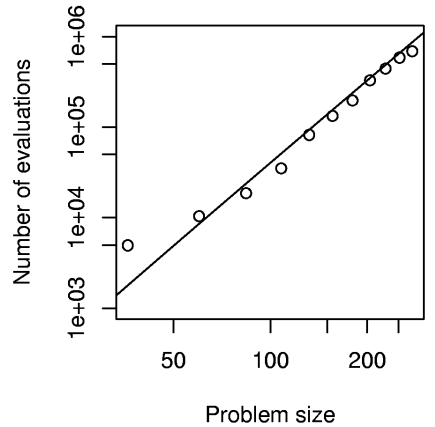
(a) Final population distribution for OneMax ($\ell=200$).



(b) Growth for OneMax ($O(\ell)$). Figure plotted using linear axes.



(c) Final population distribution for 4-bit *deceptive trap* ($\ell=200$).



(d) Growth for 4-bit traps ($O(\ell^3)$). Figure plotted using logarithmic axes.

Fig. 3. Results obtained using SI3E. Figures show the results solving two different optimization functions.

learn these underlying patterns, being later exploited in the breeding phase of the algorithm. Theoretical studies show that simple GA scales exponentially

with the length of individuals when solving deceptive trap functions, while competent GA, like BOA [8], achieve this goal in subquadratic time [18]. Figure 3(d) summarizes the results obtained. Results suggested polynomial scalability of the number of evaluations related to the problem size. Empirical results suggested, at most, $O(\ell^3)$ scalability. Although this boundary is one order of magnitude bigger than well-known competent GA, these preliminary results need a deeper analysis. In particular, the population sizing criteria adopted and the artificial $\mathcal{D}$ data set formation criteria should be studied more deeply.

5 Conclusions

This paper has explored the usage of inductive and statistical learning for improving the breeding process of GA. The paper relies on the existence of learnable patterns in the population. These patterns are usually the result of the existing regularities introduced by the fitness function. Thus, using the fitness function, the individuals can be ranked and split in two different subsets. The first one contains the best individuals seen so far in the evolution. Similarly, the second one is formed by the worst individuals seen so far. As a result of this kind of splitting of the population, it can be mined using inductive and statistical learning, looking for the underlying patterns that describe both sets.

SI3E implements these ideas. Using a sorted population, where the best and worst individuals have been identified, SI3E uses ID3, as well as some PBIL ideas, obtaining the underlying existing schemas in the population. These schemas are used later to produce a new population of good and bad individuals. Thus, the breeding phases is wisely guided by these mined patterns. Preliminary results obtained using SI3E suggest the competence of this machine learning based GA for function optimization. This fact is also suggested by the empirical analysis of the scalability of the algorithm, although it should be studied deeply, as summarized in section 4.

Acknowledgments. This work was supported by the Technology Research, Education and Commercialization Center (TRECC), a program of the University of Illinois at Urbana-Champaign, administered by the National Center for Supercomputing Applications (NCSA) and funded by the Office of Naval Research (N00014-01-1-0175). Funding was also provided the Air Force Office of Scientific Research, USAF, (F49620-00-0163), and the National Science Foundation (DMI-9908252).

References

1. Baluja, S.: Population-based incremental learning: A method for integrating genetic search function optimization and competitive learning. Technical Report CMU-CS-94-163, Carnegie Mellon University (1994)

2. Baluja, S.: An empirical comparison of seven iterative and evolutionary function optimization heuristics. Technical Report CMU-CS-95-193, Carnegie Mellon University (1995)
3. Juels, A., Wattenberg, M.: Stochastic hillclimbing as a baseline method for evaluating genetic algorithms. Technical Report CSD-94-834, Computers Science Department, University of California at Berkeley, USA (1995)
4. Mühlenbein, H., Paaß, G.: From recombination of genes to estimation of distributions I. In: *Lecture Notes in Computer Science 1411: Parallel Problem Solving from Nature - PPSN IV*, Springer (1996) 178–187
5. Larrañaga, P., Lozano, J.: *Estimation of Distribution Algorithms. GENA 2*. Kluwer Academic Publishers (2002)
6. Mühlenbein, H.: The equation for response to selection and its use for prediction. *Evolutionary Computation* **5** (1998) 303–346
7. Harik, G., Lobo, F., Goldberg, D.: The compact genetic algorithm. In: *Proceedings of the IEEE Conference on Evolutionary Computation*, IEEE press (1998) 523–528
8. Pelikan, M.: *Bayesian Optimization Algorithm: from single level to hierarchy*. PhD thesis, University of Illinois at Urbana Champaign, Urbana, Illinois, USA (June, 2002)
9. Pelikan, M., Goldberg, D., Cantú-Paz, E.: BOA: the Bayesian Optimization Algorithm. In: *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-99)*, Morgan Kaufmann (1999) 525–532
10. Michalski, R.: *Learnable Evolution: Combining Symbolic and Evolutionary Learning*. In: *Proceedings of the 4th International Workshop on Multistrategy Learning*, Decenzano del Garda, Italy. (June 11-14, 1998)
11. Michalski, R.: *Learnable Evolution Model: Evolutionary Processes Guided by Machine Learning*. *Machine Learning* **38** (2000) 9–40
12. Wnek, J., Kaufman, K., Bloedorn, E., Michalski, R.: *Inductive Learning System AQ15c: The Method and User's Guide*. Reports of the Machine Learning and Inference Laboratory, MLI 95-4, George Mason University, Fairfax, VA (March, 1995)
13. Kaufman, K., Michalski, R.: *The AQ18 Machine Learning and Data Mining System: An Implementation and User's Guide*. Reports of the Machine Learning and Inference Laboratory, George Mason University, Fairfax, VA (1999)
14. Quinlan, J.R.: *Induction of decision trees*. *Machine Learning* **1** (1986) 81–106
15. Quinlan, J.R.: *C4.5: programs for machine learning*. Morgan Kaufmann (1993)
16. Baluja, S., Caruana, R.: Removing the genetics from standard genetic algorithms. In: *International Conference on Machine Learning*, Morgan Kaufmann (1995) 38–46
17. Mitchell, T.M.: *Machine Learning*. McGraw Hill (1997)
18. Goldberg, D.E.: *The Design of Innovation: Lessons from and for Competent Genetic Algorithms*. Kluwer Academic Publishers (2002)
19. Holland, J.H.: *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*. MIT Press/Bradford Books edition (1975)
20. Gonzalez, C., Lozano, J., Larranarraga, P.: Analyzing the pbil algorithm by means of discrete dynamical systems. *Complex Systems* **12** (2001) 465–479
21. Haussler, D.: Quantifying inductive bias: AI learning algorithms and Valiant's learning framework. *Artificial Intelligence* **36** (1988) 177–221
22. Valiant, L.: A theory of the learnable. *Communications of ACM* **27** (1984) 1134–1142

Facts and Fallacies in Using Genetic Algorithms for Learning Clauses in First-Order Logic

Flaviu Adrian Mărginean

Department of Computer Science, The University of York
Heslington, York YO10 5DD, United Kingdom
`flav@cs.york.ac.uk`

Abstract. Over the last few years, a few approaches have been proposed aiming to combine genetic and evolutionary computation (GECCO) with inductive logic programming (ILP). The underlying rationale is that evolutionary algorithms, such as genetic algorithms, might mitigate the combinatorial explosions generated by the inductive learning of rich representations, such as those used in first-order logic. Particularly, the binary representation approach presented by Tamaddoni-Nezhad and Muggleton has attracted the attention of both the GECCO and ILP communities in recent years. Unfortunately, a series of systematic and fundamental theoretical errors renders their framework moot. This paper critically examines the fallacious claims in the mentioned approach. It is shown that, far from restoring completeness to the learner PROGOL's search of the subsumption lattice, the binary representation approach is both overwhelmingly unsound and severely incomplete.

1 Introduction

Over the last few years there has been a surge of interest in combining the expressiveness afforded by first-order logic representations in inductive logic programming with the robustness of evolutionary search algorithms [7,20,21,22]. It is hoped that such hybrid systems would retain the powerful logic programming formalism and its well-understood theoretical foundations, while bringing in to the search the versatility of evolutionary algorithms, their inherent parallelism, and their adaptive characteristics [21].

PROGOL is a first-order inductive reasoner widely regarded as state of the art. Owing to its relative importance, its soundness and completeness have been the object of numerous theoretical studies [3,8,9,10,14,24]. PROGOL has also been investigated from the point of view of its tractability [16,17]. Search in first-order logic is notoriously difficult because of the expressive power of the hypotheses that generates combinatorial explosions.

Owing to these two issues, (in)completeness and (in)tractability, the announcement by Tamaddoni-Nezhad and Muggleton [20,21,22] that genetic algorithms can solve both problems via a simple binary change of representation has attracted interest. In this paper we demonstrate that, unfortunately, such

hopes are unfounded. We review Tamaddoni-Nezhad and Muggleton's aforementioned approach and show that it is provably flawed at every level. Specifically, we consider the following claims by the authors, which are central to their approach:

Fallacy 1. The proposed binary representation for clauses is novel. It encodes the subsumption lattice lower bounded by the bottom clause in a compact and complete way.

Fallacy 2. A fast evaluation mechanism for clauses has been given.

Fallacy 3. The proposed task-specific genetic operators can be viewed as a form of refinement, titled genetic refinement [20] or stochastic refinement [21].

Respectively, we show:

Fact 1. The proposed binary representations have been known for some time [1,2] and shown to be incomplete for subsumption even for function-free languages. The binary encoding of the subsumption lattice is both incomplete and unsound. Owing to unsoundness, for even a single shared variable in the bottom clause, the proposed space of binary representations is $\frac{2^{\binom{n}{2}}}{B_n}$ times bigger than the number of valid clauses that it manages to encode, where $n \stackrel{\text{def}}{=} \text{the number of predicates that share the variable}$ and B_n is the n -th Bell Number. Therefore the space of binary representations is not compact. An infinity of good clauses are left out (the encoding is *incomplete*) and a huge number of spurious binary strings get in (the encoding is *unsound* and *noncompact*).

Fact 2. The proposed evaluation mechanism for clauses is provably unsound.

Fact 3. The proposed task-specific genetic operators are not refinement operators because of their being provably unsound and incomplete.

The errors that we pinpoint in this paper appear to have no easy fix. They are very fundamental theoretical errors, which undermine the whole binary representation approach. We wish to emphasise that the problem of combining evolutionary computation with first-order logic learning is worth investigating, and in this respect the binary representation attempt is meritorious. However, the flaws need to be corrected. The paper is organised as follows. In Section 2 we review some preliminaries, such as inductive logic programming and inverse entailment. In Section 3 we expose the fallacies undermining the binary representation approach in correlation with our counter-arguments, while Sections 4 and 5 present the conclusions of this paper.

2 Preliminaries

The reader is assumed to be familiar with the basic formalism of first-order clausal logic. The paragraphs below are intended as brief reminders. A good general reference for inductive logic programming is [15] and Muggleton's seminal paper on PROGOL is [13].

2.1 Inductive Logic Programming (ILP)

Under ILP's normal setting (also called the *monotonic* setting, or the *learning-from-entailment* setting) one considers background knowledge B , positive and negative examples E^+ and E^- , and a hypothesis H . B , E^+ , E^- and H are all finite sets of clauses (*theories*). Further restrictions can be applied, for instance by requiring that B , H , E^+ , E^- are logic programs, rather than general theories, or imposing that positive examples are ground unit definite clauses and negative examples are ground unit headless Horn clauses. The central goal is to induce (*learn*) H such that the following two conditions are satisfied:

$$\begin{cases} B \wedge H \models E^+ \\ B \wedge H \wedge E^- \not\models \square \end{cases}$$

This looks rather much like solving a system of inequations (logical entailment $\models$ is a quasi-order, i.e. reflexive and transitive), except that it is a rather complicated one. However, if one only considers the positive examples, in the first instance, the following simpler system is obtained:

$$\begin{cases} B \wedge H \models E^+ \\ B \wedge H \not\models \square \end{cases}$$

Progress has been made towards characterising the solutions to this system as follows:

2.2 Inverse Entailment

Definition 1 (Subsumption). *Let C and D be clauses. Then C θ -subsumes D , denoted by $C \succeq D$, if there exists a substitution θ such that $C\theta \subseteq D$ (i.e. every literal in $C\theta$ is also a literal in D).*

Definition 2 (Inverse Entailment). *Inverse Entailment is a generic name for any computational procedure that, given B, E^+ as input, will return a bottom clausal theory $\perp(E^+, B)$ as output, such that the following condition is satisfied:*

$$H \models \perp(E^+, B) \iff \begin{cases} B \wedge H \models E^+ \\ B \wedge H \not\models \square \end{cases}, \forall H$$

Inoue (2001) has provided the only known example of Inverse Entailment in the general case (under the name of Consequence-Finding). It was hoped that entailment on the left-hand side might be replaced with subsumption or another decidable quasi-order, as entailment is only semidecidable in the general case. However, this hope was largely unfulfilled. In more restricted settings, the following results have been obtained:

For H , E restricted to be clauses rather than theories, Yamamoto (1997) gives a computational procedure that computes a bottom clause $\perp(E^+, B)$ such that the following condition is satisfied:

$$H \succeq \perp(E^+, B) \iff \begin{cases} H \succeq_B E^+ \\ H \not\preceq_B \square \end{cases}, \forall H$$

Note that entailment has been replaced with the more restricted subsumption on the left-hand side and Plotkin's relative subsumption on the right-hand side.

For H , E restricted to be function-free clauses and B a function-free Horn theory, Muggleton (1998) gives a computational procedure that computes a bottom clause $\perp(E^+, B)$ such that the following condition is satisfied:

$$H \succeq \perp(E^+, B) \not\Longleftrightarrow \begin{cases} B \wedge H \models E^+ \\ B \wedge H \not\models \square \end{cases}, \forall H$$

Note that entailment has been replaced with the more restricted subsumption on the left-hand side but the soundness of Inverse Entailment ($\implies$) has been lost.

For H , E restricted to be function-free Horn clauses and B a function-free Horn theory, Muggleton (1995) gives a computational procedure that computes a bottom Horn clause $\perp(E^+, B)$ such that the following condition is satisfied:

$$H \succeq \perp(E^+, B) \Longleftrightarrow \begin{cases} B \wedge H \models E^+ \\ B \wedge H \not\models \square \end{cases}, \forall H$$

Note that entailment has been replaced with the more restricted subsumption on the left-hand side but the completeness of Inverse Entailment ($\Longleftrightarrow$) has this time been lost. Completeness can be restored to this version if either entailment is restored on the left-hand side or the unique bottom clause is replaced with a family $\{\perp_i(E^+, B)\}_i$ of bottom clauses (subsaturants of the unique bottom clause computed by Muggleton's procedure):

$$\bigvee_i [H \succeq \perp_i(E^+, B)] \Longleftrightarrow \begin{cases} B \wedge H \models E^+ \\ B \wedge H \not\models \square \end{cases}, \forall H$$

Subsumption is, in general, preferred to entailment on the left-hand side since it is decidable. However, as apparent from before, it only guarantees completeness and soundness of Inverse Entailment when the general ILP setting is restricted and multiple bottom clauses are generated.

3 The Binary Representation Approach: Facts and Fallacies

In the context of the Inverse Entailment procedure discussed in the preceding section, Tamaddon-Nezhad and Muggleton consider the case where one has a function-free bottom Horn clause $\perp(E^+, B)$ and claim that the space of solutions $\{H \mid H \succeq \perp(E^+, B), H \text{ is a function-free Horn clause}\}$ can be described as a boolean lattice obtained from the variable sharing in the bottom clause according to a simple procedure (Fig. 1).

3.1 Fallacy 1 — Fact 1

We first give a description of the proposed binary representation. In [20,21] the following definition is given:

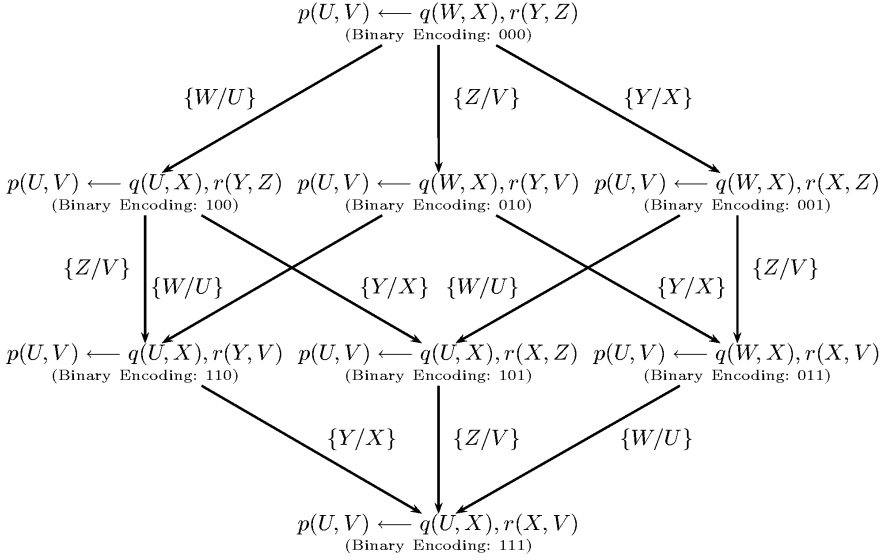


Fig. 1. Tamaddoni-Nezhad and Muggleton’s “subsumption lattice” bounded below by the bottom clause $p(U, V) \leftarrow q(U, X), r(X, V)$

Definition 3 (Binding Matrix). Suppose B and C are both clauses and there exists a variable substitution θ such that $C\theta = B$. Let C have n variable occurrences representing variables $\langle v_1, v_2, \dots, v_n \rangle$. The binding matrix of C is an $n \times n$ matrix M in which m_{ij} is 1 if there exist variables v_i, v_j and u such that v_i/u and v_j/u are in θ and m_{ij} is 0 otherwise. We write $M(v_i, v_j) = 1$ if $m_{ij} = 1$ and $M(v_i, v_j) = 0$ if $m_{ij} = 0$.

This definition is unsound because the binding matrix of C is defined with respect to an arbitrary clause B . It is obvious that such a binding matrix may not be unique. We therefore assume that the authors meant B to be a fixed bottom clause and the binding matrix of C was defined with respect to this fixed bottom clause. Let us consider the bottom clause $p(U, V) \leftarrow q(U, X), r(X, V)$ in Fig. 1. Using the equality predicate we can re-write the clause as follows:

$$p(X_1, X_2) \leftarrow q(X_3, X_4), q(X_5, X_6), X_1 = X_3, X_2 = X_6, X_4 = X_5$$

We note that the variable sharing in the bottom clause is now completely described by the three equalities. Any other clause in Fig. 1 can be re-written as a combination of the common factor $p(X_1, X_2) \leftarrow q(X_3, X_4), q(X_5, X_6)$ and a subset of the three equalities $\{X_1 = X_3, X_2 = X_6, X_4 = X_5\}$ that describe the variable sharing in the bottom clause. For instance, clause $p(U, V) \leftarrow q(U, X), r(Y, Z)$ will become:

$$p(X_1, X_2) \leftarrow q(X_3, X_4), q(X_5, X_6), X_1 = X_3$$

It is now clear that we do not need the common factor, every clause in Fig. 1 being describable by simply noting which of the three equalities in the bottom clause it sets off. If we use the binary string $(1, 1, 1)$ to indicate that the bottom clause satisfies all three equalities, then the second clause above may be encoded as $(1, 0, 0)$. This is the binary representation approach, an instantiation of a technique more commonly known as propositionalisation. This approach was first investigated rigorously within the European ILP2 project supported by ESPRIT Framework IV, which ended in 1999. The deliverables of the ILP2 project [1,2] showed clearly that the approach could not yield completeness for subsumption, not even in the simple case of function-free (Datalog) languages [1]. We now show that this is indeed the case for subsumption lattices bounded below by bottom clauses.

Binary Representation Space is Incomplete. The following clauses are missing from Tamaddoni-Nezhad and Muggleton's subsumption lattice, as pictured in Fig. 1:

$\leftarrow$, the empty clause, subset of the bottom clause
 $p(U, V) \leftarrow$, subset of the bottom clause
 $q(U, X) \leftarrow$, subset of the bottom clause
 $r(X, V) \leftarrow$, subset of the bottom clause
 $p(U, V) \leftarrow q(W, X)$, maps into the bottom clause by substitution $\theta = \{W/U\}$
 $p(U, V) \leftarrow r(X, Z)$, maps into the bottom clause by substitution $\theta = \{Z/V\}$
 $\leftarrow q(U, X), r(Y, V)$, maps into the bottom clause by substitution $\theta = \{Y/X\}$

These clauses, together with the ones in Fig. 1, are the ones that *weakly* subsume the bottom clause¹, i.e. those that map literals injectively to the bottom clause. In addition, an infinity of other clauses that subsume $p(U, V) \leftarrow q(U, X), r(X, V)$ are also missing, for instance: $p(U, V) \leftarrow \{q(U, X_i), r(X_j, V) \mid i \neq j, 1 \leq i, j \leq n\}$ for $n \geq 2$, which maps onto the bottom clause by substitution $\{X_i/X\}_{1 \leq i \leq n}$. We may wonder whether completeness has instead been achieved under subsumption equivalence, i.e. one clause from each equivalence class under subsumption is present in the search space. However, this is not the case: neither of the exemplified missing clauses are subsume-equivalent with any of the clauses already in the search space, nor are they subsume-equivalent between themselves.

The quasi-order used in Fig. 1 is therefore not subsumption but the much weaker atomic subsumption $\succeq_a$ as defined in [15, p. 244]. If, as before, we denote entailment by $\models$, subsumption by $\succeq$, weak subsumption by $\succeq_w$ and the atomic subsumption by $\succeq_a$, we have the following relationship between the four orders:

$$\succeq_a \not\Rightarrow \succeq_w \not\Rightarrow \succeq \not\Rightarrow \models$$

Consequently, we have:

¹ See [3] for a complete and sound encoding of the weak subsumption lattice with a refinement operator.

$$H_{\succeq_a} \subsetneq H_{\succeq_w} \subsetneq H_{\succeq} \subsetneq H_{\models}$$

where

$$H_{\succeq_i} \stackrel{\text{def}}{=} \{H \mid H \succeq_i \perp(E^+, B), H \text{ is a function-free Horn clause}\}$$

for $\succeq_i \in \{\succeq_a, \succeq_w, \succeq, \models\}$. PROGOL's existing refinement encodes $H_{\succeq_w}$, which is incomplete with respect to subsumption: $H_{\succeq_w} \subsetneq H_{\succeq}$. However, the binary representation approach only captures $H_{\succeq_a}$, which is even more incomplete: $H_{\succeq_a} \subsetneq H_{\succeq_w}$.

Binary Representation Space is Unsound. As mentioned before, the incompleteness of propositionalisation for subsumption is known. Now we show that the binary representation is also unsound, i.e. a counter-example may be given wherein the binary representation codifies binary strings that do not correspond to any clause in the subsumption lattice. The bottom clause chosen in Fig. 1 is well-behaved, in the sense that all three equalities involve distinct variables. This was the original example given in [21]. Let us consider, however, the following bottom clause:

$$p(U, U) \leftarrow q(U, X), r(Y, Z)$$

Using equality to re-write the clause we arrive at:

$$p(X_1, X_2) \leftarrow q(X_3, X_4), q(X_5, X_6), X_1 = X_2, X_1 = X_3, X_2 = X_3$$

If we describe this bottom clause by $(1, 1, 1)$ as the binary representation approach prescribes, then $(1, 1, 0)$, $(1, 0, 1)$ and $(0, 1, 1)$ will correspond to no clause. Because of the transitivity of equality, when one has $X_1 = X_2$, $X_1 = X_3$ for instance, one will also have $X_2 = X_3$. In other words, a certain number of binary strings will be spurious. Interestingly, in [21] definitions 3, 4 and 5 capture precisely the space of binary strings that are mapped to valid clauses, i.e. those corresponding to normalised binding matrices or, in the language of this paper, those subsets of binary equalities that are closed under the transitivity of equality. However, it appears from [20,21] that the authors' encoding of the search space used in implementations is not the set of normalised binding matrices or corresponding normalised strings but the unsound space of all boolean binary strings. Now we show that the number of such spurious strings can grow wildly compared to the number of valid clauses encodable by this approach.

Binary Representation Space is Not Compact. Let us consider a bottom clause that has n predicates, all sharing the first variable. For simplicity we assume that all the other variables are distinct:

$$p_0(X, \dots) \leftarrow p_1(X, \dots), p_2(X, \dots), \dots, p_{n-1}(X, \dots)$$

Using equality to re-write the clause we arrive at:

$$p_0(X_0, \dots) \leftarrow p_1(X_1, \dots), \dots, p_{n-1}(X_{n-1}, \dots), X_0 = X_1 \cdots = X_{n-1}$$

Note that we have slightly abused the notation in order to write the clause more compactly. The number of binary equalities $X_0 = X_1, X_0 = X_2, \dots$ etc. is now $\binom{n}{2}$, which will yield $2^{\frac{n(n-1)}{2}}$ binary strings upon encoding. On the other hand, the number of clauses that can be obtained by anti-unification of variables (as we have seen, Tamaddoni-Nezhad and Muggleton do not consider adding/dropping literals, thereby generating incompleteness) is given by the n -th Bell number, i.e. the number of all partitions of the set of variables $\{X_0, \dots, X_{n-1}\}$. The space of encodings will therefore be $\frac{2^{\binom{n}{2}}}{B_n}$ times bigger than the number of valid clauses that it encodes. To get a feeling for this difference, for $n = 12$ the space of encodings will contain

$$2^{66} = 73786976294838206464$$

binary strings, while the space of clauses will contain $B_{12} = 4213597$ valid clauses, i.e. about 1 good encoding for every 17.5 trillion spurious ones. In ILP it is not uncommon for bottom clauses to contain tens or hundreds of predicates with complex variable sharing. It can be shown that, as n grows, this gets worse and worse, i.e. the asymptotic behaviour confirms this tendency: $\frac{2^{\binom{n}{2}}}{B_n} \xrightarrow{n \rightarrow \infty} \infty$.

Because of the sheer number of spurious binary encodings that parasitise the representation space even for low values of n , the search domain may become unsamplable. Furthermore, the authors observe in footnote 4 of [21, p. 642] and paragraph following Example 2 in [20, p. 249] that certain operations on binding matrices may result in matrices that are not normalised, this leading to unsoundness ('inconsistency' in their language). They say that this does not affect the genetic search in practice and could be avoided by normalisation closure. However, this is not the case. We have shown above that unsoundness can cause serious computational problems. Normalisation closure, on the other hand entails itself computational difficulties: in order to be able to compute normalisation closures, one has to revert to variables or keep a separate list of normalisation rules, valid for the bottom clause at hand. For instance, to infer $X_2 = X_3$ from $X_1 = X_2$ and $X_1 = X_3$ one has to encode the variables. Otherwise, if $X_1 = X_2, X_1 = X_3, X_2 = X_3$ are encoded as 3-bit binary strings (Y_1, Y_2, Y_3) , then one needs to know that $Y_1 = 1 \wedge Y_2 = 1 \implies Y_3 = 1$ etc. However, the real problem is that, with the proposed encoding, normalisation closures are syntactically computable but meaningless from a semantic point of view. Suppose that, as in our example before, we get by random sampling (seeding) the string $(1, 1, 0)$. The 0 bit in the string does not mean that we do not know that $X_2 = X_3$. It means that we know that $X_2 \neq X_3$. This is because the encoding is made under a Closed World Assumption: it is assumed that a binary string encodes the variable sharing information completely. If this were not so, encodings would not represent clauses but subsets of clauses, i.e. all clauses in the search domain that are compatible with the variable sharing described by the 1-bits in the binary string. Since a 0 does not reflect incomplete information in the binary string but encodes negative information, normalisation by closure does not fulfill its intended meaning, which is to fill in inferred information.

Normalisation by closure remains to be but one of many alternative ways of resolving inconsistency. It proceeds by switching the minimum number of 0-bits to 1-bits that allows the consistency rules to be satisfied. To resolve inconsistency we can proceed with any other algorithm that, switching a number of bits, would achieve consistency under the normalisation rules. For instance, we can decide that, instead of switching 0-bits to 1-bits, we switch 1-bits to 0-bits. While we can choose, for instance, that the number of bits so changed be minimal (i.e. we determine all valid clauses at minimal Hamming distance), or some other criterion, we do not have any rational basis for preferring one criterion over another. The normalisation by closure is therefore syntactically computable but semantically undefined.

3.2 Fallacy 2 — Fact 2

Let us consider Example 2 in [21]. It is claimed that the clause $p(U, V) \leftarrow q(U, X), r(X, Z)$ is the *mg*² of the clauses $p(U, V) \leftarrow q(U, X), r(Y, Z)$ and $p(U, V) \leftarrow q(W, X), r(X, Z)$ under subsumption $\succeq$. However, this is not true. The *mg* of the two clauses under $\succeq$, according to [15, p. 251], is $p(U, V) \leftarrow q(U, X), q(W, X'), r(Y, Z), r(X', Z')$. We may wonder if this is not subsume-equivalent with $p(U, V) \leftarrow q(U, X), r(X, Z)$. Suppose, by *reductio ad absurdum*, that the latter clause subsumes the former. Then there should be a substitution mapping it onto a subset of the former. Necessarily, $\{U/U, V/V\}$ because of the common head. Then $q(U, X)$ is mapped to some $q(U, \_)$ and the only such literal available is $q(U, X)$, therefore $\{X/X\}$. Then $r(X, Z)$ is mapped to some $r(X, \_)$ but there are no such literals in the former clause, contradiction. In fact the *mg* given in [21] is computed under atomic subsumption $\succeq_a$. Crucially, for *mg*'s under $\succeq_a$ Theorem 4 in [21] does not hold, i.e. one may not compute coverages of such *mg*'s by intersecting the coverages of the parent clauses as described in [21]. This is because coverages of clauses are computed under entailment $\models$ (or, in some circumstances, subsumption $\succeq$) but not under the much weaker atomic subsumption $\succeq_a$. For instance, consider the clause $p(U, V) \leftarrow q(U, X), q(W, X'), r(Y, Z), r(X', Z')$ referred to above. Under $\succeq$ it is covered by both $p(U, V) \leftarrow q(U, X), r(Y, Z)$ and $p(U, V) \leftarrow q(W, X), r(X, Z)$, therefore it belongs to the intersection of their coverages. However, as shown, it is not covered by the clause $p(U, V) \leftarrow q(U, X), r(X, Z)$, the alleged *mg*. This renders the proposed fast evaluation mechanism unsound and therefore moot.

3.3 Fallacy 3 — Fact 3

According to the theory of refinement in ILP [11,15], refinement operators maintain the search space implicitly. Various qualities may be required of any well-defined refinement operator, such as soundness, completeness (weak or strong), local finiteness, properness, minimality etc. Soundness and completeness are minimal requirements. However, we have seen in the preceding sections that the

² most general instantiation

search space explored by the task-specific genetic operators in [20,21] is both unsound and incomplete. The most we could hope for is that such operators, once applied to valid encodings, will produce a valid encoding. However, this is not the case. Both *mg*i and mutation may produce spurious outcomes starting from valid parent strings, for instance taking *mg*i of (1, 0, 0) and (0, 1, 0) in our running counter-example will give the inconsistent (1, 1, 0), while 1-bit mutation applied to (1, 0, 0) will yield two inconsistent strings and a consistent one.

4 Potential Solutions

This paper was initially written and reviewed in a two-part format, containing both our criticism of the binary representation approach and our alternative thereto. Unfortunately, for reasons strictly connected with the 12-page limit allowed for full papers in the conference proceedings, the second part had to be dropped. The author plans to submit an extended version of this paper to a journal in the Artificial Intelligence field, including our proposed solution to the binary representation flaws described in this paper. The interested reader is invited to contact the author for details.

5 Conclusions

Although combining genetic algorithms with inductive logic programming is potentially a valuable approach, it is not straightforward. Inductive logic programming owes its existence both to the first-order representations that it uses, but also to the recognition of the fact that it employs search mechanisms that are not easily translatable in the propositional domain. Even when such transformations can be made, it is usually under heavy restrictions or at the expense of exponential blow-ups in complexity [6]. It is why the inductive logic programming community has painstakingly developed inductive mechanisms that work with first-order representations directly. Particularly, refinement operators defined on clauses lie at the core of many inductive logic programming systems.

Tamaddoni-Nezhad and Muggleton have proposed [20,21] an approach that recycles some well-known propositionalisation techniques under the name of binary representations. Although it is claimed that this approach achieves completeness in respect of PROGOL's search, it is in fact even more incomplete than PROGOL's existing search. Although they use the phrases "genetic refinement" in [20] and "stochastic refinement" in [21] to name this type of search, no definition of these terms is given and no explanation on how they relate to the very consistent body of research on refinement [11,12,15,23]. At present it is not possible to determine with enough precision how the flaws in their theoretical framework affect their implementation since they do not give the algorithm on which the implementation is based, nor enough detail regarding their experimental settings. It is therefore not possible to either replicate or criticise their experimental evidence. However, in [20] the authors state:

“In our first attempt, we employed the proposed representation to combine Inverse Entailment in C-PROGOL4.4 with a genetic algorithm. In this implementation genetic search is used for searching the subsumption lattice bounded below by the bottom-clause ($\perp$). According to Theorem 3 the search space bounded by the bottom clause can be represented by $S(\perp)$.”

However, a careful consideration of Theorem 3 given in their paper will show that the theorem’s statement does not entail the identity between $H_{\perp}$ and $S(\perp)$. Furthermore, the argument can not be repaired: $S(\perp)$ is the space of binary representations shown in this paper to be incomplete, unsound, and noncompact with respect to $H_{\perp}$. Since, by the results of this paper, the statement quoted above does not hold, we should conjecture that caution is needed in accepting the authors’ alleged experimental results until such time that the flaws unearthed in this paper are given a proper solution.

Acknowledgements. Thanks to James Cussens for proofreading a preliminary version of this paper. The author thanks the four anonymous reviewers for their interest, careful evaluation and commentary, part of which is going to be used in future extensions of this work. This paper is dedicated to my maternal grandparents: Maria Iclănzan (Buni) and Andrei Iclănzan (Bunicu’), who taught me the value of imagination and rational endeavour, respectively; and to the memory of my departed paternal grandparents, Anica Mărginean and Ioan Mărginean, who gave me my name and taught me the meaning of faith and dignity in face of adversity.

References

1. É. Alphonse and C. Rouveirol. Object Identity for Relational Learning. Technical report, LRI, Université Paris-Sud, 1999. Supported by ESPRIT Framework IV through LTR ILP2.
2. É. Alphonse and C. Rouveirol. Test Incorporation for Propositionalization Methods in ILP. Technical report, LRI, Université Paris-Sud, 1999. Supported by ESPRIT Framework IV through LTR ILP2.
3. L. Badea and M. Stanciu. Refinement Operators Can Be (Weakly) Perfect. In S. Džeroski and P. Flach, editors, *Inductive Logic Programming, 9th International Workshop, ILP-99*, volume 1634 of *Lecture Notes in Artificial Intelligence*, pages 21–32. Springer, 1999.
4. J. Cussens and A. Frisch, editors. *Inductive Logic Programming—ILP 2000, Proceedings of the 10th International Conference on Inductive Logic Programming*, Work-in-Progress Reports, Imperial College, UK, July 2000. Work-in-Progress Reports.
5. J. Cussens and A. Frisch, editors. *Inductive Logic Programming—ILP 2000, Proceedings of the 10th International Conference on Inductive Logic Programming*, volume 1866 of *Lecture Notes in Artificial Intelligence*, Imperial College, UK, July 2000. Springer.

6. L. De Raedt. Attribute-Value Learning versus Inductive Logic Programming: The Missing Links. In Page [18], pages 1–8.
7. F. Divina and E. Marchiori. Knowledge Based Evolutionary Programming for Inductive Learning in First-Order Logic. In Spector et al. [19], pages 173–181.
8. K. Furukawa and T. Ozaki. On the Completion of Inverse Entailment for Mutual Recursion and its Application to Self Recursion. In Cussens and Frisch [4], pages 107–119.
9. K. Inoue. Induction, Abduction and Consequence-Finding. In C. Rouveirol and M. Sebag, editors, *Inductive Logic Programming—ILP 2001*, volume 2157 of *Lecture Notes in Artificial Intelligence*, pages 65–79. Springer, 2001.
10. K. Ito and A. Yamamoto. Finding Hypotheses from Examples by Computing the Least Generalization of Bottom Clauses. In S. Arikawa, editor, *Proceedings of Discovery Science*, volume 1532 of *Lecture Notes in Artificial Intelligence*, pages 303–314. Springer-Verlag, 1998.
11. P.R. Laag. An Analysis of Refinement Operators in Inductive Logic Programming. Technical Report 102, Tinbergen Institute Research Series, 1995.
12. Philip D. Laird. *Learning from Good Data and Bad*. PhD thesis, Yale University, 1987.
13. S. Muggleton. Inverse entailment and Progol. *New Generation Computing*, 13:245–286, 1995.
14. S. Muggleton. Completing Inverse Entailment. In Page [18], pages 245–249.
15. S-H. Nienhuys-Cheng and R. de Wolf. *Foundations of Inductive Logic Programming*. Springer-Verlag, Berlin, 1997. LNAI 1228.
16. K. Ohara, N. Babaguchi, and T. Kitahashi. An Efficient Hypothesis Search Algorithm Based on Best-Bound Strategy. In Cussens and Frisch [4], pages 212–225.
17. H. Ohwada, H. Nishiyama, and F. Mizoguchi. Concurrent Execution of Optimal Hypothesis Search for Inverse Entailment. In Cussens and Frisch [5], pages 165–173.
18. D. Page, editor. *Inductive Logic Programming, Proceedings of the 8th International Conference, ILP-98*, volume 1446 of *Lecture Notes in Artificial Intelligence*. Springer, July 1998.
19. L. Spector, E.D. Goodman, A. Wu, W.B. Langdon, H-M. Voigt, M. Gen, S. Sen, M. Dorigo, S. Pezeshk, M.H. Garzon, and E. Burke, editors. *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO-2001*, San Francisco, CA, July 7–11 2001. AAAI, Morgan Kaufmann.
20. A. Tamaddoni-Nezhad and S.H. Muggleton. Searching the Subsumption Lattice by a Genetic Algorithm. In Cussens and Frisch [5], pages 243–252.
21. A. Tamaddoni-Nezhad and S.H. Muggleton. Using Genetic Algorithms for Learning Clauses in First-Order Logic. In Spector et al. [19], pages 639–646.
22. A. Tamaddoni-Nezhad and S.H. Muggleton. A Genetic Algorithms Approach to ILP. Proceedings of the Twelfth International Conference on Inductive Logic Programming, 2002. In Press.
23. F. Torre and C. Rouveirol. Private Properties and Natural Relations in Inductive Logic Programming. Technical report, LRI, Université Paris-Sud, July 1997.
24. A. Yamamoto. Which Hypotheses Can Be Found with Inverse Entailment? In N. Lavrač and S. Džeroski, editors, *Inductive Logic Programming, 7th International Workshop, ILP-97*, volume 1297 of *Lecture Notes in Artificial Intelligence*, pages 296–308. Springer, 1997.

Comparing Evolutionary Computation Techniques via Their Representation

Boris Mitavskiy

Department of Mathematics
University of Michigan
Ann Arbor MI 48109
bmitavsk@umich.edu

Abstract. In the current paper a rigorous mathematical language for comparing evolutionary computation techniques via their representation is developed. A binary semi-genetic algorithm is introduced, and it is proved that in a certain sense any reasonable evolutionary search algorithm can be re-encoded by a binary semi-genetic algorithm (see corollaries 15 and 16). Moreover, an explicit bijection between the set of all such re-encodings and the collection of certain n -tuples of invariant subsets is constructed (see theorem 14). Finally, all possible re-encodings of a given heuristic search algorithm by a classical genetic algorithm are entirely classified in terms of invariant subsets of the search space in connection with Radcliffe's forma (see [9] and theorem 20).

1 Introduction

Over the past 25 years evolutionary algorithms have been widely exploited to solve various optimization problems. In order to apply an evolutionary algorithm to attack a specific optimization problem, one needs to model the problem in a suitable manner. That is, one needs to construct a search space Ω (the set whose elements are all possible solutions to the problem) together with a computable positive valued fitness function $f : \Omega \rightarrow (0, \infty)$ and an appropriate family of “mating” and “mutation” transformations. One can say, therefore, that a representation of a given problem by an evolutionary algorithm is an ordered 4-tuple $(\Omega, \mathcal{F}, \mathcal{M}, f)$ where Ω is the search space, $\mathcal{F}$ is a family of binary operations on Ω and $\mathcal{M}$ is the family of unary transformations on Ω , that is, $\mathcal{M}$ is just a family of functions from Ω to itself. Intuitively $\mathcal{F}$ is the family of mating transformations: every element of $\mathcal{F}$ takes two elements of Ω (the parents) and produces one element of Ω (the child).¹ while $\mathcal{M}$ is the family of mutations (or asexual reproductions) on Ω . For theoretical purposes it is usually assumed that $\mathcal{M}$ is ergodic in the sense that the only invariant subsets under $\mathcal{M}$ are the $\emptyset$ and

¹ In general there is no reason to assume that a child has exactly two parents. All of the results in this paper are valid for the families of m -ary operations on Ω . The only reason $\mathcal{F}$ is assumed to be the family of binary transformations is to simplify the notation.

the entire search space Ω . (The ergodicity assumption ensures that the Markov process modelling the algorithm is irreducible (see, for instance, [4]). A typical

evolutionary algorithm works as follows: A population $P = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_{2m} \end{pmatrix}$ with $x_i \in \Omega$

is selected randomly. The algorithm cycles through the following stages:

Evaluation:

Individuals of P are evaluated:

$$\begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_{2m} \end{pmatrix} \rightarrow \begin{pmatrix} f(x_1) \\ f(x_2) \\ \vdots \\ f(x_{2m}) \end{pmatrix}$$

Selection:

A new population

$$P' = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_{2m} \end{pmatrix}$$

is obtained where $y_i = x_j$ with probability $\frac{f(x_j)}{\sum_{l=1}^{2m} f(x_l)}$.

In other words, all of the individuals of P' are these of P , and the expectation of the number of occurrences of any individual of P in P' is proportional to the number of occurrences of that individual in P times the individual's fitness value. In particular, the fitter the individual is, the more copies of that individual are likely to be present in P' . On the other hand, the individuals having relatively small fitness value are not likely to enter into P' at all. This is designed to imitate the natural survival of the fittest principle.

Partition:

The individuals of P' are partitioned into m pairwise disjoint couples for mating according to some probabilistic rule: For instance the couples could be

$$Q_1 = \begin{pmatrix} y_{i_1^1} \\ y_{i_2^1} \end{pmatrix} \quad Q_2 = \begin{pmatrix} y_{i_1^2} \\ y_{i_2^2} \end{pmatrix} \quad \dots \quad Q_j = \begin{pmatrix} y_{i_1^j} \\ y_{i_2^j} \end{pmatrix} \quad \dots \quad Q_m = \begin{pmatrix} y_{i_1^m} \\ y_{i_2^m} \end{pmatrix}$$

Reproduction:

Replace every one of the selected couples $Q_j = \begin{pmatrix} y_{i_1^j} \\ y_{i_2^j} \end{pmatrix}$ with the couples

$$Q' = \begin{pmatrix} T_1(y_{i_1^j}, y_{i_2^j}) \\ T_2(y_{i_1^j}, y_{i_2^j}) \end{pmatrix}$$

for some couple of transformations $(T_1, T_2) \in \mathcal{F}^2$. The couple (T_1, T_2) is selected according to a fixed probability distribution on $\mathcal{F}^2$. This gives us a new population

$$P'' = \begin{pmatrix} z_1 \\ z_2 \\ \vdots \\ z_{2m} \end{pmatrix}$$

Mutation:

Finally, with small probability we replace z_i with $F(z_i)$ for some randomly chosen $F \in \mathcal{M}$. This, once again, gives us a new population $P''' = \begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_{2m} \end{pmatrix}$

Upon completion of mutation start all over with the initial population P''' . The cycle is repeated a certain number of times depending on the problem. A more general and extensive description is given in [18]. The importance of choosing a reasonable representation for a specific problem is emphasized in some of the modern research. See, for instance, [10]. A few special evolutionary algorithms are introduced in the next section.

2 Special Evolutionary Algorithms

Classical Genetic Algorithm with Masked Crossover:

Let $\Omega = \prod_{i=1}^n A_i$. For every subset $M \subseteq \{1, 2, \dots, n\}$, define a binary operation L_M on Ω as follows:

$$L_M(\mathbf{a}, \mathbf{b}) = (x_1, x_2, \dots, x_i, \dots, x_n)$$

where $\mathbf{a} = (a_1, a_2, \dots, a_n)$ and $\mathbf{b} = (b_1, \dots, b_n) \in S$ and $x_i = \begin{cases} a_i & \text{if } i \in M \\ b_i & \text{otherwise} \end{cases}$.

The reader will recognize L_M as a masked crossover operator with mask M .

Let $\mathcal{F} = \{L_M \mid M \subseteq \{1, 2, \dots, n\}\}$. That is, $\mathcal{F}$ in this example is simply the family of masked crossover transformations. The probability distribution on the set $\mathcal{F}^2$ is concentrated on the pairs of the form $(L_M, L_{\bar{M}})$ where $\bar{M}$ denotes the complement of the set M in $\{1, 2, \dots, n\}$. Most often the pairs are equally likely to be chosen from that diagonal-like subset.

Example: Let $n = 5$ and $A_i = \{0, 1, \dots, i + 1\}$. Suppose a given population P consists of 6 individuals which are the rows of the matrix

$$\begin{pmatrix} 2 & 3 & 4 & 5 & 6 \\ 0 & 1 & 2 & 3 & 4 \\ 1 & 2 & 3 & 4 & 5 \\ 0 & 0 & 1 & 2 & 3 \\ 1 & 1 & 0 & 1 & 2 \\ 1 & 2 & 1 & 5 & 4 \end{pmatrix}$$

Say, after selection stage is complete one obtains the following population

$$P' = \begin{pmatrix} 2 & 3 & 4 & 5 & 6 \\ 2 & 3 & 4 & 5 & 6 \\ 1 & 2 & 3 & 4 & 5 \\ 0 & 0 & 1 & 2 & 3 \\ 0 & 1 & 2 & 3 & 4 \\ 1 & 2 & 3 & 4 & 5 \end{pmatrix}$$

Now the following individuals are paired for mating: (masked crossover in this case)

$$Q_1 = \begin{pmatrix} 2 & 3 & 4 & 5 & 6 \\ 0 & 0 & 1 & 2 & 3 \end{pmatrix}, Q_2 = \begin{pmatrix} 2 & 3 & 4 & 5 & 6 \\ 1 & 2 & 3 & 4 & 5 \end{pmatrix}, \text{ and } Q_3 = \begin{pmatrix} 0 & 1 & 2 & 3 & 4 \\ 1 & 2 & 3 & 4 & 5 \end{pmatrix}$$

Suppose we have chosen the masks $M_1 = \{1, 4, 5\}$, $M_2 = \{1, 2\}$ and $M_3 = \{3, 4\}$ for the crossover of pairs Q_1 , Q_2 and Q_3 respectively. In the language of this paper it means we have chosen the pairs of transformations $(T_{M_1}, T_{\bar{M}_1})$ for Q_1 , $(T_{M_2}, T_{\bar{M}_2})$ for Q_2 and $(T_{M_3}, T_{\bar{M}_3})$ for Q_3 respectively. Upon applying these we obtain

$$Q_1 \rightarrow \begin{pmatrix} T_{M_1}((2, 3, 4, 5, 6), (0, 0, 1, 2, 3)) \\ T_{\bar{M}_1}((2, 3, 4, 5, 6), (0, 0, 1, 2, 3)) \end{pmatrix} = \begin{pmatrix} 2 & 0 & 1 & 5 & 6 \\ 0 & 3 & 4 & 2 & 3 \end{pmatrix}$$

Likewise

$$Q_2 \rightarrow \begin{pmatrix} T_{M_2}((2, 3, 4, 5, 6), (1, 2, 3, 4, 5)) \\ T_{\bar{M}_2}((2, 3, 4, 5, 6), (1, 2, 3, 4, 5)) \end{pmatrix} = \begin{pmatrix} 2 & 3 & 3 & 4 & 5 \\ 1 & 2 & 4 & 5 & 6 \end{pmatrix}$$

and, finally,

$$Q_3 \rightarrow \begin{pmatrix} T_{M_3}((0, 1, 2, 3, 4), (1, 2, 3, 4, 5)) \\ T_{\bar{M}_3}((0, 1, 2, 3, 4), (1, 2, 3, 4, 5)) \end{pmatrix} = \begin{pmatrix} 1 & 2 & 2 & 3 & 5 \\ 0 & 1 & 3 & 4 & 4 \end{pmatrix}$$

The family of mutation transformations, $\mathcal{M}$ in this (and in all of the following examples) consists of the transformations $M_{\mathbf{a}} : \Omega \rightarrow \Omega$ where $\mathbf{a} \in \bigcup_{S \subseteq \{1, 2, \dots, n\}} \prod_{i \in S} A_i$ so that $\mathbf{a} = (a_{i_1}, a_{i_2}, \dots, a_{i_k})$ for $i_1 \leq i_2 \leq \dots \leq i_k \in S_{\mathbf{a}} \subseteq \{1, 2, \dots, n\}$ defined as follows: $\forall \mathbf{x} = (x_1, x_2, \dots, x_n) \in \Omega$ we have

$$M_{\mathbf{a}}(\mathbf{x}) = \mathbf{y} = (y_1, y_2, \dots, y_n) \text{ where } y_q = \begin{cases} a_q & \text{if } q = i_j \text{ for some } j \\ x_q & \text{otherwise} \end{cases} \text{ In other}$$

words, $M_{\mathbf{a}}$ simply replaces the q^{th} coordinate of its argument with $a_q \in A_i$ whenever $q \in S_{\mathbf{a}}$.

Random Respectful Recombination

This type of algorithm first appeared in [9] under the name of Random Respectful Recombination, but it didn't seem to be useful at first². Here the search space Ω and the family of mutation transformations, $\mathcal{M}$, are exactly the same as

² Recently a variation of this technique, known as "gene pool recombination" has been considered in [16], [7] and [19]

in the example of classical genetic algorithm, and the family of mating transformations is described below: In [8] these were named Holland transformations (because their corresponding fixed family of subsets is precisely the collection of subsets of Ω determined by the classical Holland schemata together with the empty set. See examples following corollary 12 in the next section). For every given point $\mathbf{u} = (u_1, u_2, \dots, u_n) \in \Omega$ define a Holland transformation $T_{\mathbf{u}} : \Omega^2 \rightarrow \Omega$ as follows: for every $\mathbf{a} = (a_1, a_2, \dots, a_n)$ and $\mathbf{b} = (b_1, b_2, \dots, b_n) \in S$

$$T_{\mathbf{u}}(\mathbf{a}, \mathbf{b}) = (x_1, x_2, \dots, x_n)$$

where

$$x_i = \begin{cases} a_i & \text{if } a_i = b_i \\ u_i & \text{otherwise} \end{cases}$$

In other words, if the i^{th} coordinates of $\mathbf{a}$ and $\mathbf{b}$ coincide, then the i^{th} coordinate of $T_{\mathbf{u}}(\mathbf{a}, \mathbf{b})$ also coincides with them. If, on the other hand, the i^{th} coordinates of $\mathbf{a}$ and $\mathbf{b}$ differ, then the i^{th} coordinate of $T_{\mathbf{u}}(\mathbf{a}, \mathbf{b})$ is that of $\mathbf{u}$, namely, u_i . Let $\mathcal{F} = \{T_{\mathbf{u}} \mid \mathbf{u} \in S\}$ be the family of all Holland transformations. At every iteration of the algorithm, once a new population P is obtained, a new probability distribution on $\mathcal{F}$ is defined: $T_{\mathbf{u}}$ is chosen from $\mathcal{F}$ so that u_i occurs in $\mathbf{u}$ with the probability proportional to its fitness in P . Every transformation in the pair $(T_{\mathbf{u}}, T_{\mathbf{v}})$ is chosen independently.

Binary Genetic Algorithm with Masked Crossover:

When every $A_i = \{0, 1\}$ (which means that $\Omega = \{0, 1\}^n$) in the example above, one obtains the classical binary genetic algorithm.

Binary Random Respectful Recombination

The search space Ω and the family of mating transformations $\mathcal{F}$ and the family of mutations $\mathcal{M}$ are exactly the same as these for the binary genetic algorithm with masked crossover described above. The only difference is that the probability distribution on $\mathcal{F}^2$ is now completely uniform. (rather than being concentrated on the diagonal-like subset described in the classical genetic algorithm example) For instance, if $n = 5$, $M_1 = \{2, 3, 4\}$, $M_2 = \{1, 3, 5\}$ and the pair (T_{M_1}, T_{M_2}) is selected for mating, we have, for instance,

$$\begin{pmatrix} 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 \end{pmatrix} \mapsto \begin{pmatrix} T_{M_1}((1, 0, 0, 1, 1), (1, 1, 0, 0, 1)) \\ T_{M_2}((1, 0, 0, 1, 1), (1, 1, 0, 0, 1)) \end{pmatrix} = \begin{pmatrix} 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 \end{pmatrix}$$

This type of a binary search algorithm can be classified by the following property: If both parents have a 1 in the i^{th} position then the offspring also has a 1 in the i^{th} position. Likewise, if both parents have a 0 in the i^{th} position then the offspring also has a 0 in the i^{th} position. If, on the other hand, the alleles of the i^{th} gene don't coincide, then the i^{th} allele could be either a 0 or a 1.

The following type of algorithm may seem useless at first. Its importance will become clear in the next section when we present the binary embedding theorem which shows that the binary semi-genetic algorithm (described below) possesses an interesting universal property.

Binary Semi-genetic Algorithm:

The search space $\Omega = \{0, 1\}^n$, just as in the case of the binary genetic algorithm. The family of mating transformations $\mathcal{F}$ is defined as follows: Fix $\mathbf{u} = (u_1, u_2, \dots, u_n) \in \Omega$. Define a semi-crossover transformation $F_{\mathbf{u}} : \Omega^2 \rightarrow \Omega$ as follows: For any given pair $(\mathbf{x}, \mathbf{y}) \in \Omega^2$ with $\mathbf{x} = (x_1, x_2, \dots, x_n)$ and $\mathbf{y} = (y_1, y_2, \dots, y_n)$ we have $F_{\mathbf{u}}(\mathbf{x}, \mathbf{y}) = \mathbf{z} = (z_1, z_2, \dots, z_n) \in \Omega$ where

$$z_i = \begin{cases} 1 & \text{if } x_i = y_i = 1 \\ u_i & \text{otherwise} \end{cases}$$

In other words, $F_{\mathbf{u}}$ preserves the i^{th} gene if it is equal to 1 in all of the rows of P , and replaces it with u_i otherwise. Let $\mathcal{F} = \{F_{\mathbf{u}} \mid \mathbf{u} \in \Omega\}$ be the family of all semi-crossover transformations. The family of mutation transformations $\mathcal{M}$ is exactly the same as in the examples above.

Example: With $n = 5$ and $\mathbf{u}_1 = (0, 1, 1, 0, 1)$, $\mathbf{u}_2 = (0, 1, 0, 0, 1)$ we have

$$\begin{pmatrix} 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 \end{pmatrix} \mapsto \begin{pmatrix} F_{\mathbf{u}_1 2}((1, 0, 0, 1, 1), (1, 1, 0, 0, 1)) \\ F_{\mathbf{u}_2 2}((1, 0, 0, 1, 1), (1, 1, 0, 0, 1)) \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 1 \end{pmatrix}$$

Notice, that if 1 is present in the i^{th} position of both parents, then it remains in the i^{th} position of both offsprings. There are absolutely no other restrictions, though.

In practice the choice of the search space Ω is primarily determined by the specific problem and related circumstances. The general methodology for the construction of the search spaces first appeared in the work of Radcliffe (see, for instance, [9]). Radcliffe introduced the notion of a forma which captures the essential properties of the Holland schemata in a representation independent setting. A forma is simply a partition of the search space into equivalence classes. A given collection of forma with suitable properties (see [9]) is, in a sense, no different from the collection of the classical Holland schemata provided that one encodes the search space using the "genetic representation function" which is also introduced in [9]. The connection between all of the possible families of mating transformations on a given search space Ω and the corresponding families of invariant subsets established in [8] will allow us to extend Radcliffe's notion of the genetic representation function to compare various evolutionary algorithms via possible encodings of their search spaces. This idea will be made clear in the following section.

3 The Binary Embedding Theorem

As we have seen in the introduction, a given evolutionary heuristic search algorithm is determined primarily by the ordered 4-tuple $(\Omega, \mathcal{F}, \mathcal{M}, f)$. In the current paper we shall only be concerned with the search space Ω , the family of mating transformations $\mathcal{F}$ and the family of mutations $\mathcal{M}$. As mentioned in the introduction, the family of mutation transformations is ergodic, meaning that

the only invariant subsets under $\mathcal{M}$ is the $\emptyset$ and the entire search space Ω . This motivates the following definitions:

Definition 1 For a given family of m -ary operations Γ on a set Ω (that is, functions from Ω^m into Ω) a subset $S \subseteq \Omega$ is invariant under Γ if and only if $\forall T \in \Gamma$ we have $T(S^m) \subseteq S$. We shall denote by Λ_Γ the family of all invariant subsets of Ω under Γ . In other words, $\Lambda_\Gamma = \{S \mid S \subseteq \Omega, T(S^m) \subseteq S \forall T \in \Gamma\}$.

Definition 2 A heuristic 3-tuple $\Omega = (\Omega, \mathcal{F}, \mathcal{M})$ is a 3-tuple where Ω denotes an arbitrary set, $\mathcal{F}$ is a family of binary operations on Ω (in other words, a family of functions from Ω^2 to Ω) and $\mathcal{M}$ is a family of unary transformations on Ω (in other words, a family of functions from Ω to itself) such that $\Lambda_{\mathcal{M}} = \{\emptyset, \Omega\}$.

It is easy to verify (see Proposition A1 of [8]) that the family Λ_Γ is closed under arbitrary intersections and contains Ω . It then follows that for every element $x \in \Omega$ there is a unique element of Λ_Γ containing x (namely the intersection of all the members of Λ_Γ containing x .)

Definition 3 Given a heuristic 3-tuple $\Omega = (\Omega, \mathcal{F}, \mathcal{M})$, denote by S_x^Ω the smallest element of $\Lambda_{\mathcal{F}}$ containing x .

The following definition is a natural extension of the notion of a genetic representation function.

Definition 4 Given two heuristic 3-tuples $\Omega_1 = (\Omega_1, \mathcal{F}_1, \mathcal{M}_1)$ and $\Omega_2 = (\Omega_2, \mathcal{F}_2, \mathcal{M}_2)$, a morphism³ $\delta : \Omega_1 \rightarrow \Omega_2$ is just a function $\delta : \Omega_1 \rightarrow \Omega_2$ which respects the mating transformations in the following sense: $\forall T \in \mathcal{F}_1$ and $\forall \mathbf{x} = (x_1, x_2) \in \Omega_1^2 \exists F_{\mathbf{x}} \in \mathcal{F}_2$ such that $\delta(T(x_1, x_2)) = F_{(x_1, x_2)}(\delta(x_1), \delta(x_2))$. Analogously, we must have $\forall M \in \mathcal{M}_1$ and $\forall x \in \Omega_1 \exists H_x \in \mathcal{M}_2$ such that $\delta(M(x)) = H_x(\delta(x))$. We shall denote by $Mor(\Omega_1, \Omega_2)$ the collection of all morphisms from Ω_1 into Ω_2 .

A morphism $\delta : \Omega_1 \rightarrow \Omega_2$ provides the means for encoding the heuristic 3-tuple Ω_1 by the heuristic 3-tuple Ω_2 . Unless the underlying function δ is one to one, there is some nontrivial coarse graining involved. We, therefore have a special name for these morphisms whose underlying functions are injective.

Definition 5 We say that a morphism $\delta : \Omega_1 \hookrightarrow \Omega_2$ is an embedding if the underlying function $\delta : \Omega_1 \rightarrow \Omega_2$ is one-to-one.

Already at this stage one can see the importance of the family of invariant subsets $\Lambda_{\mathcal{F}}$:

Proposition 6 *Let $\delta : \Omega_1 \rightarrow \Omega_2$ be a morphism of heuristic 3-tuples. Then $S \in \Lambda_{\mathcal{F}_2} \implies \delta^{-1}(S) \in \Lambda_{\mathcal{F}_1}$. In words, a preimage of an invariant set under a morphism is invariant.*

³ Heuristic 3-tuples along with the morphisms between them do form a mathematical structure called a Category (see [6] for a detailed exposition). Some properties of the Category of heuristic k -tuples will be presented in the forthcoming paper.

Proof. Fix $S \in \Lambda_{\mathcal{F}_2}$. Let $(x_1, x_2) \in \delta^{-1}(S)$. Then $\forall T \in \mathcal{F}_1$ we have $\delta(T(x_1, x_2)) = F_{(x_1, x_2)}(\delta(x_1), \delta(x_2))$ for some $F_{(x_1, x_2)} \in \mathcal{F}_2$. But $S \in \mathcal{F}_2$ by assumption so that $\delta(T(x_1, x_2)) = F_{(x_1, x_2)}(\delta(x_1), \delta(x_2)) \in S \implies T(x_1, x_2) \in \delta^{-1}(S)$. This shows that $\delta^{-1}(S)$ is, indeed, invariant under $\mathcal{F}_1$.

Although the converse of proposition 6 is not true in general, the mathematical apparatus developed in Appendix A of [8] allows us to establish a partial converse of 6. First we need the notion of a composition closed family which is studied in appendix A of [8]. For the sake of completeness we include the definition below:

Definition 7 We say that a given family of m -ary operations Γ on a set Ω (that is a family of functions from Ω^m to Ω) is composition closed if the following two conditions hold:

1. $\forall T_0, T_1, T_2, \dots, T_m \in \Gamma$ the operation $T : \Omega^m \rightarrow \Omega$ sending any given $\mathbf{x} = (x_1, x_2, \dots, x_m) \in \Omega^m$ to $T(\mathbf{x}) = T_0(T_1(\mathbf{x}), T_2(\mathbf{x}), \dots, T_m(\mathbf{x}))$ is also a member of Γ .
2. $S \subseteq \Omega$ we have $\bigcup_{T \in \Gamma} T(S^m) \subseteq S$.

Remark 8 Notice that if a given family of mating transformations Γ is pure in the sense of [9] (meaning that $\forall T \in \Gamma$ and $\forall x \in \Omega$ we have $T(x, x, \dots, x) = x$. See also [11] and [8]) then condition 2 of definition 7 is satisfied automatically. Every one of the families of mating transformations for the algorithms introduced in section 2 is pure.

It is fairly straightforward to verify that every one of the families of mating transformations involved in the examples of section 2 is composition closed. In fact, it has been already shown in [8] that the families of masked crossover transformations and gene Holland transformations (these which are convenient for modelling random respectful recombination) are composition closed (see proposition 2.1 and Theorem 3.6 of [8]). It only remains to show that the family of semi-crossover transformations is composition closed:

Proposition 9 *The family of binary semi-crossover transformations as defined in the description of the binary semi-genetic algorithm is composition closed.*

Proof. Fix arbitrary $\mathbf{u}, \mathbf{v}, \mathbf{w} \in \Omega = \{0, 1\}^n$. We want to show that the transformation $F : \Omega^2 \rightarrow \Omega$ sending any given pair $\mathbf{z} = (\mathbf{x}, \mathbf{y}) \in \Omega^2$ to $T_{\mathbf{u}}(T_{\mathbf{v}}(\mathbf{z}), T_{\mathbf{w}}(\mathbf{z}))$ is of the form $T_{\mathbf{t}}$ for some $\mathbf{t} \in \Omega$. It is routine to verify using the definition that $\mathbf{t} = T_{\mathbf{u}}(\mathbf{v}, \mathbf{w})$ does the job.

The following fact justifies the importance of definition 7:

Proposition 10 *Let $\Omega_1 = (\Omega_1, \mathcal{F}_1, \mathcal{M}_1)$ and $\Omega_2 = (\Omega_2, \mathcal{F}_2, \mathcal{M}_2)$ denote heuristic 3-tuples with $\mathcal{F}_2$ and $\mathcal{M}_2$ being composition closed. Now given any function $\delta : \Omega_1 \rightarrow \Omega_2$ such that $\forall S \in \Lambda_{\mathcal{F}_2}$ we have $\delta^{-1}(S) \in \Lambda_{\mathcal{F}_1}$, δ is a morphism of heuristic 3-tuples.*

Proof. Fix arbitrary $x_1, x_2 \in \Omega$ and a mating transformation $T \in \mathcal{F}_1$. Our goal is to find a transformation $F \in \mathcal{F}_2$ such that $F(\delta(x_1), \delta(x_2)) = \delta(T(x_1, x_2))$. Consider the smallest element of $\Lambda_{\mathcal{F}_2}$ containing both, $\delta(x_1)$ and $\delta(x_2)$, call it $K_{\{\delta(x_1), \delta(x_2)\}}$. ($K_{\{\delta(x_1), \delta(x_2)\}}$ is simply the intersection of all the members of $\Lambda_{\mathcal{F}_2}$ containing $\delta(x_1)$ and $\delta(x_2)$). Since $\Lambda_{\mathcal{F}_2}$ is closed under arbitrary intersections, $K_{\{\delta(x_1), \delta(x_2)\}} \in \Lambda_{\mathcal{F}_2}$.) Since $K_{\{\delta(x_1), \delta(x_2)\}} \in \Lambda_{\mathcal{F}_2}$, by assumption $\delta^{-1}(K_{\{\delta(x_1), \delta(x_2)\}}) \in \Lambda_{\mathcal{F}_1}$. But then $\delta(T(x_1, x_2)) \in K_{\{\delta(x_1), \delta(x_2)\}}$. Since $\mathcal{F}_2$ is composition closed, by Lemma A.8 of [8], $\exists \in \mathcal{F}_2$ such that $F(\delta(x_1), \delta(x_2)) = \delta(T(x_1, x_2))$ which is exactly what we were after. Notice that condition $\delta(M(x)) = H_x(\delta(x))$ for some $H_x \in \mathcal{M}_2$ is fulfilled automatically since by definition 2 $\Lambda_{\mathcal{M}} = \{\emptyset, \Omega_2\}$ and, by assumption $\mathcal{M}_2$ is composition closed so, by Lemma A.8 of [8], $\forall y \in \Omega_2 \exists H_y \in \mathcal{M}_2$ such that $H_y(\delta(x)) = y$.

As noted before, for any family of m -ary operations on Ω the corresponding family of invariant subsets Λ_Γ is closed under arbitrary intersections. Moreover, for any function $\delta : \Omega_1 \rightarrow \Omega_2$ the inverse image of the intersection of two subsets of Ω_2 is the intersection of the inverse images of these subsets: $\delta^{-1}(U \cap V) = \delta^{-1}(U) \cap \delta^{-1}(V)$. This motivates the following definition:

Definition 11 Given a family of m -ary operations Γ on Ω , we say that a family of subsets $\widetilde{\Lambda}_\Gamma \subseteq \Lambda_\Gamma$ is a base of Λ_Γ if for every $K \in \Lambda_\Gamma$ there exists a collection $\Lambda_K \subseteq \widetilde{\Lambda}_\Gamma$ such that $K = \bigcap_{S \in \Lambda_K} S$. (Equivalently, if $K = \bigcap_{S \in \widetilde{\Lambda}_\Gamma, S \supset K} S$).

Corollary 12 Let $\Omega_1 = (\Omega_1, \mathcal{F}_1, \mathcal{M}_1)$ and $\Omega_2 = (\Omega_2, \mathcal{F}_2, \mathcal{M}_2)$ denote heuristic 3-tuples with $\mathcal{F}_2$ and $\mathcal{M}_2$ being composition closed, and $\delta : \Omega_1 \rightarrow \Omega_2$ be a function. Then the following are equivalent:

1. $S \in \widetilde{\Lambda}_{\mathcal{F}_2} \implies \delta^{-1}(S) \in \Lambda_{\mathcal{F}_1}$.
2. $S \in \Lambda_{\mathcal{F}_2} \implies \delta^{-1}(S) \in \Lambda_{\mathcal{F}_1}$.
3. $\delta : \Omega_1 \rightarrow \Omega_2$ is a morphism of heuristic 3-tuples.

Proof. An immediate consequence of propositions 6 and 10 together with the discussion preceding definition 11.

Below we list the families of invariant subsets together with a naturally chosen bases for each of the examples presented in section 2.

Classical Genetic Algorithm. In this case, the family of invariant subsets $\Lambda_{\mathcal{F}} = \{\prod_{i=1}^n T_i \mid T_i \subseteq A_i\}$. This is precisely the family of subsets determined by Antonisse's schemata (see corollary 2.4 of [8]). A bases for $\Lambda_{\mathcal{F}}$ is the family $\widetilde{\Lambda}_{\mathcal{F}} = \{\prod_{i=1}^n T_i \mid T_i = A_i \text{ for all but one } i\}$. The reader can see that $|\widetilde{\Lambda}_{\mathcal{F}}| = 2^{\sum_{i=1}^n |A_i|}$. Every element of $\widetilde{\Lambda}_{\mathcal{F}}$ can be thought of as a union of subsets determined by the Holland schemata having exactly one fixed position at the same gene.

Random Respectful Recombination. $\Lambda_{\mathcal{F}} = \{\prod_{i=1}^n T_i \mid T_i = \{a_i\} \text{ or } T_i = A_i\} \cup \{\emptyset\}$. This is precisely the family of subsets determined by the Holland schemata together with the empty set (see corollary 3.5 of [8]). A bases for $\Lambda_{\mathcal{F}}$ is the family $\widetilde{\Lambda}_{\mathcal{F}} = \{\prod_{i=1}^n T_i \mid \exists! \text{ with } T_j = \{a_j\}. \text{ For } i \neq j \text{ } T_i = A_i\}$. This is

precisely the family of subsets determined by Holland schemata having exactly one fixed position.

Binary Semi-genetic Algorithm. It is not hard to verify that $\Lambda_{\mathcal{F}} = \{\prod_{i=1}^n T_i \mid T_i = \{1\} \text{ or } T_i = \{0, 1\} \cup \{\emptyset\}\}$. This is precisely the family of subsets determined by Holland schemata whose fixed positions can only equal to 1 (can't be equal to 0). A bases for $\Lambda_{\mathcal{F}}$ is the family $\Lambda_{\mathcal{F}} = \{\prod_{i=1}^n T_i \mid \exists! \text{ with } T_j = \{1\} \text{ for } j = i \text{ and } T_i = \{0, 1\} \text{ for } i \neq j\}$ which is precisely the family of subsets determined by Holland schemata having exactly one fixed position, and that fixed position is equal to 1.

Corollary 12 allows us to characterize all possible morphisms from various heuristic 3-tuples to the standard types described in section 2. Our first result, the Binary Embedding Theorem, establishes an explicit one-to-one correspondence between the set of all embeddings of a given heuristic 3-tuple into a binary semi-genetic algorithm of length n and a certain collection of ordered n -tuples of Ω -invariant subsets.

Definition 13 Fix any heuristic 3-tuple $\Omega = (\Omega, \mathcal{F}, \mathcal{M})$. We say that collection

$$\Upsilon_n = \{\mathcal{I} \mid \mathcal{I} = (I_1, I_2, \dots, I_n) \text{ } I_j \in \Lambda_{\Omega}, \forall x, y \in \Omega \text{ with } x \neq y \exists 1 \leq j \leq n$$

such that either $(x \in I_j \text{ and } y \notin I_j)$ or vise versa: $(y \in I_j \text{ and } x \notin I_j)\}$

is a family of separating n -tuples. Notice that $\Upsilon_n \subseteq (\Lambda_{\mathcal{F}})^n$.

Theorem 14 Fix a heuristic 3-tuple $\Omega = (\Omega, \mathcal{F}, \mathcal{M})$. We now have the following bijection $\phi : \Lambda_{\mathcal{F}} \rightarrow \text{Mor}(\Omega, \mathcal{S}_n)$ (here $\mathcal{S}_n$ denotes the binary semi-genetic heuristic 3-tuple with the search space $\{0, 1\}^n$, see also definition 4 for the meaning of $\text{Mor}(\Omega, \mathcal{S}_n)$) which is defined explicitly as follows: Given an ordered n -tuple of sets from Λ_{Ω} , call it $\mathcal{I} = (I_1, I_2, \dots, I_n) \in (\Lambda_{\mathcal{F}})^n$ let $\phi(\mathcal{I}) = \delta_{\mathcal{I}}$ where

$$\delta_{\mathcal{I}}(x) = (x_1, x_2, \dots, x_n) \in S = \{0, 1\}^n \text{ with } x_j = \begin{cases} 1 & \text{if } x \in I_j \quad \forall x \in \Omega. \\ 0 & \text{otherwise} \end{cases}$$

over, $\delta_{\mathcal{I}}$ is an embedding (injective) if and only if $\mathcal{I} \in \Upsilon_n$ (see definition 13). In other words, the restriction of ϕ to Υ_n is a bijection onto the collection of all embeddings of Ω into $\mathcal{S}_n$.

Proof. Given any map $\delta : \Omega \rightarrow \{0, 1\}$, for $1 \leq j \leq n$ let $I_j = \delta^{-1}(\prod_{i=1}^n T_i)$ with $T_i = \{0, 1\}$ if $i \neq j$ and $T_j = \{1\}$. Recall from examples following definition 11 that $\{\prod_{i=1}^n T_i \mid \exists! \text{ with } T_j = \{1\} \text{ for } j = i \text{ and } T_i = \{0, 1\} \text{ for } i \neq j\}$ forms a bases for the family of subsets invariant under semi-crossover transformations. Therefore, according to corollary 12, $\delta : \Omega \rightarrow \mathcal{S}_n$ is a morphism of heuristic 3-tuples if and only if $(I_1, I_2, \dots, I_n) \in (\Lambda_{\mathcal{F}})^n$. This shows that $\phi : \Lambda_{\mathcal{F}} \rightarrow \text{Mor}(\Omega, \mathcal{S}_n)$ is a well-defined bijection. It is routine to check that $\delta_{\mathcal{I}}$ is injective if and only if $\mathcal{I} \in \Upsilon_n$ (see definition 13).

It turns out that the conditions under which a given heuristic 3-tuple can be embedded into a binary semi-genetic heuristic 3-tuple are rather mild and naturally occurring as the following two corollaries demonstrate:

Corollary 15 *Given a heuristic 3-tuple $\Omega = (\Omega, \mathcal{F}, \mathcal{M})$, the following are equivalent:*

1. Ω can be embedded into an n -dimensional semi-genetic heuristic k -tuple for some n .
2. $\forall x, y \in \Omega$ with $x \neq y$ we have either $x \notin S_y^\Omega$ (see definition 3) or vice versa: $y \notin S_x^\Omega$.
3. $\forall x, y \in \Omega$ with $x \neq y$ we have $S_x^\Omega \neq S_y^\Omega$. (Another way to say this, is that the map sending x to S_x^Ω is one-to-one.)

Moreover, if an embedding exists for some n , then there exists one for $n = |\Omega|$. We also must have $n \geq \lceil \log_2 |\Omega| \rceil$.

Proof. One simply shows that $\forall x, y \in \Omega$ with $x \neq y$ we have either $x \notin S_y^\Omega$ or $y \notin S_x^\Omega$ if and only if $|\Omega|$ -tuple $\mathcal{S} = (S_{x_1}^\Omega, S_{x_2}^\Omega, \dots, S_{x_{|\Omega|}}^\Omega)$ where $\{x_i\}_{i=1}^n$ is an enumeration of all the elements of Ω is separating (i. e. $\mathcal{S} \in \Upsilon_n$, see definition 13) if and only if $\Upsilon_n \neq \emptyset$ which, in turn, according to theorem 14, happens if and only if Ω can be embedded into an n -dimensional semi-genetic heuristic k -tuple for some n . This establishes the equivalence of 1 and 2. Clearly 2 implies 3. To see the converse, we show that “Not 2” implies “Not 3”. Indeed, if $x \in S_y^\Omega$ and $y \notin S_x^\Omega$, then, by minimality, (see definition 2) we have $S_x^\Omega \subseteq S_y^\Omega$ and $S_y^\Omega \subseteq S_x^\Omega$, so that $S_x^\Omega = S_y^\Omega$.

Corollary 16 *Given a heuristic 3-tuple $\Omega = (\Omega, \mathcal{F}, \mathcal{M})$, if for every $T \in \mathcal{F}$, T is pure in the sense of [9] (in other words, $\forall x \in \Omega T(x, x) = x$) then Ω can be embedded into a binary semi-genetic heuristic k -tuple of dimension less than or equal to $|\Omega|$.*

Proof. The desired conclusion follows immediately from corollary 15 by observing that $\forall x, y \in \Omega$ with $x \neq y$ we have $S_x^\Omega = \{x\}$ so that $x \in \{x\} = S_x^\Omega$ while $y \notin \{x\} = S_x^\Omega$.

Notice that purity by itself is sufficient for the existence of an embedding of a given heuristic 3-tuple into a binary semi-genetic heuristic 3-tuple. Of course, the embedding may not be surjective by any means. The main virtue of theorem 14 is not so much the results such as corollary 16, but rather the explicit bijective correspondence between $Mor(\Omega, \mathcal{S}_n)$ and the collection $(\Lambda_{\mathcal{F}})^n$. The main tool involved in the proof of theorem 14 is corollary 12. In the next section we demonstrate how corollary 12 can be applied to establish a similar bijective correspondence between certain kinds of sequences of Radcliffe’s forma and $Mor(\Omega, \mathcal{G}_{\{A_i\}_{i=1}^n})$ where $\mathcal{G}_{\{A_i\}_{i=1}^n}$ denotes the heuristic 3-tuple corresponding to the genetic algorithm with $\prod_{i=1}^n A_i$ as its underlying search space.

4 Characterizing the Morphisms from a Given Heuristic 3-Tuple into a Genetic Heuristic 3-Tuple in Terms of Radcliffe’s Forma

For the reader’s convenience we restate a few basic notions considered in [9]:

Definition 17 Given a set Ω , denote by $\mathbb{E}(\Omega)$ the set of all possible partitions of Ω into disjoint nonempty subsets. (Partitions and equivalence relations are in a natural bijective correspondence. See, for instance [9], or any standard textbook on basic mathematical structures and concepts for details) Given an element $\Xi \in \mathbb{E}(\Omega)$, the elements of Ξ are called forma. Given an n -tuple $\Psi = (\Xi_1, \Xi_2, \dots, \Xi_n)$ of elements of $\mathbb{E}(\Omega)$, let $\Xi(\Psi) = \prod_{i=1}^n \Xi_i$. A genetic representation function $\rho : \Omega \rightarrow \Xi(\Psi)$ sends a given $x \in \Omega$ to $(X_1, X_2, \dots, X_n) \in \Xi(\Psi)$ where $x \in X_i$ (remember that such an $X_i \in \Xi_i$ exists and is unique since Ξ_i is a partition of Ω so that ρ is, indeed, well-defined).

The following definition sets the stage for the main theorem of this section:

Definition 18 Given a heuristic 3-tuple $\Omega = (\Omega, \mathcal{F}, \mathcal{M})$, and a function $\delta : \Omega \rightarrow \prod_{i=1}^n A_i$, let $\Psi_\delta = (X_1^\delta, X_2^\delta, \dots, X_n^\delta)$ where X_j is the collection of all nonempty preimages under δ of the subsets of $\prod_{i=1}^n A_i$ which are determined by the classical Holland schemata having exactly one fixed position in the j^{th} gene. Explicitly, $X_j^\delta = \{\delta^{-1}(\prod_{i=1}^n T_i) \mid T_j = \{a_j\} \text{ for some } a_j \in A_j \text{ and } T_i = A_i \text{ for } i \neq j\} - \{\emptyset\}$.

Definition 19 To shorten the notation we shall denote by $\mathcal{G}_{\{A_i\}_{i=1}^n}$ the heuristic 3-tuple representing the classical genetic algorithm, and by $\mathcal{P}_{\{A_i\}_{i=1}^n}$ the heuristic 3-tuple representing the random respectful recombination algorithm with the underlying search space $\prod_{i=1}^n A_i$.

The following theorem is an immediate consequence of corollary 12:

Theorem 20 *Given a heuristic 3-tuple $\Omega = (\Omega, \mathcal{F}, \mathcal{M})$, and a function $\delta : \Omega \rightarrow \prod_{i=1}^n A_i$, the following are true:*

1. $\delta : \Omega \rightarrow \mathcal{P}_{\{A_i\}_{i=1}^n}$ is a morphism of heuristic 3-tuples if and only if $\forall 1 \leq j \leq n$ every forma in X_j^δ (see definition 18) is invariant under $\mathcal{F}$ (if and only if every forma in X_j^δ is a member of $\Lambda_{\mathcal{F}}$).
2. $\delta : \Omega \rightarrow \mathcal{G}_{\{A_i\}_{i=1}^n}$ is a morphism of heuristic 3-tuples if and only if $\forall 1 \leq j \leq n$ every union of forma in X_j^δ (see definition 18) is invariant under $\mathcal{F}$ (if and only if for every subset of forma $Y \subseteq X_j^\delta$ we have $(\bigcup_{S \in Y} S) \in \Lambda_{\mathcal{F}}$).

Proof. In case of a random respectful recombination all forma in X_j^δ , and, in case of a classical genetic algorithm, all unions of forma in X_j^δ are precisely the preimages under δ of the sets in $\widetilde{\Lambda}_\Gamma$ where Γ is the family of Holland transformations in case of random respectful recombination, and the family of masked crossover transformations in case of a classical genetic algorithm (see examples following corollary 12). The desired conclusion now follows at once from corollary 12.

The difference between theorem 20 and results like theorem 25 of [9] is that theorem 20 classifies all possible re-encodings of a given evolutionary search algorithm in terms of a given genetic algorithm (or in terms of a “random respectful recombination”) while Radcliffe’s results provide a foundation for designing a genetic algorithm to model a specific problem in question.

5 Conclusions

In the current paper the following contributions have been made:

1. An appropriate notion for comparing evolutionary computation techniques via their representation (a morphism between heuristic 3-tuples) has been introduced. (See definition 4)
2. An important connection between the family of invariant subsets of the search space (see definition 1) and the morphisms of heuristic 3-tuples has been established (see corollary 12).
3. A binary semi-genetic algorithm has been introduced and it was shown that virtually any evolutionary heuristic search algorithm can be embedded into a binary semi-genetic algorithm (see theorem 14 and corollaries 15 and 16).
4. All possible morphisms (re-encodings and coarse graining) of a particular evolutionary heuristic search tuple by a classical genetic algorithm, or by a random respectful recombination have been characterized in terms of Radcliffe's forma (see theorem 20).

Acknowledgements. I want to thank Professor John Holland for the helpful discussions and for the encouragement I've received from him to write this paper. I also want to thank my thesis advisor, Professor Andreas Blass for the numerous helpful advisor meetings which have stimulated many ideas for this and for my future work. In addition I would like to thank my fellow graduate student of mathematics, Ronald Walker and the University of Michigan Complex Systems Group for helpful discussions. Finally, I want to thank Jon Rowe and the other anonymous referees for the helpful suggestions and remarks they have made.

References

1. Antonisse, J. (1989). A new interpretation of schema notation that overturns the binary encoding constraint. In Ed. J. D. Schaffer *Proceedings of the Third International Conference on Genetic Algorithms*, Morgan Kaufmann, pages 86–97.
2. Booker, L. (1993). Recombination distributions for genetic algorithms. In L. Darrell Whitley, editor, *Foundations of Genetic Algorithms 2*, pages 29–44, Morgan Kaufmann.
3. Geiringer, H. (1944). On the probability of linkage in Mendelian heredity. *Annals of Mathematical Statistics*, 15:25–57.
4. Coffey, S. (1999) An Applied Probabilist's Guide to Genetic Algorithms. *A Thesis Submitted to The University of Dublin for the degree of Master in Science*.
5. Liepins, G. and Vose, M. (1992). Characterizing Crossover in Genetic Algorithms. *Annals of Mathematics and Artificial Intelligence*, 5: 27–34.
6. Mac Lane, S. (1971) Categories for the working mathematician. *Graduate Texts in Mathematics 5*, Springer-Verlag.
7. Mühlenbeim, H. and Mahnig, T. (2001). Evolutionary computation and beyond. In Y. Uesaka, P. Kanerva, and H. Asoh, editors, *Foundations of Real-World Intelligence*, CSLI Publications, pp. 123–188.

8. Mitavskiy B. (Resubmitted). Crossover Invariant Subsets of the Search Space for Evolutionary Algorithms. *Evolutionary Computation*.
<http://www.math.lsa.umich.edu/~bmitavsk/>
9. Radcliffe, N. (1992). The algebra of genetic algorithms. *Annals of Mathematics and Artificial Intelligence*, 10:339–384.
10. Rothlauf F., Goldberg D., Heinzl A. (2002). Network random keys - a tree representation scheme for genetic and evolutionary algorithms. *Evolutionary Computation*, 10(1): 75–97.
11. Rowe, J., Vose, M., and Wright, A. (2002). Group properties of crossover and mutation. *Evolutionary Computation*, 10(2): 151–184.
12. Stephens, C., Poli, R., Wright, A., and Rowe, J. (2002). Exact results from a coarse grained formulation of the dynamics of variable-length genetic algorithms. In W. B. Langdon, E. Cantu-Paz, K. Mathias, R. Roy, D. Davis, R. Poli, K. Balakrishnan, V. Honavar, G. Rudolph, J. Wegener, L. Bull, M. A. Potter, A. C. Schultz, J. F. Miller, E. Burke, and N. Jonoska, editors, *GECCO 2002: Proceedings of the Genetic and Evolutionary Computation Conference*, pages 578–585, Morgan Kaufmann Publishers.
13. Stephens, C. (2001). Some exact results from a coarse grained formulation of genetic dynamics. *Proceedings of the Genetic and Evolutionary Computation (GECCO) conference*, pages 631–638, Morgan Kaufmann.
14. Stephens, C. and Waelbroeck, H. (1999). Schemata evolution and building blocks. *Evolutionary Computation*, 7(2):109–124.
15. C. Stephens. (2002). The Renormalization Group and the Dynamics of Genetic systems, to be published in *Acta Physica Slovaca*. <http://arXiv.org/abs/cond-mat/0210271/>.
16. Syswerda, G. (1993). Simulated crossover in genetic algorithms. In L. Darrel Whitley, editor, *Foundations of Genetic Algorithms 2*, Morgan Kaufmann.
17. Vose, M. (1991). Generalizing the notion of a schema in genetic algorithms. *Artificial Intelligence*, 50(3): 385–396.
18. Vose, M. (1999). The simple genetic algorithm: foundations and theory. *MIT Press, Cambridge, Massachusetts*.
19. Wright, A., Rowe, J., Poli, R., and Stephens C. (2002). A fixed point analysis of a gene pool GA with mutation. *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)* Morgan Kaufmann.
<http://www.cs.umd.edu/u/wright/>.

Dispersion-Based Population Initialization

Ronald W. Morrison

Mitretek Systems, Inc.
3150 Fairview Park Drive South
Falls Church, VA 22043-4519
ronald.morrison@mitretek.org

Abstract. Reliable execution and analysis of an evolutionary algorithm (EA) normally requires many runs to provide reasonable assurance that stochastic effects have been properly considered. One of the first stochastic influences on the behavior of an EA is the population initialization. This has been recognized as a potentially serious problem to the performance of EAs but little progress has been made in improving the situation. Using a better population initialization algorithm would not be expected to improve the many-run average performance of an EA, but instead, it would be expected to reduce the variance of the results, without loss of average performance. This would provide researchers the opportunity to reliably examine their experimental results while requiring fewer EA runs for an appropriate statistical sample. This paper uses recent advances in the measurement and control of a population's dispersion in a search space to present a novel algorithm for better population initialization. Experimental verification of the usefulness of the new technique is provided.

1 Introduction

Execution and analysis of an evolutionary algorithm (EA) normally requires many runs to provide reasonable assurance that any negative effects of merely “bad luck” in the stochastic processes have been overcome. One of the first stochastic influences on the behavior of an EA is the population initialization. This has been recognized as a potentially serious problem to the performance of EAs in [1], and, to a lesser extent, in [2], but little progress has been made in improving the situation. In the few cases where this situation is addressed at all, populations, in very low dimensionality problems, have been initialized using techniques like the Latin Hypercube [2]. The Latin Hypercube technique guarantees uniform placement along each axis, but uniform placement along individual axes does not ensure any level of uniformity throughout the search space.

EAs are generally executed many times to overcome stochastic anomalies, so using a better population initialization algorithm would not be expected to improve the average performance of an EA. Instead, it would be expected to reduce the variance of the results, without loss of average performance. This

would provide researchers the opportunity to reliably examine their experimental results while requiring fewer EA runs for an appropriate statistical sample.

This paper provides a description of the population initialization problem, an introduction to low-discrepancy sequences, the derivation of a new and computationally efficient population initialization algorithm, and experimental results that demonstrate the usefulness of the algorithm.

2 Problem Description

We desire to improve the search capability of the initial population by spreading the individual members throughout the search space so that their positioning maximizes the probability of detection of a landscape feature of interest. Let's take a deeper look at more precisely what this means.

In the general case, for any specific population size, P , in N dimensions, this is equivalent to solving an N -dimensional sphere packing problem. The problem, however, is expressed in terms of the packing space size and the number of spheres, and must be solved for the sphere radius and the coordinates of the packed spheres. The sphere center coordinates would then be the population positions. While much research has been done in sphere packing problems [3] [4], efficient methods for solving the above problem have not been identified. Note that we would also like to avoid adding significant computational complexity to our EA.

The population placement problem becomes even more difficult if we wish to use a variety of population sizes. We would like the ability to initialize different sized population for a problem without needing to re-perform a complex population initialization algorithm. We would like, therefore, to be able to appropriately add or delete population members to an initialization distribution that we have already computed.

The identification of the largest unoccupied volume for the placement of the "next" population member in N -space already occupied by a population is an interesting problem. The normal N -dimensional space partitioning techniques such as Delaunay triangulation [5], or other space partitioning methods involving polytopes, do not assist in the identification of the largest unoccupied N -dimensional volume. Application of these techniques to our problem could result in an unintentional division of the largest unoccupied volume by inappropriate space partitioning edge placement, rather than point placement.

Another area of research that could be used for placing points in N -dimensional space includes meshing techniques such as t-m-s nets [6]. As a powerful mechanism for placing points uniformly in N -dimensional space, t-m-s nets cover space in base b in dimension N , requiring b^N points to assure uniform spatial coverage. Since the process starts by partitioning the space into b^N cells and we wish to use a varying number of points, the mesh-generating techniques are generally inappropriate for our application.

Since we are trying to place all population members such that the distance to a any point in the search space is minimized, ideally we would like to place an

additional population member into an existing population at the mid-point of the largest unoccupied symmetrical volume. What we are in need of is a computationally efficient algorithm that is better than a random number generator in efficiently placing a population uniformly throughout a search space, occupying unoccupied volumes first. The specialty field in mathematics that has done some exploration into this type of problem is called low-discrepancy sequence generation [7]. The key idea in this field of mathematics is discrepancy measurement, so first we will provide a short introduction to discrepancy measurement.

3 Discrepancy Measurement

Discrepancy measurement identifies how uniformly a set of points sample a search space. Discrepancy measurement involves concepts that are loosely based on the same ideas as the Kolmogorov-Smirnov (K-S) test, when used to compare a distribution of points to the uniform distribution. The K-S test compares an ordered set of P points X_i to a continuous distribution by computing the difference between the discrete cumulative distribution of the points with cumulative distribution of the desired continuous distribution. The hypothesis regarding the distributed form of the points is rejected if a test statistic, D , is greater than the critical value.

Unfortunately, the cumulative probability distribution is not well defined in more than one dimension. So for higher dimensions the common measure of spatial uniformity is discrepancy. While not providing a test statistic for uniformity, discrepancy can be used to compare sequences of points in N -space to determine which sequence is more uniform. Discrepancy measures consider the N -dimensional $\mathbb{R}^N$ space modulo 1 or, equivalently, the N -dimensional torus $T^N = \mathbb{R}^N / \mathbb{Z}^N$. Instead of identifying the largest difference between cumulative distributions of points and the desired distribution, as the K-S test does, discrepancy identifies the largest difference between the proportion of points included in an N -dimensional space partition and the proportion of the total volume (Lebesgue measure) included in that partition, over all possible space partitions.

Discrepancy, $\mathbb{D}_P$, of a sequence of points, $(x_p)_{p \geq 1}$, $x_p \subseteq \mathbb{R}^N$, as further elaborated in [7], is useful for comparing the uniformity sequences, because a sequence is uniformly distributed modulo 1 if and only if:

$$\lim_{P \rightarrow \infty} \mathbb{D}_P(x_P) = 0. \quad (1)$$

As a mathematically sound method for computing the uniformity of a distribution of points in an N -dimensional space, it is unfortunate that discrepancy measurement is generally too computationally expensive to be practically employed in most EA problems involving assessment of the dispersion of points in a search space. Recently [8] developed a computationally efficient method for measuring a population's dispersion throughout a search space, called the dispersion index, Δ . This measure is based on the concepts of discrepancy theory and provides practical improvements over the more commonly used diversity measurements.

Applying discrepancy theory concepts to our population initialization problem, we turn to research in the generation of low-discrepancy sequences. This research is focused on methods for placing points uniformly in multi-dimensional spaces. Most of the research into low-discrepancy sequences is involved in establishing upper and lower bounds on the discrepancy of various sequence generation algorithms, through algorithmic proofs or example construction.

The research in this field yields sequences with the smallest values of discrepancy (in the limit) currently known [9] [10]. There is, however, no evidence that these sequences will perform well in an application such as ours, where only a small number of points near the beginning of the sequence will actually be used. Additionally, our application has computational efficiency requirements, and none of the research appears to adequately address this requirement. In the following sections, applying the low-discrepancy sequence generation methods, we derive a computationally efficient, heuristic sequence generation algorithm for an N -dimensional space that has the attributes we desire.

3.1 Population Placement Basis

For the construction of our population placement algorithm, we used some of the research on the $k\alpha$ sequence for an irrational α . This is a well-studied sequence and is known to be uniformly distributed modulo 1 [7].

This sequence has been extended into N -dimensions, through what is known as the Kroenecker sequence [10]. The Kroenecker sequence is based on the following theorem, from [7]. Let $\beta_1, \dots, \beta_N \in \mathbb{R}$. Then:

Theorem 1:

The N -dimensional sequence: $x_k = (k\beta_1, \dots, k\beta_N)$ is uniformly distributed modulo 1 if and only if $1, \beta_1, \dots, \beta_N$ are linearly independent over the integers $\mathbb{Z}$.

This is an extension of the $k\alpha$ sequence since, for $\beta_1 \dots \beta_N$ to be linearly independent over the integers, there are no integers $m_i \in \mathbb{Z}$ such that $1, m_1\beta_1 + \dots + m_N\beta_N = 0$. In other words, the β 's must be irrational. So, using our notation convention, we will refer to these numbers in future references as α 's (i.e., $k\alpha_1, \dots, k\alpha_N$).

The mathematical study of low-discrepancy sequences is usually focused on the sequence discrepancy in the limit for large numbers of points. We, however, aren't really interested in theoretical convergence for large sequences. We are interested in the uniformity of the distribution of the first few points (perhaps as high as several hundred) of the sequence. The extreme computational efficiency of the Kroenecker sequence, once the α 's are identified, makes it very attractive for our application.

So our contribution here is to devise a set of rules for using the concepts of the Kroenecker sequence to construct a reasonably uniformly distributed set of points in N -space for the range of dimensions and range of number of points of interest for EA applications. We will then use those rules to derive a set of α 's that EA practitioners and researchers can use. To understand the problem, we

will first illustrate that the Kroenecker sequence does not, by itself, necessarily provide the desired uniformity of the initial points in a sequence. Figure 1 illustrates one type of non-uniform initial point distributions that can be generated a by badly chosen, but irrational, α 's in two dimensions. This sequence will be uniformly distributed in the long run, but is a very bad choice with respect to the initial few points.

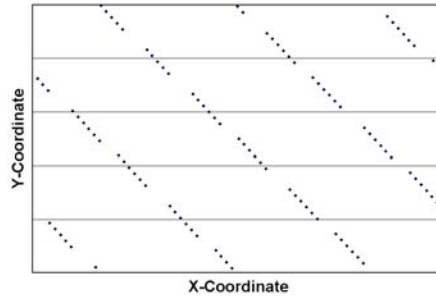


Fig. 1. 50 Kroenecker Sequence Points #1

4 Heuristic Rules

What we need is “good mixing” for the first few hundred points of the sequence. To accomplish this, we have developed a set of heuristic rules to choose α 's with good inter-dimensional mixing. The rules will first be delineated, then the reasons for each of these heuristics will be described.

1. Base the incremental intervals for each dimension on multiples of the “Golden Ratio” which is: $\phi = \frac{\sqrt{5}+1}{2}$.
2. Select the multiples of ϕ for each dimension from sequences of prime numbers.
3. Examine the resultant modulo 1 sequences for the quasi-period with which they revisit a value near the initial value (this is done by examining the sequential distance difference as described below). If a multiplier results in either a very short or trivial sequence of sequential differences, or closely matches the sequential difference sequence of a previously selected multiplier, do not use that multiplier, use the next prime number instead.

The reasons for these heuristics are fairly straightforward. The first rule is based on the fact that to achieve good mixing in the initial points of the sequence, we need a “very” irrational number. Irrational numbers are classified by how easy they are to approximate with continued-fraction ratios of integers. For example, $\sqrt{2}$, which is irrational, would be inappropriate to use since it is well approximated by a continued fraction sequence.

The “Golden Ratio” mentioned previously, normally represented by ϕ , is the irrational number that is most poorly approximated through continued fractions [11]. This is the obvious choice for uniformity of a one-dimensional placement, since the more irrational α is, the more uniformly distributed is the first part of the sequence of P points $P_i = \{K_i\alpha\}$, where $K_i = 1, 2, 3, \dots, P$ [10]. However, the use of ϕ only guarantees best placement in one dimension. If the same α (or any integer multiple of the same α) is used for a Kronecker sequence in more than one dimension, the points are merely placed along a diagonal in those dimensions. This is why the Kronecker sequence requires linear independence of the α 's among $\mathcal{Z}$. Since it can be very difficult to prove that a number is irrational or that multiple irrational numbers are linearly independent over $\mathcal{Z}$, this brings us to the second heuristic rule.

The second rule is to select the multiples of ϕ for each dimension from sequences of prime numbers, avoiding the prime numbers 1 and 2. What we need to accomplish here is to force the pair-wise relationship between dimensions to be irrational. Ratios of prime numbers approximate irrational numbers and this method makes the ratios of the step sizes between any two dimensions relatively irrational.¹ This approximation is adequate for our purposes, as long as the sequences generated provide good placement and good inter-dimensional mixing. Since, however, multiples of ϕ do not necessarily have the same placement uniformity as ϕ , we come to the third heuristic rule.

This third heuristic rule is designed to improve mixing across multiple dimensions. In a sequence $kZ_1\alpha$ modulo 1, for $k = 1, 2, 3, \dots$ and $Z_i \in \mathcal{Z}$, values within a distance ε of the first point are visited quasi-periodically, with the sequential differences between these quasi-periods forming an easily recognizable pattern, but with occasional and irregular breaks in the pattern. An example of this is shown as Figure 2. In this figure, the distance to the starting point of a Kronecker sequence is plotted for 50 points. As can be seen, for $\varepsilon = .10$ the distance is less than ε at $k = 6, 14, 18, 22, 26, 31, 35, 39, 43, 48, \dots$. The sequential difference between the k values are 8, 4, 4, 4, 5, 4, 4, 4, 4, ... When these sequential differences form trivial patterns like this (or, for example, 3, 3, 3, 3, 3, 5, 3, 3, 3, 3, 3, 5), or are identical to the patterns already selected for other prime multipliers, the fill patterns across multiple dimensions are usually not adequately mixed. These sequences are easily checked using a simple spreadsheet, and, if an undesired sequence is encountered, the next prime number in the sequence should be selected as the multiplier.

Table 1 provides recommended α values for 1 through 12 dimensions that were derived using the heuristic rules described above. The Table also provides the Dispersion Index, Δ from [8] for populations of the first 50 points generated using these α 's. The Dispersion Index, Δ measures the uniformity of the population dispersion and has a maximum value of 1.00 for a completely uniform distribution. While the recommended α 's in Table 2 are not guaranteed to be

¹ Sequential very large prime numbers must also not be used, since the ratio of two sequential very large prime numbers closely approximates 1.0 and will result in a violation of the next heuristic rule.

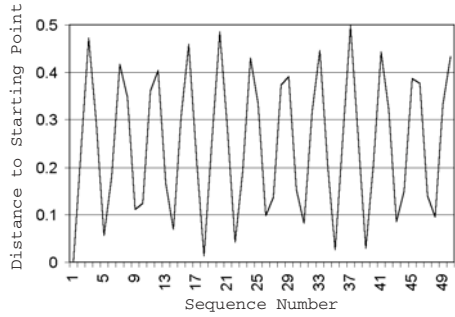


Fig. 2. Quasi-Periodic Distance to the First Point

the optimal α 's for use with any specific EA problem, these prime numbers are known to provide reasonably uniform spatial distributions for the first 200 points in dimensions up through 12.

Table 1. Prime α 's and Dispersion Indices for 50 Points, Dimensions 1 through 12

Dimension	α	Δ
1	41	0.9871
2	43	0.9841
3	47	0.9702
4	59	0.9874
5	83	0.9738
6	107	0.9795
7	109	0.9853
8	173	0.9853
9	311	0.9853
10	373	0.9910
11	401	0.9999
12	409	0.9910

The resultant heuristic sentinel placement algorithm works as follows. First scale and offset all search-space dimensions as necessary to set each dimensional range from 0 to 1. Second, randomly place the initial sentinel point, x_i , within the search space, for i equal 1 to N . Finally, compute the coordinates of each subsequent point P , $x_{i,p} \in P$, as:

$$x_{i,p} = \text{mod1}[x_{i,p-1} + Z_i\phi], \tag{2}$$

where the Z_i 's are selected for each dimension in accordance with the heuristic rules specified.

As can be seen, the coordinates of each point can easily (and very computationally efficiently) be determined from the location of the previous point, once

the Z_i 's are selected through an off-line process for the range of dimensions and population sizes appropriate for your problem domain. Note that nothing more need be known about the problem space than the number of dimensions and the approximate size of the population of sentinels to be used, and that this information is used off-line in a simple spreadsheet calculation.

4.1 Population Placement in Very High-Dimensional Search Spaces

Very high-dimension search spaces would require a large set of prime numbers that obey the heuristic rules for selection. These can be difficult to find, although all effort expended in finding them is performed off line and does not affect the EA performance. More important, however, is the fact that it is probably not worth the effort. In large dimensional spaces, the distances between points increase rapidly. With very large distances between only a few points, the benefit of placing population members “uniformly” versus “randomly” diminishes. For this and other reasons, several mathematical researchers have expressed doubts about the usefulness of uniform distribution methods for dimensions higher than about 12 [12] [9]. Because of this, if the dimensionality of the problem is greater than 12, random population placement is recommended. Up through dimension 12, Table 1 provides a pre-computed set of α 's.

4.2 Population Placement in Complex Search Spaces

Up until this point, we have only been describing population placement in symmetric, real-numbered search spaces. Fortunately, the transition to other search spaces is quite straightforward.

The use of this technique for asymmetric search spaces is straightforward. Since the population placement algorithm is so computationally efficient, simply compute the population as if the search space was symmetric in all axes (using the largest axis, scaled 0 to 1 as the basis), and disregard any points falling outside the search space. Compute additional points, continuing to discard those falling outside of the search space, until the desired number of population members fall within the search space.

Similarly, the use of this technique for search spaces involving non-binary combinatorial problems is also straightforward. If, for example, one of the axes of the search space involves four non-ordinal values, a simple partitioning of the axis into four equal-length segments to represent the four values will suffice for problem initialization. Since the placement algorithm assures reasonably uniform inter-dimensional dispersion for problem initialization, this technique will provide an appropriate representation of population members dispersed throughout the search space.

5 Experimental Verification

To examine the usefulness of the sentinel placement algorithm for population initialization, a simple genetic algorithm (GA) with gray code binary represen-

tation, fitness-proportional selection, uniform crossover, and a mutation rate of 0.001 was used. This GA was run against a variety of different problems of varying complexity. A few representative results are provided here. The problems presented are:

- De Jong Test Function 2, also called Rosenbrock’s Function, in 2 dimensions [13].
- De Jong Test Function 3, a step function, in 5 dimensions [13].
- Michalewicz’s Function, in 10 dimensions [13].

A population of 25 was used. The De Jong test functions #2 and #3 and the Michalewicz test function are usually implemented as minimization functions, but were implemented as maximization functions for these experiments, applying the following equations respectively:

$$f(\bar{x}) = 3907 - 100(x_1^2 - x_2)^2 + (1 - x_1)^2, \text{ for } x_{1,2} \in [-2.048, 2.048]. \quad (3)$$

$$f(\bar{x}) = 25.0 - \sum_{i=1}^N |x_i|, \text{ for } x_i \in [-2.048, 2.048]. \quad (4)$$

$$f(\bar{x}) = \sum_{i=1}^N ((\sin(x_i))(\sin(\frac{ix_i^2}{\pi}))^{20}), \text{ for } x_i \in [0, \pi]. \quad (5)$$

The GA was run on each of these problems 100 times using random population initialization and 100 times using placed population initialization (recall that, since the placement algorithm uses a random place for the first placed point, each “placed population” will be different, but equivalently dispersed in the search space). Since these are static problems, the average “best-so-far,” along with its variance, for the 100 runs are reported as the measures of effectiveness in this chapter.

The results of the first 25 generations of the above experiments are graphed in Figures 3 through 8. As can be seen in Figures 3 and 4 there appears to be a considerable reduction in variance for De Jong Test Function 2. The other charts show less dramatic results, so the results were further analyzed to determine their statistical significance.

6 Analysis

At any specific generation, the F-statistic can be used to determine whether the differences between the two variances are statistically significant.

Table 2 provides a look at the significance of the variance difference seen at generation 10 in each of the figures from the previous section. For each technique, the mean best-so-far (BSF) fitness at generation 10, the variance of the mean

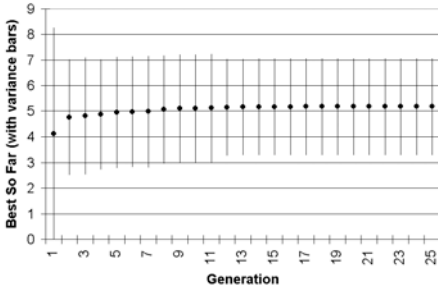


Fig. 3. Best So Far, Random Initialization, DeJong Test Function 2

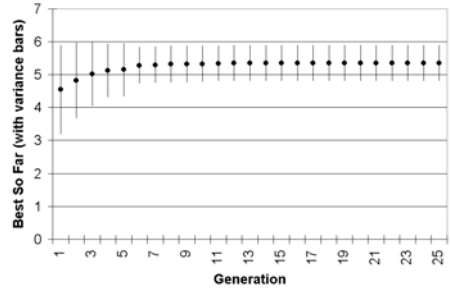


Fig. 4. Best So Far, Placed Initialization, DeJong Test Function 2

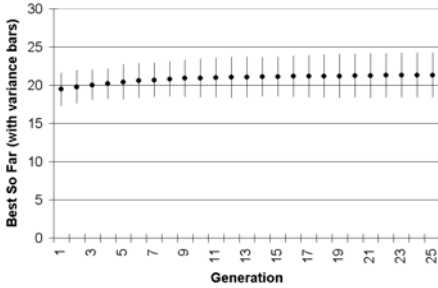


Fig. 5. Best So Far, Random Initialization, DeJong Test Function 3, 5-Dimensions

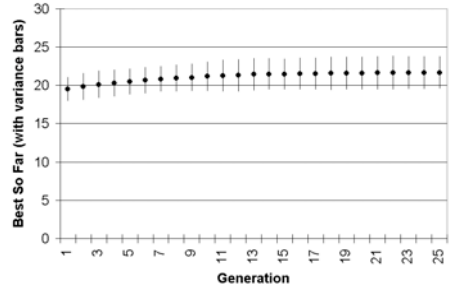


Fig. 6. Best So Far, Placed Initialization, DeJong Test Function 3, 5-Dimensions

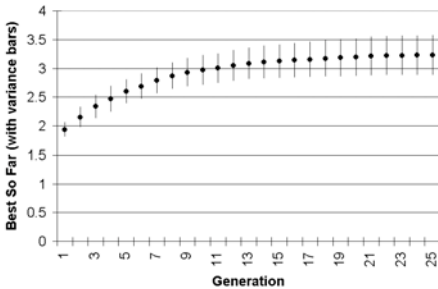


Fig. 7. Best So Far, Random Initialization, Michaeliwick's Function, 10-Dimensions

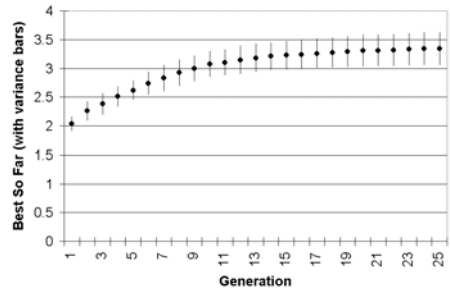


Fig. 8. Best So Far, Placed Initialization, Michaeliwick's Function, 10-Dimensions

Table 2. Results of Different Population Initialization at Generation 10

Test	BSF Mean	Variance	F	Confidence
De Jong 2 2D random	3905.10	2.12	3.867	99+%
De Jong 2 2D placed	3905.32	0.55	-	-
-	-	-	-	-
De Jong 3 5D random	20.90	2.53	1.326	76%
De Jong 3 5D placed	21.15	1.91	-	-
-	-	-	-	-
Michalewicz 10D random	3.00	0.26	1.135	77%
Michalewicz 10D placed	3.10	0.23	-	-

fitness at generation 10, the F statistic computed as the ratio of the squares of the two variances, and the confidence level that the variances are different.

As expected, there was no significant difference in the mean best-so-far performance of the EAs attributable to the population initialization algorithm (the t statistic is therefore omitted for brevity), but there was generally a reduction in the variance of the mean best-so-far performance. As addressed previously, at high dimensions the advantage of the placement algorithm is reduced.

7 Summary

In this paper we have derived a population initialization algorithm for EAs in static environments. Use of the technique does not improve mean best-so-far performance over a large number of runs. Instead, it reduces the variance of the mean best-so-far performance without loss of average performance, thereby providing researchers the opportunity to reliably examine their experimental results needing fewer EA runs for an appropriate statistical sample. This may provide an opportunity for researchers to address more complex problems without an attendant increase in required computational resources.

References

1. Kallel, L. and Schoenauer, M.: Alternative Random Initialization in Genetic Algorithms. In: Proceedings of the Seventh International Conference on Genetic Algorithms. Morgan Kaufman (1997) 268–275
2. Branke, J.: Evolutionary Optimization in Dynamic Environments. Kluwer Academic Publishers (2002)
3. Shimada, K. and Gossard, D.: Bubble Mesh: Automated Triangular Meshing of Non-manifold Geometry by Sphere Packing. In: Proceedings of the Third Symposium on Solid Modeling and Applications. IEEE (1995) 225–237
4. Rush, J.: Sphere Packing and Coding Theory. In: Handbook of Discrete and Computational Geometry. CRC Press (1997) 185–208
5. Weisstein, E.: Hypersphere. In: World of Mathematics. Wolfram Research, Inc. <http://mathworld.wolfram.com> (2001)

6. Neiderreiter, H.: Low-discrepancy and Low-dispersion Sequences. In: Journal of Number Theory 30-1 (1987) 51–70.
7. Drmota, M. and Tichy, R.: Sequences, Discrepancies and Applications, Lecture Notes in Mathematics 1651. Springer-Verlag (1997)
8. Morrison R.: Designing Evolutionary Algorithms for Dynamic Environments. Ph.D. Dissertation. George Mason University (2002)
9. Bratley, P., Fox, B. and Neiderreiter, H.: Implementation and Tests of Low-discrepancy Sequences. In: ACM Transactions on Modeling and Computer Simulation 2(3). ACM (1992) 195–213
10. Alexander, J., Beck, J. and Chen, W.: Geometric Discrepancy Theory and Uniform Distribution. In: Handbook of Discrete and Computational Geometry. CRC Press (1997) 185–208
11. American-Mathematical-Society: The most irrational number.
<http://www.ams.org/new-in-math/cover/irrational3.html>. 2001
12. Owen, A.: Monte Carlo Variance of Scrambled Net Quadrature. In: Journal of Numerical Analysis 34(5). Society for Industrial and Applied Mathematics (1997) 1884–1910
13. Liao, Y. and Sun, C.: An Educational Genetic Algorithm Learning Tool. In: IEEE Transactions on Education 44(2) 2001 CD-ROM Directory 14

A Parallel Genetic Algorithm Based on Linkage Identification

Masaharu Munetomo, Naoya Murao, and Kiyoshi Akama

Hokkaido University, North 11, West 5, Sapporo 060-0811, JAPAN.

{munetomo, naoya.m, akama}@cims.hokudai.ac.jp

Abstract. Linkage identification algorithms identify linkage groups — sets of loci tightly linked — before genetic optimizations for their recombination operators to work effectively and reliably. This paper proposes a parallel genetic algorithm (GA) based on the linkage identification algorithm and shows its effectiveness compared with other conventional parallel GAs such as master-slave and island models. This paper also discusses applicability of the parallel GAs that tries to answer “which method of the parallel GA should be employed to solve a problem?”

1 Introduction

To solve large, complex combinatorial optimization problems, parallelization of optimization algorithms is crucial. Evolutionary computation models such as genetic algorithms (GAs) are considered suitable for parallel computation because they deal with a number of strings that can be processed independently in their fitness evaluations and genetic operators such as mutations and crossovers. Therefore, the parallel GA is an active research area and a number of papers have been published in this area. Typical approaches of the parallel GAs are master-slave models, island models, massively parallel GAs, and so on. Although they succeeded in parallelizing GAs, it does not mean that they succeeded in parallelizing their optimization, in other words, they can reduce time to obtain optimal solutions properly as compared with the number of processors employed. Speedup analysis is essential to discussing efficiency of parallel GAs. Cantu-Paz et. al[1] performed theoretical and empirical analysis on speedup for master-slave models, island models, and so on. The analysis shows the conditions for the models to work effectively; for example, master-slave models work effectively when time to evaluate fitness is much greater than that for communications among processors. In this paper, we first review and discuss current technologies and the conditions that they work effectively.

Discussing efficiency of parallelizations, we need to discuss effectiveness of the GA itself in solving a problem. This is because when a GA (with some operators, selection schemes, etc.) cannot solve a problem essentially, we cannot solve the problem even though parallelized version of the GA is employed. This happens especially when we employ simple GAs to solve difficult problems that have deceptiveness and encoded strings do not have tight linkage. If linkage is

loose, building blocks are easily disrupted by simple crossovers and therefore genetic optimizations are easily trapped into deceptive attractors. On the consequence of this, we cannot expect the GA to obtain optimal solutions because it is easily converged to local optima. To solve this problem, linkage identification procedures such as the LINC (Linkage Identification by Nonlinearity Check) [7, 6] and the LIEM (Linkage Identification with Epistasis Measures) have been proposed to identify linkage groups — sets of loci tightly linked to form building blocks. The merit of this approach is not only it can avoid disruption of building blocks but also it can easily be parallelized because its calculation is highly independent. This paper discusses merit of the parallel GA based on the linkage identification compared with conventional parallel GAs.

2 Parallel GAs

A number of papers have been published on parallel GAs. Majority of them are classified into the following categories:

- *master-slave models* that parallelize fitness evaluations
- *island models* that divide a population into subpopulations
- *massively parallel GAs* that place strings on a mesh, etc.
- *hierarchical models* that combine more than two methods

A master-slave model consists of a master that performs GAs and slaves that evaluate fitness values. This model performs fitness evaluations in parallel to enhance its performance. To apply this model, communication overheads should be relatively small compared with time to evaluate fitness values. This is analyzed by Cantu-Paz[1] in his doctoral thesis. In the following analysis based on his thesis, we denote time to evaluate a fitness value as T_f , communication time between processors as T_c , the overall population size as n and the number of processors of the target parallel machine as P . We can easily calculate overall execution time T_p by the P -processor parallel machine as follows:

$$T_p = PT_c + \frac{nT_f}{P}. \quad (1)$$

Speedup factor S is obtained by calculating nT_f/T_p :

$$S = \frac{nT_f}{T_p} = \frac{nT_f}{PT_c + nT_f/P}. \quad (2)$$

This is illustrated in figure 1. This figure shows that parallelization by the master-slave model is effective when $T_f \gg T_c$.

From the above calculations, we obtain the optimal number of processors P^* that maximizes the speedup factor S as follows:

$$P^* = \sqrt{nT_f/T_c}. \quad (3)$$

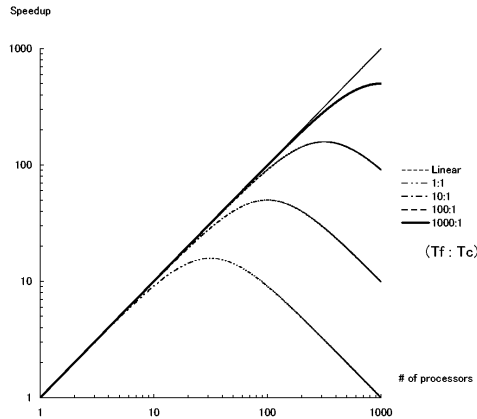


Fig. 1. Speedup by master-slave PGAs

Island models also called subpopulation-based parallel GAs divide a population into subpopulations assigned to processors. Each processor performs a GA for its subpopulation and migrates strings between subpopulations to exchange building blocks. Even though speedup realized by this model is relatively small as illustrated in figure 2, this is considered natural implementation of parallel GAs and may be employed when master-slave models does not work effectively.

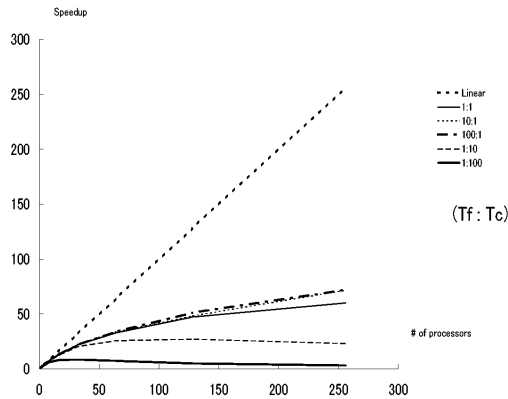


Fig. 2. An Empirical Result of Speedup by an island model PGA

The parallel GAs based on the island model expect that each subpopulation searches different candidates of building blocks to be exchanged among them, however, such favorable situations may not be realized in some problems. For example, when fitness contribution of building blocks to overall fitness is ex-

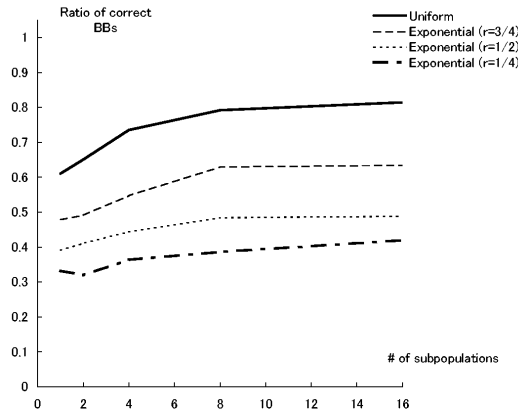


Fig. 3. Ratio of BBs correctly identified by an island model

ponentially decreasing (or increasing), all subpopulations must search the same building block with maximum fitness contribution at first, then search that with the second maximum, and so on. This is reported by Goldberg in discussing continuation operators[4].

Figure 3 shows that subpopulation-based parallel GAs performs well when fitness contribution of building blocks is uniform and their performance gets worse when the contribution becomes exponential. In the figure, we employ the sum of 5-bit trap functions as a test function and we assign a weight w_i to each i -th trap subfunction ($i = 1, 2, \dots, 20$). In uniform case, $w_i = 1$ for all i . In exponential case, $w_1 = 1.0$ and $w_i = r \times w_{i-1}$, that is, the weights are exponentially decreasing. When $r < 1/2$, the sum of all weights for $i = 2, 3, \dots$ becomes smaller than w_1 , the maximum weight value. This means that fitness contribution by the first BB exceeds those for the sum of other BBs, and therefore at first only the BB with the maximum weight should be found in all the subpopulations. This is because diversity cannot be ensured even though a number of subpopulations are employed in the island models.

3 Linkage Identification

A series of linkage identification techniques have been proposed to ensure tight linkage among loci to form building blocks. The Linkage Identification by Non-linearity Check (LINC)[7,6] identifies linkage groups by introducing bit-wise perturbations for pairs of loci to detect nonlinear interactions of fitness changes. In the LINC, we calculate the following values for each pair of loci (i, j) and for all the strings s in a randomly initialized population that has $O(c2^k)$ strings where c is a constant and k is the maximum order of linkages:

$$\begin{aligned}\Delta f_i(s) &= f(\dots \bar{s}_i \dots) - f(\dots s_i \dots) \\ \Delta f_j(s) &= f(\dots \bar{s}_j \dots) - f(\dots s_j \dots)\end{aligned}$$

$$\Delta f_{ij}(s) = f(..\bar{s}_i.\bar{s}_j..) - f(..s_i.s_j...), \quad (4)$$

where $f(s)$ is a fitness functions of a string s and $\bar{s}_i = 1 - s_i$ ($0 \rightarrow 1$ or $1 \rightarrow 0$) stands for a bitwise perturbation at the i -th locus. We can consider the following two cases:

1. If $\Delta f_{ij}(s) \neq \Delta f_i(s) + \Delta f_j(s)$, then s_i and s_j are surely members of a linkage group, so we add i to the linkage group of locus j and add j to the linkage group of locus i .
2. If $\Delta f_{ij}(s) = \Delta f_i(s) + \Delta f_j(s)$, then s_i and s_j may not be a member of a linkage group, or they are linked but linearity exists in the current context. We do nothing in this case.

The LINC checks the above conditions for all the string s in a population. When the nonlinear condition $\Delta f_{ij}(s) \neq \Delta f_i(s) + \Delta f_j(s)$ is satisfied for at least one string s , the pair (i, j) should be considered to be tightly linked. This is because a pair of loci can be optimized independently when linearity is detected for all the string. The detection of nonlinearity is the key idea of the LINC.

The Linkage Identification with Epistasis Measures (LIEM) extends the idea of the LINC by introducing an *epistasis measure* that represents *tightness of linkage* based on the nonlinearity conditions of the LINC. The epistasis measure of the LIEM is defined as follows:

$$e_{ij} = \max_{s \in P} |\Delta f_{ij}(s) - (\Delta f_i(s) + \Delta f_j(s))|, \quad (5)$$

where $\Delta f_i(s)$, $\Delta f_j(s)$, $\Delta f_{ij}(s)$ are the same as those in equations (4).

This measure shows the maximum distance from where the LINC condition is satisfied, therefore, this shows a *degree of dissatisfaction* of the condition. When the measure for a pair of loci (i, j) is equal to zero, the pair should be separable from the LINC conditions. By introducing the epistasis measure, we can relax strict condition of the LINC and can introduce a clear definition of *tightness of linkage* for pairs of loci.

Figure 4 shows the algorithm of the LIEM. First, we randomly initialize a large enough initial population. From theoretical investigations for the LINC, we need to have $O(k2^k)$ strings in a population to identify linkage groups of order k . Secondly, we calculate an epistasis measure e_{ij} for each pair of loci (i, j) by applying perturbations to the pair of loci. After the calculations, the measures are sorted by descendent order. Linkage groups are generated by picking up loci from the first to the k -th sorted measures.

Although the algorithm of the LIEM seems too simple to identify accurate linkage groups, it can identify 100% correct linkage groups for problems with linear combinations of nonlinear subfunctions such as the sum of trap functions even though their bit positions are randomly encoded, and it also achieves accurate identifications for quasi-linear or weak nonlinear combinations of nonlinear subfunctions such as weak nonlinear functions of the sum of trap functions[5]. This is because the LIEM differentes strong nonlinearity from linear or weak nonlinear interactions.


```

algorithm LIEM
N = c*2^difficulty;
P = initialize N strings;
/* Calculate epistasis measure e[i][j] */
for i = 0 to l-1
  for j = 0 to l-1
    e[i][j] = 0;
    if i != j then
      for each s in P
        s' = perturb(s, i);
        f1 = fitness(s') - fitness(s);
        s' = perturb(s, j);
        f2 = fitness(s') - fitness(s);
        s' = perturb(s', j);
        f12 = fitness(s') - fitness(s);
        ep[s] = |f12 - (f1+f2)|;
        if(ep[s] > e[i][j]) then e[i][j] = ep[s];
      endfor
    endif
  endfor
endfor
/* Generate linkage group l[i][k]
   where k = 0, 1,..., difficulty-1 */
for i = 0 to l-1
  for j = 0 to l-1
    id[j] = j;
  endfor
  sort e[i][j] with j by descendent order;
  /* select linkages */
  for k = 0 to difficulty-1
    if(e[i][k] > epsilon) l[i][k] = id[i][k];
    else break;
  endfor
endfor

```

Fig. 4. The Linkage Identification with Epistasis Measure (LIEM)

4 A Parallel GA Based on Linkage Identification

In this paper, we propose a parallel GA based on the linkage identification technique. The linkage identification algorithms such as the LINC and the LIEM are easy to parallelize, because their calculations of epistasis measures are considered highly independent. Figure 5 illustrates an execution flowchart of our method.

To parallelize linkage identifications, we assign calculations of epistasis measures to processors in a parallel computer. First, a master processor randomly initializes a population of $N = O(k2^k)$ strings and broadcasts them to all the slave processors. As a consequence of this, all the processors have the same population and they can calculate their assigned epistasis measures. Computational cost for calculating epistasis measures for all the pair of loci is $O(l^2)$ (more precisely, we need to calculate $(l^2 - l)/2$ epistasis values). In order to calculate one epistasis measure, we need to evaluate $3N$ fitness values because calculations of $\Delta f_i(s)$, $\Delta f_j(s)$, $\Delta f_{ij}(s)$ are necessary for each s in the population. After cal-

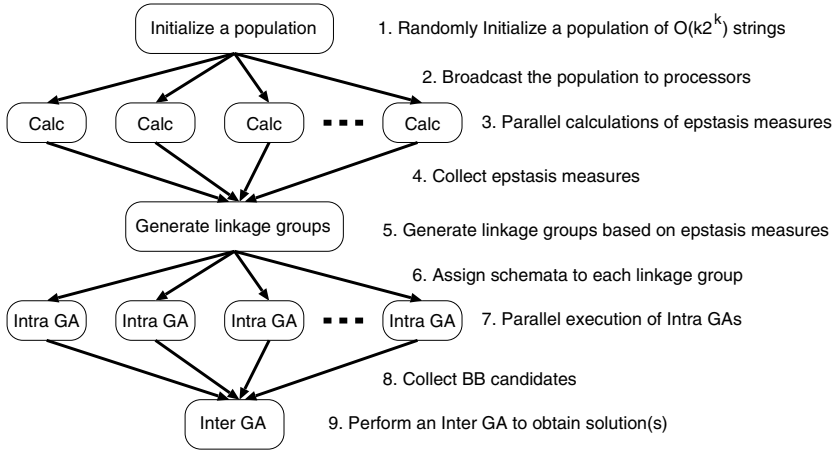


Fig. 5. An overview of PGA based on linkage identification

culations of the epistasis measures, they are collected to the master processor which generates linkage groups based on them. This is essentially the same as the serial version of the LIEM.

When we have P processors in a parallel computer, overall computation time T_p for the parallel linkage identification becomes as follows:

$$T_p = \frac{3NT_f(l^2 - l)/2}{P} + T_cN + T'_c(l^2 - l), \quad (6)$$

where T_f and T_c are the same as in equation (1) and T'_c is the time to send/receive one epistasis measure ($T'_c < T_c$). Similar to the master-slave model, we can calculate speedup factor S for the proposed algorithm as follows:

$$S = (3T_fN(l^2 - l)/2)/T_p = \frac{3T_fN(l^2 - l)/2}{\frac{3T_fN(l^2 - l)/2}{P} + T_cN + T'_c(l^2 - l)/2}. \quad (7)$$

We can expect effective parallelization with this algorithm because communication time to send an epistasis measure is much smaller than that for calculating it.

After the generation of linkage groups, we perform Intra GAs in slave processors to search candidates of building blocks and an Inter GA in the master processor to mix and test the candidates to find optimal solutions. The Intra GAs and the Inter GA were originally introduced in a report that proposes a GA based on the LINC[6]. To perform the Intra GAs, we divide initial strings into schemata based on the obtained linkage groups. Figure 6 illustrates this decomposition process.

The Intra GA applies ranking selections, uniform crossovers and simple mutations to search candidates of building blocks in the linkage group. To evaluate schemata, competitive template in messy GA[3] is employed. After the Intra

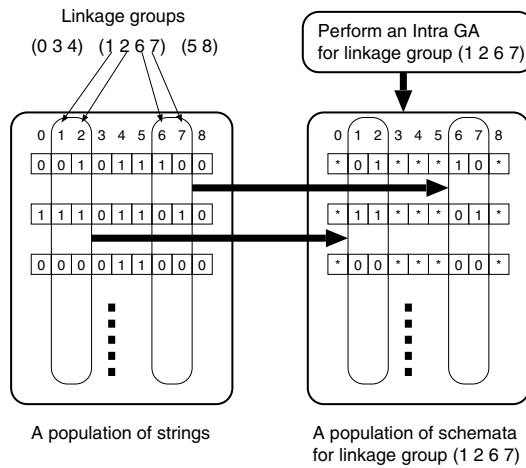


Fig. 6. Division of strings into schemata based on linkage groups

GAs, we select a limited number of well-performed schemata as building block candidates in each linkage group.

The Inter GA processes obtained building block candidates by applying crossovers based on the linkage groups and ranking selections repeatedly. Figure 7 shows crossover operator of the Inter GA. Building block candidates are mixed by this operator and their combinations are tested through selections in order to obtain optimal solutions.

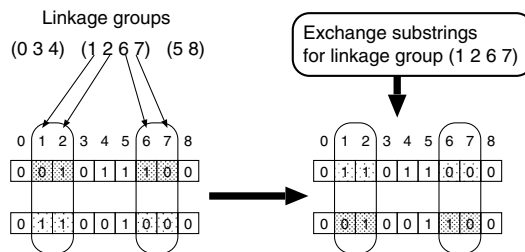


Fig. 7. Crossover operator of the Inter GA

5 Empirical Results

We perform numerical experiments for the proposed parallel GA based linkage identification. We employ the following weighted sum of trap subfunctions as a test function.

$$f(s) = \sum_{i=1}^L w_i f_i(u_i). \quad (8)$$

where $w_i > 0$ is the weight of subfunction f_i of unitation u_i (the number of 1's in the K -bit i -th substring of s) defined as follows:

$$f_i(u_i) = \begin{cases} K - u_i & \text{if } 0 \leq u_i \leq K - 1 \\ K & \text{if } u_i = K \end{cases} \quad (9)$$

In the following experiments, we consider the following two cases:

Uniform : All weights have the same value: $w_i = 1$ for all i .

Exponential : Weights are exponentially decreasing: $w_1 = 1$, $w_i = r w_{i-1}$ ($i = 2, 3, \dots, L$, $0 < r < 1$.)

Fig 8 shows the result on speedup by the parallel GA based on linkage identification. In this experiment, we employ uniform case of the test function. For exponential functions, similar result should be obtained because linkage identification algorithms do not depend on scaling of the fitness function. In this experiment, we change ratio of time for fitness evaluations (T_f) and that for communications (T_c).

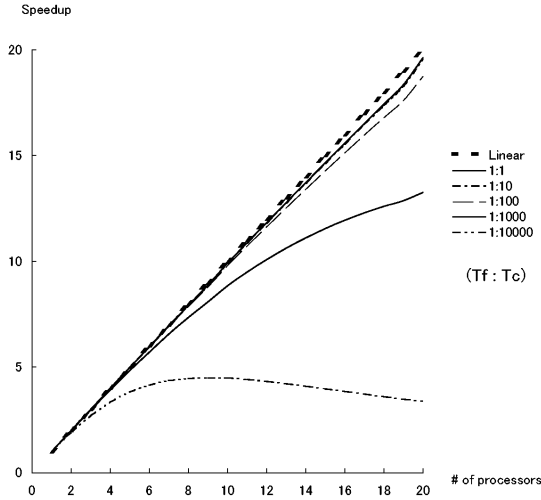


Fig. 8. Speedup by a PGA with linkage identification

This result shows that the PGA with linkage identification achieves near linear speedup except when communication overheads are extraordinary large compared with those for fitness evaluations. (Note that ratios of T_f and T_c in this figure are different from those in the figure 1.)

In figure 9, we compare overall performances of an island model PGA to that with parallel linkage identification for the exponential test function. In the figure, the x-axis shows problem size (total length of the string = LK) and the y-axis shows overall time to obtain optimal solutions ($\times T_f$). In this experiment, we assume the parallel machine has 8 processors and we optimize population size for each model and each string length.

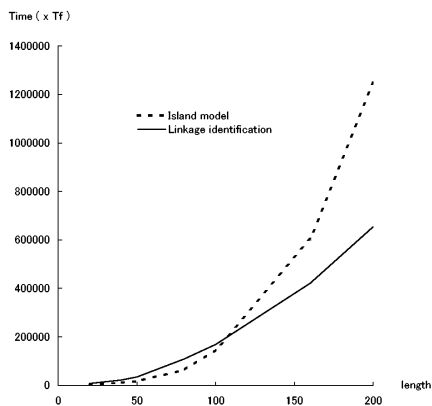


Fig. 9. Problem size vs. time (exponential function)

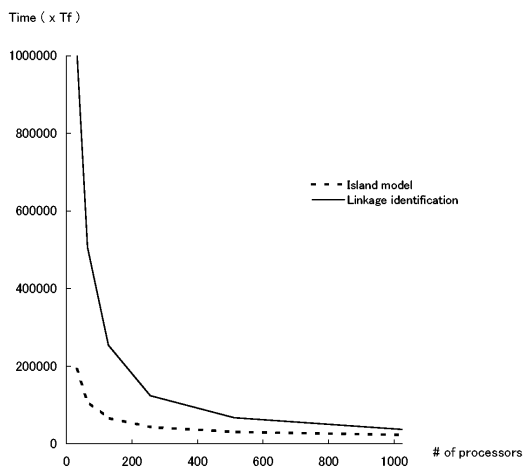


Fig. 10. # of processors vs. time (uniform function)

This figure shows that PGA with linkage identification can find optimal solutions with less computational cost compared with that based on an island model

when string length is larger than around a hundred. This is because an island model cannot search effectively a variety of building blocks in their subpopulations, on the other hand, the PGA based on linkage identification can search building block candidates separately in each linkage group. Another reason for the performance difference is that an island model needs larger population size when signal difference of subfunctions becomes small. This is easily understood from the equation of population sizing calculated by Goldberg et. al[2] as follows:

$$n = 2c\kappa \frac{\sigma_M^2}{d^2}, \quad (10)$$

where n is the necessary initial population size, c is a constant determined by sampling noise, κ is the cardinality of competing schemata, σ_M is the standard deviation of the fitness distribution, and d is the signal difference of the fitness between the best and the second best schemata.

When signal difference of fitness d in the equation becomes smaller, the population size n necessary to obtain optimal solutions becomes larger. In the exponential case, when string length becomes longer, minimum difference of the fitness decreases exponentially, therefore, an island model PGA needs much larger size of population. In contrast, the PGA with linkage identification do not depend on such fitness scaling effect and its necessary population size is constant with respect to signal differences.

Figure 10 shows time to obtain optimal solutions for an island model PGA and that with linkage identification for the uniform case. In this figure, the x-axis shows the number of processors employed and the y-axis shows the time to obtain optimal solutions. In this experiment, we employ a uniform test function whose string length is 500, and we optimize initial population size for both models.

This figure clearly shows that the island model performs better than the PGA with linkage identification in the uniform test function. This is because an island model works effectively by maintaining variety of building block candidates in each subpopulation. On the other hand, PGA with linkage identification needs relatively large computational overheads to obtain accurate linkage information.

6 Conclusions

In this paper, we propose a PGA based on linkage identification. The PGA achieves quasi-linear speedup except when communication overheads is extremely large compared with that for fitness evaluations. In summary, we can select one of the PGA models to solve optimization problems according to the following guidelines:

- If the problem is difficult to ensure tight linkage in advance with prior knowledge, linkage identifications are necessary which can be parallelized easily.
- If time to evaluate fitness values are much larger than that of communications among processors, the master-slave models work effectively.

- If the fitness function of the problem consists of uniformly weighted subfunctions, island models can maintain diversity of building block candidates and are expected to perform well.
- If the fitness function consists of non-uniform subfunctions whose weights are largely different, parallel linkage identifications are expected to perform better than island models.

Our future works includes applications of the proposed model to real world design problems such as broadband network design problems, and we also plan to compare the PGA with linkage identification with parallelized version of advanced GAs and estimation of distribution algorithms such as the Bayesian optimization algorithm.

References

1. Erick Cantú-Paz. *Designing Efficient and Accurate Parallel Genetic Algorithms*. PhD thesis, University of Illinois at Urbana-Champaign, 1999.
2. D. E. Goldberg. Sizing populations for serial and parallel genetic algorithms. pages 70–79, 1989. (Also TCGA Report 88004).
3. D. E. Goldberg, K. Deb, and B. Korb. Messy genetic algorithms revisited: Studies in mixed size and scale. *Complex Systems*, 4:415–444, 1990.
4. David E. Goldberg. Using time effectively: Genetic-evolutionary algorithms and the continuation problem. Technical Report IlliGAL Report No.99002, University of Illinois at Urbana-Champaign, 1999.
5. Masaharu Munetomo. Linkage identification based on epistasis measures to realize efficient genetic algorithms. In *Proceedings of the 2002 Congress on Evolutionary Computation*, 2002.
6. Masaharu Munetomo and David E. Goldberg. Designing a genetic algorithm using the linkage identification by nonlinearity check. Technical Report IlliGAL Report No.98014, University of Illinois at Urbana-Champaign, 1998.
7. Masaharu Munetomo and David E. Goldberg. Identifying linkage by nonlinearity check. Technical Report IlliGAL Report No.98012, University of Illinois at Urbana-Champaign, 1998.

Generalization of Dominance Relation-Based Replacement Rules for Memetic EMO Algorithms

Tadahiko Murata ¹, Shiori Kaige ², and Hisao Ishibuchi ²

¹ Department of Informatics, Kansai University
2-1-1 Ryozenji-cho, Takatsuki, Osaka 569-1095, Japan
murata@res.kutc.kansai-u.ac.jp,

<http://www.res.kutc.kansai-u.ac.jp/~murata/>

² Department of Industrial Engineering, Osaka Prefecture University
1-1 Gakuen-cho, Sakai, Osaka 599-8531, Japan
{hisaoi, shior}@ie.osakafu-u.ac.jp,
http://www.ie.osakafu-u.ac.jp/~hisaoi/ci_lab_e/

Abstract. In this paper, we generalize the replacement rules based on the dominance relation in multiobjective optimization. Ordinary two replacement rules based on the dominance relation are usually employed in a local search (LS) for multiobjective optimization. One is to replace a current solution with a solution which dominates it. The other is to replace the solution with a solution which is not dominated by it. The movable area in the LS with the first rule is very small when the number of objectives is large. On the other hand, it is too huge to move efficiently with the latter. We generalize these extreme rules by counting the number of improved objectives in a candidate solution for LS. We propose a LS with the generalized replacement rule for existing EMO algorithms. Its effectiveness is shown on knapsack problems with two, three, and four objectives.

1 Introduction

Since Schaffer's study [1], evolutionary algorithms have been applied to various multiobjective optimization problems for finding their Pareto-optimal solutions. Recently evolutionary algorithms for multiobjective optimization are often referred to as EMO (evolutionary multiobjective optimization) algorithms. The task of EMO algorithms is to find Pareto-optimal solutions as many as possible. In recent studies (e.g., [2-6]), emphasis was placed on the convergence speed to the Pareto-front as well as the diversity of solutions. In those studies, some form of elitism was used as an important ingredient of EMO algorithms. It was shown that use of elitism improved the convergence speed to the Pareto-front [5].

One promising approach for improving the convergence speed to the Pareto-front is the use of local search in EMO algorithms. Hybridization of evolutionary algorithms with local search has already been investigated for single-objective optimization problems in many studies (e.g., [7], [8]). Such a hybrid algorithm is often referred to as a memetic algorithm. See Moscato [9] for an introduction to this field and [10]–[12] for recent developments. The hybridization with local search for multiobjective

optimization was first implemented in [13], [14] as a multiobjective genetic local search (MOGLS) algorithm where *a scalar fitness function with random weights* was used for the selection of parents and the local search for their offspring. Jaszekiewicz [15] improved the performance of the MOGLS by modifying its selection mechanism of parents. While his MOGLS still used the scalar fitness function with random weights in selection and local search, it did not use the roulette wheel selection over the entire population. A pair of parents was randomly selected from a pre-specified number of the best solutions with respect to the scalar fitness function with the current weights. This selection scheme can be viewed as a kind of mating restriction in EMO algorithms. Knowles & Corne [16] combined their Pareto archived evolution strategy (PAES [2], [4]) with a crossover operation for designing a memetic PAES (M-PAES) [17]. In their M-PAES, *the Pareto-dominance relation* and *the grid-type partition of the objective space* were used for determining the acceptance (or rejection) of new solutions generated in genetic search and local search. The M-PAES had a special form of elitism inherent in the PAES. In those studies, the M-PAES was compared with the PAES, the MOGLS of Jaszekiewicz [15], and an EMO algorithm. In the above-mentioned hybrid EMO algorithms (i.e., multiobjective memetic algorithms [13]–[17]), local search was applied to individuals in *every generation*. In some studies [18], [19], local search is restrictedly applied to every generation by limited local search *only to non-dominated solutions* [18] or introducing *the tournament selection* and *the selection probability* of candidate solutions for local search [19]. Another way of application of local search is proposed in [20], [21], where local search was applied to individuals only in the final generation.

In order to design a local search for multiobjective optimization, a rule for replacing a current solution with another solution should be defined in advance. Murata *et al.* [22] showed experimental results on pattern classification problems where a scalar fitness function-based replacement rule was better than the dominance relation-based replacement rules. In the dominance relation-based replacement rules, the current solution is replaced with a solution which dominates the current one or a solution which is at least a non-dominated solution with the current one. Ishibuchi *et al.* [23] also pointed this matter by experimental results on scheduling problems. As mentioned in [17], [22] and [23], the replacement rule to accept a non-dominated solution has a weak search pressure since almost all pairs of solutions (a current solution and a candidate solution) will be non-dominated with respect to each other especially in problems with a large number of objectives. On the other hand, the replacement rule to accept a dominating solution does not work well because it is difficult to find a dominating solution for the current solution.

We generalize the replacement rules based on the dominance relation by counting the number of better objective values. Details of the dominance relation-based replacement rules are shown in the next section. We employ a local search using the proposed replacement rule to improve the performance of existing EMO algorithms such as SPEA [3] and NSGA-II [6]. We apply them with the local search to multi-objective knapsack problems as benchmark problems [3], and show the effectiveness of our generalization.

2 Dominance Relation-Based Replacement Rules

2.1 Previous Extensions of Dominance Relation

In order to improve the performance of local search using the dominance relation, several ideas to extend it have been already proposed in [17], [24], and [25]. Knowles and Corne [17] proposed a replacement rule by which a current solution is replaced with a non-dominated solution if it dominates other solutions in the set of non-dominated solutions obtained so far. Ikeda et al. [24] proposed their “ α -dominance” where a small detriment in one or several of the objectives is permitted if an attractive improvement in the other objective(s) is achieved. While these two methods try to extend the area of dominating solutions of the current solution, Laumanns et al. [25] proposed their “ ε -dominance” where a solution with a small improvement in every normalized objective does not dominate the current one.

Each of these three methods can be considered as an extension of the dominance relation. Before explaining these extensions, we show the dominance relation defined in multiobjective optimization. Without loss of generality, we assume the following N -objective maximization problem:

$$\text{Maximize } \mathbf{z} = (f_1(\mathbf{x}), f_2(\mathbf{x}), \dots, f_N(\mathbf{x})), \quad (1)$$

$$\text{subject to } \mathbf{x} \in \mathbf{X}, \quad (2)$$

where $\mathbf{z}$ is the objective vector with N objectives to be maximized, $\mathbf{x}$ is the decision vector, and $\mathbf{X}$ is the feasible region in the decision space. A solution $\mathbf{x} \in \mathbf{X}$ is said to dominate another solution $\mathbf{y} \in \mathbf{X}$ if the following two conditions are satisfied.

$$f_i(\mathbf{x}) \geq f_i(\mathbf{y}), \quad \forall i \in \{1, 2, \dots, N\}, \quad (3)$$

$$f_i(\mathbf{x}) > f_i(\mathbf{y}), \quad \exists i \in \{1, 2, \dots, N\}. \quad (4)$$

If there is no solution which dominates $\mathbf{x}$ in $\mathbf{X}$, $\mathbf{x}$ can be said to be a Pareto-optimal solution. Fig. 1 shows that there are four areas of candidate solutions for the solution $\mathbf{x}$ in the case of two-objective problems. When we employ this dominance relation in local search, two replacement rules can be used in the local search as follows:

Rule A: Move to dominating solutions:

Replace the solution $\mathbf{x}$ with a solution which dominates it (Area A in Fig. 1).

Rule B: Move to non-dominated solutions:

Replace the solution $\mathbf{x}$ with a solution which is not dominated by $\mathbf{x}$ (Areas A - C).

The movable area in the local search with Rule A is very small when the number of objectives is large. On the other hand, it is too huge to move efficiently with Rule B. Therefore some extensions for the dominance relation should be considered.

As shown in Fig. 2, Knowles and Corne [17] extended the area of dominating solutions using non-dominated solutions obtained so far. Fig. 3 shows the dominating area of the current one defined by the α -dominance relation [24]. While these two methods enlarge the dominating area of the current solution, the ε -dominance relation [25] reduces the dominating area as shown in Fig. 4. Since the aim of the ε -dominance is reducing the number of non-dominated solutions obtained by this dominance relation, an opposite strategy is used. However, we can see that the area of

non-dominated solutions (B and C) is also reduced by the hatched area which consists of dominated solutions (D in Fig. 4). Therefore we can see that this is also a method to reduce the area of non-dominated solutions.

As we observe from Figs. 2 - 4, the area of non-dominated solutions is reduced by these three methods. However, each of them needs more computational efforts. The method in [17] needs to compare the candidate solution with non-dominated solutions. Its performance may depend on the quality of the obtained set of non-dominated solutions. As for the α -dominance [24], the decision maker (DM) should define parameters β and γ for every pair of objectives in advance. The DM should also define a parameter ε in advance to use the ε -dominance [25], and ε should be determined carefully since a large ε makes a solution obtained by this method far from the true Pareto-front.

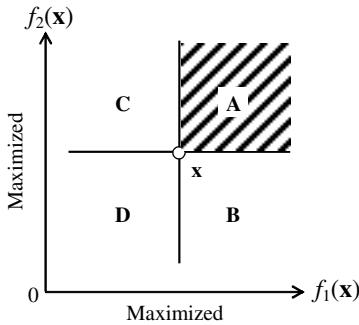


Fig. 1. The area of candidate solutions which replace the current solution x by the dominance relation.

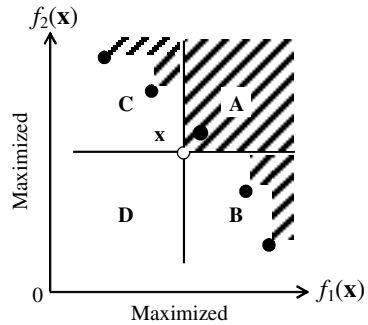


Fig. 2. The area of candidate solutions which replace the current solution x by the method in [17].

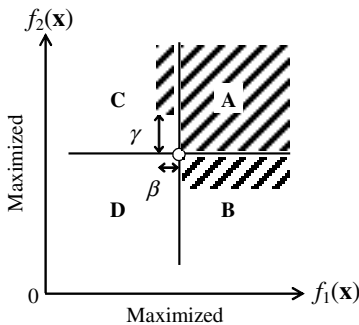


Fig. 3. The area of candidate solutions which replace the current solution x by the α -dominance [24].

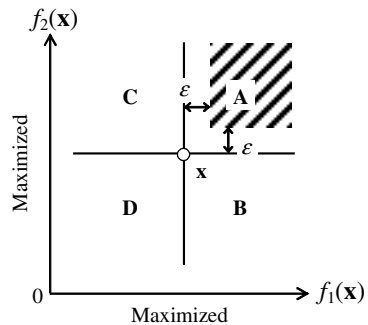


Fig. 4. The area of candidate solutions which replace the current solution x by the ε -dominance [25].

2.2 Generalization of the Dominance Relation

In this section, we generalize the two replacement rules (i.e., Rules A and B) shown in the previous section by counting the number of improved objectives. Fig. 5 shows that there are eight possible spaces for the solution $\mathbf{x}$ in the case of three-objective problems. Every solution in Space A dominates the solution $\mathbf{x}$. On the other hand, the solution $\mathbf{x}$ dominates solutions in Space H. Therefore Rule A allows the current solution to move to a solution in only one space, Space A. On the other hand, Rule B enables it to move to neighborhood solutions in all spaces except a dominated space H in Fig. 5. This means that $(2^N - 1)$ spaces are allowed out of 2^N spaces in Rule B. We can see that the number of accepted spaces is extreme in each of both cases. That is, while the number of accepted spaces is only one as for Rule A, it is $(2^N - 1)$ for Rule B. We generalize these two extreme cases by counting the number of improved objectives.

The number of improved objectives for the solution $\mathbf{x}$ is different in each space. For example, Fig. 5 shows that the number of improved objectives for a solution in Space A is three. It is zero for a solution in Space H. There are other spaces where the number of improved objectives is one or two. That is, Spaces B, C, and E have two improved objectives, and Spaces D, F, and G have one.

In the case of N -objective problems, the number of possible spaces from the current solution is 2^N . The number of improved objectives varies from zero to N in this case. We generalize the replacement rules A and B by considering the number of improved objectives d . That is, the current solution $\mathbf{x}$ is replaced with a solution which has d or more improved objective values. We have the following generalized rule:

Rule d : Move to d -Improved Solutions:

Replace the current solution $\mathbf{x}$ with a solution which has d or more improved objectives.

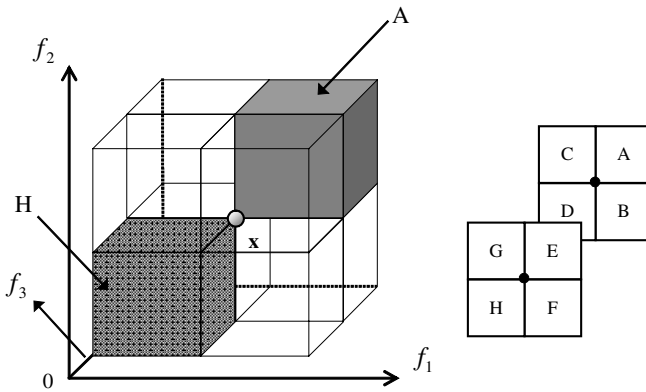


Fig. 5. Eight spaces for the current solution in the case of three-objective problems.

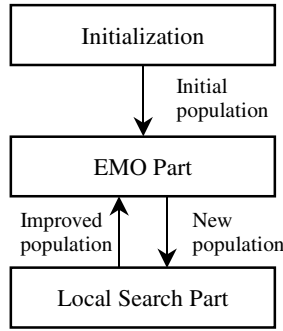


Fig. 6. Generic form of our local search part and EMO part.

By varying the value of d , we have the following rules where N is the number of objectives:

Rule N : Accept a solution which has N better objective values.

Rule $N - 1$: Accept a solution which has $N - 1$ or more better objective values.

⋮

Rule 2: Accept a solution which has at least two better objective values.

Rule 1: Accept a solution which has at least one better objective value.

Therefore, Rules A and B in Subsection 2.1 are Rule N and Rule 1 of the proposed rule, respectively.

3 Local Search Using the Generalized Replacement Rule

The outline of our local search can be written in a generic form as Fig. 6. This figure shows a basic structure of simple memetic algorithms. As shown in Fig. 6, our local search part can be applied to every EMO algorithm. For other types of memetic algorithms, see Krasnogor [26] where taxonomy of memetic algorithms was given using an index number D . This type of memetic algorithms is a $D = 4$ memetic algorithm in his taxonomy (for details, see [26]).

We design our local search as follows:

[Proposed Local Search]

Iterate the following seven steps N_{pop} times, where N_{pop} is the number of populations to be governed by genetic operations such as crossover and mutation in an EMO algorithm. Then replace the current population with N_{pop} solutions obtained by the following steps.

Step 1: Randomly choose two individuals from the current population.

Step 2: Count the number of better objective values between the two solutions. Select a solution which has a larger number of better objective values.

- Step 3: Select another solution from the current population, and back to Step 2 until t solutions are compared from the current population.
- Step 4: Apply local search with the local search probability p_{LS} . If it is applied, go to Step 5. If not, go to Step 7.
- Step 5: Generate a neighborhood solution of the current solution, and calculate the objectives of the generated solution. Count the number of improved objectives by the generated solution.
- Step 6: If the number of improved objective values is d or more, replace the current solution with the generated solution, and back to Step 5 for examining the neighborhood solution for the generated solution. If not, back to Step 5 until the number of examined solution for the current solution becomes k . If there is no better solution among k neighborhood solutions, go to Step 7.
- Step 7: Back to Step 1 until N_{pop} solutions are selected for the local search.

When local search is applied to the selected solution in Step 4, the final solution obtained by the local search is included in the next population. If local search is not applied, the selected solution is included in the next population. Therefore Steps 1 - 3 can be considered as the tournament selection for selecting candidate solutions for local search. In this tournament selection, we also employ the idea of the generalized replacement rule. That is, we select a solution with respect to the number of better objective values among t solutions. We use the local search probability p_{LS} for decreasing the number of solutions to which local search is applied. In this way, local search is not applied to all the selected solutions. If local search is applied to all the solution among the population, the computation time may be wasted to improve dominated solutions. Moreover, we also employ the number of examined solutions k in Step 6 in order to control the balance between genetic search and local search.

Since this proposed local search can be applied to any EMO algorithm, we apply our local search to SPEA [3] and NSGA-II [6]. In order to show its effectiveness, we employ multiobjective knapsack problems [3]. We show results of computer simulations in the next section.

4 Computer Simulations on Multiobjective Knapsack Problems

4.1 Multiobjective Knapsack Problems

We employ multiobjective knapsack problems [3] to which we applied EMO algorithms with the proposed local search. Those test problems are available from the web site (<http://www.tik.ee.ethz.ch/~zitzler/>). Generally, a 0/1 knapsack problem consists of a set of items, weight and profit associated with each item, and an upper bound for the capacity of the knapsack. The task is to find a subset of items which maximizes the total profits within the prespecified total weight of the items. This single-objective problem was extended to the multiobjective case by allowing an arbitrary number of knapsacks in [3]. In the multiobjective knapsack problem, there are m items and N knapsacks. Profits of items, weights of items, and capacities of knapsacks are denoted as follows:

p_{ij} : profit of item j according to knapsack i , (5)

w_{ij} : weight of item j according to knapsack i , (6)

c_i : capacity of knapsack i , (7)

where $j = 1, \dots, m$ and $i = 1, \dots, N$. (8)

The decision vector $\mathbf{x}$ of this problem is $\mathbf{x} = (x_1, x_2, \dots, x_m)$, where x_j is a binary decision value. $x_j = 0$ and $x_j = 1$ mean that the item j is not included in the knapsacks, and it is included in the knapsacks, respectively. Thus N -objective problem can be written as follows:

$$\text{Maximize } \mathbf{z} = (f_1(\mathbf{x}), f_2(\mathbf{x}), \dots, f_N(\mathbf{x})), \quad (9)$$

$$\text{subject to } \mathbf{x} \in \{0,1\}^m, \text{ and } \sum_{j=1}^m w_{ij} \cdot x_j \leq c_i \text{ for } i = 1, \dots, N, \quad (10)$$

where each objective function is described in the following form:

$$f_i(\mathbf{x}) = \sum_{j=1}^m p_{ij} \cdot x_j \text{ for } i = 1, \dots, N. \quad (11)$$

On the website of the first author of [3], problems with 100, 250, 500, and 750 items for 2, 3 and 4 knapsacks are available. We employed the SPEA [3] and the NSGA-II [6] as representative EMO algorithms. These algorithms are known as high performance algorithms for multiobjective problems. We preliminarily defined the parameters in both the algorithms as shown in Table 1. We employed a one point crossover with the crossover rate 0.8, a bit-flip mutation with the mutation rate 0.01 to each bit. In our local search, we specified the tournament size as $t = 6$, the selection probability as $p_{LS} = 0.1$, and the number of examined solution as $k = 3$. We commonly used these parameters to the SPEA and the NSGA-II. We generated 30 sets of different initial solutions and applied each of EMO algorithms to them.

After that we apply the three- and four-objective knapsack problems with 250, 500, and 750 items to show the effectiveness of generalizing the replacement rule based on the dominance relation.

4.2 Experimental Results on a Two-Objective Problem

We show the effectiveness of our local search on two-objective knapsack problems. We employed a 750-item problem for two knapsacks to show non-dominated solutions obtained by EMO algorithms with/without our local search. Therefore we applied each of four EMO algorithms to the problem. In this case, we can not show the effectiveness of generalization using the number of improved objective values d because the value of d can vary one or two for two-objective problems. We can see, however, the performance of the basic architecture of our local search with $d = 2$.

First we obtained 30 sets of non-dominated solutions by each algorithm. In order to depict figures clearly, we employed the 50%-attainment surface [27]. An attainment surface is a kind of trade-off surface obtained by a single run of an EMO algorithm. And 50%-attainment surface shows the estimated attainment surface obtained by at least 50% of multiple runs. Figs. 7 and 8 show the 50%-attainment surface of 30 sets of non-dominated solutions obtained by four EMO algorithms. Each axis of the figures shows the total profit of each knapsack. In a two objective problem, the total profit of each of two knapsacks is maximized.

From Figs. 7 and 8, we can see that the 50%-attainment surface is improved by introducing the local search in the area of compromised solutions. Considering only a single objective, each of the original EMO algorithm (i.e., without the local search) found better solutions. Since we employed the local search with $d = 2$, the local search allows a solution to move to only a dominating solution in two objectives. Therefore the attainment surface in the area of compromised solutions is improved by the local search.

As for the local search using the generalized replacement rule with $d = 1$, we could not obtain better attainment surfaces than the original EMO algorithms. As explained in [23], a drawback of the replacement rule which allows a solution to move to non-dominated solutions is that the current solution can be degraded by multiple moves. That may be a reason why the local search with $d = 1$ was not effective to this problem.

Table 1. Parameter settings for EMO algorithms.

Problem (objectives, items)	# of evaluations	Population size	Secondary Population Size in SPEA
(2, 750)	125 000	250	100
(3, 250)	100 000	200	80
(3, 500)	125 000	250	100
(3, 750)	150 000	300	120
(4, 250)	125 000	250	100
(4, 500)	150 000	300	120
(4, 750)	175 000	350	140

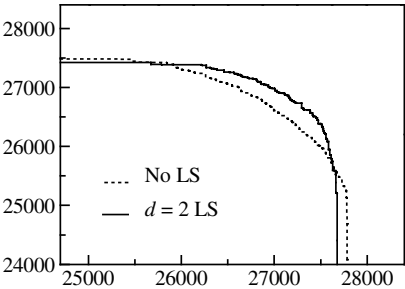


Fig. 7. 50%-attainment surface obtained by SPEA with/without local search.

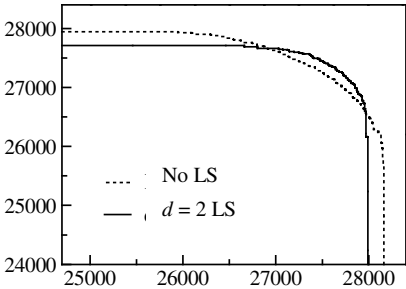


Fig. 8. 50%-attainment surface obtained by NSGA-II with/without local search.

4.3 Experimental Results on Three- and Four-Objective Problems

In the case of two-objective problems, we can depict the set distribution by two-dimensional graphs, it is difficult, however, to show the set distribution by figures for more than three-objective problems. In this paper, we use the coverage metric [3] to evaluate two sets of non-dominated solutions obtained by two EMO algorithms. Let $X', X'' \in X$ be two sets of non-dominated solutions. The coverage metric can be defined as follows:

$$C(\mathbf{X}', \mathbf{X}'') = |\{\mathbf{a}'' \in \mathbf{X}'' \mid \exists \mathbf{a}' \in \mathbf{X}' : \mathbf{a}' \succeq \mathbf{a}''\}| / |\mathbf{X}''|. \quad (12)$$

The value $C(\mathbf{X}', \mathbf{X}'') = 1$ means that all points in $\mathbf{X}''$ are dominated by or equal to points in $\mathbf{X}'$. On the other hand, $C(\mathbf{X}', \mathbf{X}'') = 0$ represents that no solutions in $\mathbf{X}''$ are covered by the set $\mathbf{X}'$. It is noted that both $C(\mathbf{X}', \mathbf{X}'')$ and $C(\mathbf{X}'', \mathbf{X}')$ have to be considered, since $C(\mathbf{X}', \mathbf{X}'')$ is not necessarily equal to $C(\mathbf{X}'', \mathbf{X}')$.

Table 2. SPEA for 3-objective problems (250, 500, and 750 items).

3 objectives	No LS	$d = 1$	$d = 2$	$d = 3$
No LS	--	0.0311	0.0088	0.0179
$d = 1$	0.7444	--	0.1960	0.2123
$d = 2$	0.8437	0.5142	--	0.3671
$d = 3$	0.8044	0.4484	0.2796	--

Table 3. NSGA-II for 3-objective problems (250, 500, and 750 items).

3 objectives	No LS	$d = 1$	$d = 2$	$d = 3$
No LS	--	0.0961	0.0862	0.1147
$d = 1$	0.4373	--	0.2274	0.2369
$d = 2$	0.4797	0.3283	--	0.3127
$d = 3$	0.4466	0.3326	0.2473	--

Table 4. SPEA for 4-objective problems (250, 500, and 750 items).

4 objectives	No LS	$d = 1$	$d = 2$	$d = 3$	$d = 4$
No LS	--	0.0038	0.0010	0.0001	0.0012
$d = 1$	0.8820	--	0.1488	0.1369	0.1697
$d = 2$	0.9353	0.4231	--	0.2394	0.2904
$d = 3$	0.9299	0.4356	0.2792	--	0.2976
$d = 4$	0.9161	0.3756	0.2122	0.2011	--

We applied the EMO algorithms with/without our local search to three three-objective problems and three four-objective problems using the parameters in Table 1. We varied the number of improved objective values d as $d = 1, 2, 3$ in three-objective problems, and $d = 1, 2, 3, 4$ in four-objective problems. Therefore we had four variants for each EMO algorithm in the case of three-objective problems, and five in the case of four-objective problems. We compare two sets of non-dominated solutions using the coverage metric, and calculate an average value over 30 trials. Tables 2–5 show the summarization of the results for each EMO algorithm. We average the values of the coverage over three different items. The second column of Tables 2 and 3 shows that the sets obtained original algorithm is covered by those obtained by algorithms with the proposed local search. For example, 0.8437 in the cell of the 4th row and the second column in Table 2 shows that 84.37 % solutions obtained by the original EMO algorithm (SPEA with No LS) are covered by solutions obtained by SPEA with the local search ($d = 2$). From Tables 2–5, we can see that the better sets obtained by our local search with $d = 2$ for three-objective problems, and $d = 2, 3$ for four-objective problems.

Table 5. NSGA-II for 4-objective problems (250, 500, and 750 items).

4 objectives	No LS	$d = 1$	$d = 2$	$d = 3$	$d = 4$
No LS	--	0.0431	0.0114	0.0114	0.0155
$d = 1$	0.4701	--	0.1165	0.1233	0.1112
$d = 2$	0.5898	0.3736	--	0.2418	0.2475
$d = 3$	0.5719	0.3551	0.2214	--	0.2122
$d = 4$	0.6022	0.3643	0.2447	0.2325	--

Due to the page limitation, we don't show detail results on each problem. Further information is shown in the first authors web (<http://www.res.kut.ac.jp/~murata/research.html>). Through the experiments, we found that the proposed local search was effective in the case of larger problems. That is, it was more effective in 500-item problem than 250, and 750 than 500 for each of three- and four-objective problems.

5 Conclusion and Future Works

In this paper, we generalized the replacement rules based on the dominance relation for local search in multiobjective optimization. Simulation results on knapsack problems with three- and four-objectives showed the effectiveness of the generalized replacement rule by introducing the number of improved objectives. As shown in the experimental results for the two-objective problems, the proposed local search is weak to improve each objective value. We can improve such weakness of the local search with the generalization rule. Since we employed the proposed only to knapsack problems in this paper, we can also apply other types of problems such as permutation problems, and function approximation problems.

References

- [1] J. D. Schaffer, "Multiple objective optimization with vector evaluated genetic algorithms," *Proc. of 1st Int'l Conf. on Genetic Algorithms and Their Applications*, pp. 93–100, 1985.
- [2] J. D. Knowles and D. W. Corne, "The Pareto archived evolution strategy: A new baseline algorithm for Pareto multiobjective optimization," *Proc. of 1999 Congress on Evolutionary Computation*, pp. 98–105, 1999.
- [3] E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach," *IEEE Trans. on Evolutionary Computation*, **3** – Algorithms: Empirical Results," *Evolutionary Computation*, **8** (2), pp. 173–195, 2000.
- [6] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Trans. on Evolutionary Computation*, **6** (2), pp. 182–197, 2002.
- [7] P. Merz and B. Freisleben, "Genetic local search for the TSP: New results," *Proc. of 4th IEEE Int'l Conf. on Evolutionary Computation*, pp. 159–164, 1997.
- [8] N. Krasnogor and J. Smith, "A memetic algorithm with self-adaptive local search: TSP as a case study," *Proc. of 2000 Genetic and Evolutionary Computation Conf.*, pp. 987–994, 2000.
- [9] P. Moscato, "Memetic algorithms: A short introduction," in D. Corne, F. Glover, and M. Dorigo (eds.), *New Ideas in Optimization*, McGraw-Hill, pp. 219–234, Maidenhead, 1999.

- [10] W. E. Hart, N. Krasnogor, and J. Smith (eds.), *First Workshop on Memetic Algorithms (WOMA I)*, in *Proc. of 2000 Genetic and Evolutionary Computation Conf. Workshop Program*, pp. 95–130, 2000.
- [11] W. E. Hart, N. Krasnogor, and J. Smith (eds.), *Second Workshop on Memetic Algorithms (WOMA II)*, in *Proc. of 2001 Genetic and Evolutionary Computation Conf. Workshop Program*, pp. 137–179, 2001.
- [12] W. E. Hart, N. Krasnogor, and J. Smith (eds.), *Proc. of Third Workshop on Memetic Algorithms (WOMA III)*, 2002.
- [13] H. Ishibuchi and T. Murata, “Multi-objective genetic local search algorithm,” *Proc. of 3rd IEEE Int’l Conf. on Evolutionary Computation*, pp. 119–124, 1996.
- [14] H. Ishibuchi and T. Murata, “A multi-objective genetic local search algorithm and its application to flowshop scheduling,” *IEEE Trans. on Systems, Man, and Cybernetics - Part C: Applications and Reviews*, **28** (3), pp. 392–403, 1998.
- [15] A. Jaskiewicz, “Genetic local search for multi-objective combinatorial optimization,” *European Journal of Operational Research*, **137** (1), pp. 50–71, 2002.
- [16] J. D. Knowles and D. W. Corne, “M-PAES: A memetic algorithm for multiobjective optimization,” *Proc. of 2000 Congress on Evolutionary Computation*, pp. 325–332, 2000.
- [17] J. D. Knowles and D. W. Corne, “A comparison of diverse approaches to memetic multiobjective combinatorial optimization,” *Proc. of 2000 Genetic and Evolutionary Computation Conf. Workshop Program*, pp. 103–108, 2000.
- [18] T. Murata, H. Nozawa, Y. Tsujimura, M. Gen, and H. Ishibuchi, “Effect of local search on the performance of cellular multi-objective genetic algorithms for designing fuzzy rule-based classification systems,” *Proc. of 2002 Congress on Evolutionary Computation*, pp. 663–668, 2002.
- [19] H. Ishibuchi, T. Yoshida, and T. Murata, “Selection of initial solutions for local search in multi-objective genetic local search,” *Proc. of 2002 Congress on Evolutionary Computation*, pp. 950–955, 2002.
- [20] K. Deb and T. Goel, “A hybrid multi-objective evolutionary approach to engineering shape design,” *Proc. of 1st Int’l Conf. on Evolutionary Multi-Criterion Optimization*, pp. 385–399, 2001.
- [21] E. Talbi, M. Rahoual, M. H. Mabed, and C. Dhaenens, “A hybrid evolutionary approach for multicriteria optimization problems: Application to the flow shop,” *Proc. of 1st Int’l Conf. on Evolutionary Multi-Criterion Optimization*, pp. 416–428, 2001.
- [22] T. Murata, H. Nozawa, H. Ishibuchi, and M. Gen, “Modification of local search directions for non-dominated solutions in cellular multiobjective genetic algorithms for pattern classification problems,” *Proc. of 2nd Int’l Conf. on Evolutionary Multi-Criterion Optimization*, 2003 (to appear).
- [23] H. Ishibuchi, T. Yoshida, and T. Murata, “Balance between genetic search and local search in memetic algorithms for multiobjective permutation flowshop scheduling,” *IEEE Trans. on Evolutionary Computation* (to appear).
- [24] K. Ikeda, H. Kita, and S. Kobayashi, “Failure of Pareto-based MOEAs: Does non-dominated really mean near to optimal?” *Proc. of 2001 Congress on Evolutionary Computation*, pp. 957–962, 2001.
- [25] M. Laumanns, L. Thiele, K. Deb, and E. Zitzler, “Combining convergence and diversity in evolutionary multiobjective optimization,” *Evolutionary Computation*, **10** (3), pp. 263–282, 2002.
- [26] N. Krasnogor, “Studies on the theory and design space of memetic algorithms,” *Ph. D. Thesis*, University of the West of England, Bristol, June 2002.
- [27] C. M. Fonseca and P. J. Fleming, “On the performance assessment and comparison of stochastic multiobjective optimizers,” *Proc. of Parallel Problem Solving from Nature IV*, pp. 584–593, 1996.

Author Index

- Abbass, Hussein A. 483, 1612
Acan, Adnan 695
Ackley, David H. 150
Aguilar Contreras, Andrés 874
Aguilar-Ruiz, Jesús S. 979
Ahn, Byung-Ha 1610
Akama, Kiyoshi 1222, 1616
Alba, Enrique 955
Almeida, Jonas 1776
Ando, Shin 1926
Angelov, P.P. 1938
Aporntewan, Chatchawit 1566
Araujo, Lourdes 1951
Armstrong, Marc P. 1530
Arnold, Dirk V. 525
Arslan, Tughrul 1614
Asai, Kiyoshi 2288
Auger, Anne 512
Aune, Thomas 2277
Aupetit, S. 140
Azad, R. Muhammad Atif 1626
- Bacardit, Jaume 1818
Balan, Gabriel Catalin 1729
Balan, M. Sakthi 425
Ballester, Pedro J. 706
Banerjee, Nilanjan 2179
Banzhaf, Wolfgang 390
Baraglia, R. 2109
Barbosa, Helio J.C. 718
Baresel, André 2428, 2442
Barfoot, Timothy D. 377
Barkaoui, Mohamed 646
Barry, Alwyn 1832
Barz, Christiane 754
Bates Congdon, Clare 2034
Baugh Jr., John W. 730
Behrens, Ivesa 754
Beielstein, Thomas 1963
Belle, Terry Van 150
Benekohal, Rahim F. 2420
Benson, Karl 1690
Berger, Jean 646
Beyer, Hans-Georg 525
Bhanu, Bir 332, 2227
- Blackwell, T.M. 1
Blain, Derrel 413
Blindauer, Emmanuel 369
Bonacina, Claudio 1382
Boryczka, Mariusz 142
Botello Rionda, Salvador 573
Bottaci, Leonardo 2455
Bouvry, Pascal 1369
Bouzerdoun, Abdesselam 742
Bowersox, Rodney 2157
Branke, Jürgen 537, 754, 766, 1568
Brown, Martin 778
Buason, Gunnar 458
Bucci, Anthony 250
Buckles, Bill P. 1624
Bull, Larry 1924
Burke, Edmund 1800
Buswell, R.A. 1938
Butz, Martin V. 1844, 1857, 1906
- Camens, Doug 1590
Cantú-Paz, Erick 790, 801
Cardona, Cesar 219
Carter, Jonathan N. 706
Carvalho, Andre de 634
Castillo, Flor 1975
Cattron, James 298
Chafekar, Deepti 813
Cheang, Sin Man 1802, 1918
Chen, Hui 379
Chen, Ping 1986
Chen, Ying-Ping 825, 837, 1620
Chen, Yu-Hung 681
Choe, Heemahn 850
Choi, Sung-Soon 850, 862, 1998
Choi, Yoon-Seok 2010
Chong, H.W. 2396
Chongstitvatana, Prabhas 1566
Chuang, Yue-Ru 681
Clark, John A. 146, 2022
Clergue, Manuel 1788
Coello Coello, Carlos A. 158, 573, 640
Collard, Philippe 1788
Collins, J.J. 1320
Cruz Cortés, Nareli 158

Author Index

- Culberson, Joseph 948
Cutello, Vincenzo 171, 1570
Czech, Zbigniew J. 142
- D'Eleuterio, Gabriele M.T. 377
Daida, Jason M. 1639, 1652, 1665
Dasgupta, Dipankar 183, 195, 219, 1580
Dawson, Devon 1870
Delbem, A.C.B. 634
Dick, Grant 1572
Dijk, Steven van 886
Divina, Federico 898, 1574
Doerner, Karl 2465
Dongarra, Jack 1015
Doya, Kenji 507
Dozier, Gerry 561
Drake, Stephen 1576
Droste, Stefan 909
Dubreuil, Marc 1578
- Eikelder, Huub M.M. ten 344
Elfwing, Stefan 507
Elliott, Lionel 2046
Eppstein, Maggie 967
Espinoza, Felipe P. 922
Esterline, Albert 657
Ewald, Claus-Peter 1963
- Falke II, William Joseph 1920
Fan, Zhun 1029, 1764, 2058
Fatiregun, Deji 2511
Feinstein, Mark 61
Ferguson, Michael I. 442
Fernández, Francisco 1812, 2109
Ficici, Sevan G. 286
Fischer, Marco 2398
Forman, Sean L. 2072
Foster, James A. 456, 2313
Fry, Rodney 1804
Fu, Zhaohui 1986
Fukami, Kiichiro 2134
Furutani, Hiroshi 934
- Gaag, Linda C. van der 886
Gagné, Christian 1578
Galeano, Germán 1812
Gallagher, John C. 431, 454
Galos, Peter 1806
Gao, Yong 948
Garnica, O. 2109
- Garrell, Josep Maria 1818
Garrett, Deon 1469
Garway-Heath, David 2360
Garzon, Max H. 379, 413
Gérard, Pierre 1882
Giacobini, Mario 955
Gilbert, Joshua 967
Giráldez, Raúl 979
Goldberg, David E. 801, 825, 837, 922, 1172, 1271, 1332, 1554, 1620, 1844, 1857, 1906
Gómez, D. 243
Gomez, Faustino J. 2084
Gómez, Jonatan 195, 1580
González, Fabio 195, 219, 1580
Goodman, Erik D. 1029, 1764, 2058, 2121
Green, James 1975
Greene, William A. 1582
Guinot, C. 140
Guo, Haipeng 1584
Guo, Pei F. 322
Gustafson, Steven 1800
Gutjahr, Walter J. 2465
Gwaltney, David A. 442
- Hahn, Lance W. 2412
Hallam, Jonathan 1586
Hamza, Karim 2096
Han, Kuk-Hyun 427, 2147
Hanby, V.I. 1938
Handa, Hisashi 991
Hang, Dehua 13
Harman, Mark 2511, 2513
Hart, Emma 1295
Hayasaka, Taichi 1600
Heckendorn, Robert B. 1003
Hermida, R. 2109
Hernández Aguirre, Arturo 573, 2264
Heywood, Malcolm I. 2325
Hidalgo, J.I. 2109
Hierons, Robert 2511, 2513
Hilbers, Peter A.J. 344
Hilss, Adam M. 1639, 1652, 1665
Hiroshima, Koji 2134
Hiroyasu, Tomoyuki 1015
Holcombe, Mike 2488
Holder, Mike 2400
Holifield, Gregory A. 1588
Homaifar, Abdollah 657

- Horn, Jeffrey 298
Hornby, Gregory S. 1678
Howard, Daniel 1690
Hsu, William H. 1584
Hu, Jianjun 1029, 1764, 2058
Hu, Shangxu 134
Huang, Chien-Feng 1041
Huang, Gaofeng 1053
Huang, Yong 2121
Huang, Zhijian 2121
Hussain, Mudassar 657
- Iba, Hitoshi 470, 1259, 1715, 1816, 1926, 2288
Ichikawa, Kazuo 2134
Ichikawa, Manabu 2134
Ingham, Derek B. 2046
Ishibuchi, Hisao 1065, 1077, 1234
Isono, Katsunori 2288
- Jacob, Jeremy L. 2022
Jähn, Hendrik 2398
Jang, Jun-Su 2147
Jansen, Thomas 310
Jin, Yaochu 636
Johnson, Paul 156
Jong, Edwin D. de 262, 274
Jong, Kenneth De 1604
Julstrom, Bryant A. 2402
Just, Winfried 154
- Kaegi, Simon 122
Kaige, Shiori 1234
Kalisker, Tom 1590
Kamio, Shotaro 470
Karr, Charles L. 2157, 2400, 2404, 2406
Kavka, Carlos 1089
Keijzer, Maarten 898, 1574, 1752
Kelsey, Johnny 207
Kendall, Graham 1800
Kharma, Nawwaf 322
Khoshgoftaar, Taghi M. 1808
Kim, Jong-Hwan 427, 638, 2147
Kim, Jong-Pil 2408
Kim, Jung-Hwan 1101
Kim, Keum Joo 642
Kim, Yong-Geon 2426
Kim, Yong-Hyuk 1112, 1123, 1136, 1345, 2168, 2215, 2410
Kimiaghalam, Bahram 657
- Klein, Jon 61
Klinkenberg, Ralf 1606
Kohl, Nate 356
Kondo, Shoji 2134
Korczak, Jerzy 369
Kordon, Arthur 1975
Kramer, Gregory R. 454
Krawiec, Krzysztof 332
Kumar, Rajeev 1592, 2179
Kumar, Sujay V. 730
Kwan, Raymond S.K. 693
Kwok, N.M. 2191
Kwon, Yung-Keun 1112, 2203, 2426
Kwong, Sam 2191, 2396
Kyne, Adrian G. 2046
- Labroche, Nicolas 25
Lai, Eugene 681
Lanchares, J. 2109
Langdon, W.B. 1702
Lanzi, Pier Luca 1894, 1922
Lattaud, Claude 144
Le Bris, Claude 512
Lee, Chi-Ho 638
Lee, Kin Hong 1802, 1918
Lee, Seung-Kyu 2168, 2215
Lefley, Martin 2477
Leier, André 390
Lemonge, Afonso C.C. 718
Leung, Kwong-Sak 585, 1160, 1802, 1918
Li, Gaoping 2121
Li, Hsiaolei 1665
Li, Xiaodong 37
Liang, Yong 585, 1160
Liekens, Anthony M.L. 344
Lim, Andrew 1053, 1594, 1596, 1986
Lin, Kuo-Chi 1541
Lin, Yingqiang 2227
Lipson, Hod 1518
Liszkai, Tamás 2418
Liu, Hongwei 1715
Liu, Xiaohui 2360
Liu, Yi 1808
Lizárraga, Giovanni 573
Llorà, Xavier 1172
Long, Stephen L. 1652
Louis, Sushil J. 2424
Lu, Ming 1148
Luke, Sean 1729, 1740

Author Index

- Mahdavi, Kiarash 2513
Majumdar, Nivedita Sumi 183
Mancoridis, Spiros 2499
Marchiori, Elena 898, 1574
Mărginean, Flaviu Adrian 1184
Marín-Blázquez, Javier G. 1295
Markon, Sandor 1963
Marshall, Kenric 1975
Martín-Vide, Carlos 401
Masahiro, Hiji 2337
Masaru, Tezuka 2337
Matsui, Shouichi 1598, 2240
Mauch, Holger 1810
Mazza, Raymond H. 2034
McMinn, Phil 2488
McQuay, Brian N. 2384
Mera, Nicolae S. 2046
Messimer, Sherri 2406
Mezura-Montes, Efrén 640
Miikkulainen, Risto 356, 2084
Miki, Mitsunori 1015
Minaei-Bidgoli, Behrouz 2252
Minsker, Barbara S. 922
Misevicius, Alfonsas 598
Mistry, Paavan 693
Mitavskiy, Boris 1196
Mitchell, Brian S. 2499
Mitrana, Victor 401
Mitsuhashi, Hideyuki 470
Mohan, Chilukuri 110
Monmarché, Nicolas 25, 140
Moon, Byung-Ro 669, 850, 862, 1101, 1112, 1123, 1136, 1345, 1357, 1998, 2010, 2168, 2203, 2215, 2408, 2410, 2426
Moore, Jason H. 2277, 2412
Morrison, Ronald W. 1210
Moura Oliveira, P.B. de 510
Mueller, Rainer 742
Munetomo, Masaharu 1222, 1616
Murao, Naoya 1222
Murata, Tadahiko 1234
Myodo, Emi 2264
Myung, Hyun 638

Nasraoui, Olfa 219
Neel, Andrew 379
Nguyen, Minh Ha 483
Nicolau, Miguel 1752
Nicosia, Giuseppe 171
Niño, F. 243

Nordin, Peter 495, 1806
Northern, James III 2414

Ocenasek, Jiri 1247
Oda, Terri 122, 231
Ofria, Charles 13
Oka, Kazuhiro 1814
Okabe, Tatsuya 636
Olsen, Anne 2416
Olsén, Joel 1806
Olsen, Nancy 2277
Oppacher, Franz 1481, 1493
Orla Kimbrough, Steven 1148
Osadciw, Lisa Ann 110

Palacios-Durazo, Ramón Alfonso 371
Palmes, Paulito P. 1600
Panait, Liviu 1729, 1740
Pappalardo, Francesco 1570
Parizeau, Marc 1578
Park, Sung-Joon 1602
Paul, Topon Kumar 1259
Pauw, Guy De 549
Pavone, Mario 171
Pei, Min 2121
Pelikan, Martin 1247, 1271
Peram, Thanmaya 110
Perego, R. 2109
Pérez-Jiménez, Mario J. 401
Perry, Chris 61
Pohlheim, Hartmut 2428
Pollack, Jordan B. 250, 274, 286
Popovici, Elena 1604
Pourkashanian, Mohamed 2046
Powell, David J. 2347
Prügel-Bennett, Adam 1586
Punch, William F. 1431, 2252

Raich, Anne 2418
Ramos, Fernando 429
Ramos, Fernando Manuel 375
Ranji Ranjithan, S. 1622
Rasheed, Khaled 813
Reif, David M. 2277
Reyes-Luna, Juan F. 2096
Ringnér, Kristofer Sundén 1806
Riquelme, José C. 979
Ritthoff, Oliver 1606
Rockett, Peter 1592
Rodebaugh, Brandon 148

- Rodrigues, Brian 1594, 1986
 Rodriguez-Tello, Eduardo 1283
 Rojas, Carlos 219
 Rosenberg, Ronald C. 1029, 1764, 2058
 Ross, Peter 1295, 1920
 Rothlauf, Franz 1307, 1608
 Rowe, Jonathan E. 874, 1505
 Russell, Matthew 146
 Ruwei, Dai 152
 Ryan, Conor 1320, 1626, 1752
- Sadeghipour, Sadegh 2428
 Saeki, Shusuke 2288
 Saitou, Kazuhiro 2096
 Sánchez, J.M. 2109
 Sancho-Caparrini, Fernando 401
 Sano, Masaki 1015
 Santos, Eugene Jr. 642
 Santos, Eunice E. 642
 Sarafis, Ioannis 2301
 Sastry, Kumara 1332, 1554, 1857
 Sayyarodsari, Bijan 657
 Schiavone, Guy 1541
 Schmeck, Hartmut 1568
 Schmidt, Christian 766
 Schmidt, Thomas M. 13
 Schmitt, Lothar M. 373
 Schoenauer, Marc 512, 1089
 Schulenburg, Sonia 1295
 Schwarz, Josef 1247
 Sciortino, John C. Jr. 2384
 Scott, Douglas A. 2404
 Sendhoff, Bernhard 636
 Seo, Dong-Il 669, 1345, 1357
 Seo, Kisung 1029, 1764, 2058
 Seredyński, Franciszek 1369
 Settles, Matthew 148
 Shanblatt, Michael 2414
 Shepherd, Rob 456
 Shepperd, Martin J. 2477
 Sheu, Shiann-Tsong 681
 Shibata, Youhei 1065
 Shimohara, Katsunori 74
 Shimosaka, Hisashi 1015
 Shyu, Conrad 2313
 Sigaud, Olivier 1882
 Silva, Sara 1776
 Singh, Vishnu 2157
 Slimane, M. 140
 Smith, R.E. 1382
- Smith, Robert E. 778
 Soak, Sang-Moon 1610
 Socha, Krzysztof 49
 Solteiro Pires, E.J. 510
 Song, Dong 2325
 Soule, Terence 148
 Sousa, Fabiano Luis de 375
 Spector, Lee 61
 Stacey, A. 2422
 Stein, Gunnar 644
 Stein, Michael 1568
 Stephens, Christopher R. 874, 1394
 Stepney, Susan 146, 2022
 Sthamer, Harmen 2442
 Stone, Christopher 1924
 Stone, Peter 356
 Storch, Tobias 1406
 Streeter, Matthew J. 1418
 Streichert, Felix 610, 644
 Suen, Ching Y. 322
 Sun, Dazhi 2420
 Suzuki, Tomoya 1814
- Takahashi, Katsutoshi 2288
 Takama, Yasufumi 246
 Tanabe, Shoko 2134
 Tanaka, Kiyoshi 2134, 2264
 Tanev, Ivan 74
 Tang, Ricky 1665
 Tarakanov, Alexander O. 248
 Teich, Tobias 2398
 Tekol, Yüce 695
 Tenreiro Machado, J.A. 510
 Teo, Jason 483, 1612
 Thangavelautham, Jekanthan 377
 Tharakunnel, Kurian K. 1906
 Thierens, Dirk 886
 Tian, Lirong 1614
 Timmis, Jon 207
 Tokoro, Ken-ichi 1598, 2240
 Tomassini, Marco 955, 1788, 1812, 2109
 Tominaga, Kazuto 1814
 Tong, Siu 2347
 Topchy, Alexander 1431
 Torng, Eric 13
 Torres-Jimenez, Jose 1283
 Toussaint, Marc 86, 1444
 Trinder, Phil 2301
 Truong, T.Q.S. 2422
 Tsuji, Miwako 1616

Author Index

- Tsutsui, Shigeyoshi 1015
Tucker, Allan 2360
Tyrrell, Andy 1804
- Uchibe, Eiji 507
Ueno, Yutaka 2288
Ulmer, Holger 610, 644
Upal, M. Afzal 98
Usui, Shiro 1600
- Valenzuela-Rendón, Manuel 371, 1457
Vallejo, Edgar E. 429
Vanneschi, Leonardo 1788, 1812
Veeramachaneni, Kalyan 110
Vejar, R. 243
Venturini, Gilles 25, 140
Vigraham, Saranyan 431
Vlassov, Valeri 375
Vose, Michael D. 1505
- Wakaki, Hiromi 1816
Wallace, Jeff 2424
Waller, S. Travis 2420
Wallin, David 1320
Wang, Wei 537
Ward, David J. 1652
Watanabe, Isamu 1598, 2240
Watson, Jean-Paul 1469
Wegener, Ingo 622, 1406
West, Michael 413
White, Bill C. 2277
White, Tony 122, 231
Whiteson, Shimon 356
Whitley, Darrell 1469
Wieczorek, Wojciech 142
Wiegand, R. Paul 310
Wilson, Christopher W. 2046
Wilson, Eric 2406
Wineberg, Mark 1481, 1493
Witt, Carsten 622
Wolff, Krister 495
- Wood, David Harlan 1148
Wright, Alden H. 1003, 1505
Wright, J.A. 1938
Wu, Annie S. 1541, 1588, 2384
Wu, D.J. 1148
Wyatt, Danica 1518
- Xianghui, Dong 152
Xiao, Fei 1594
Xiao, Ningchuan 1530
Xiao, Ying 154
Xu, Zhou 1596
Xuan, Jiang 813
- Yamamoto, Takashi 1077
Yamamura, Masayuki 1602
Yang, Jinn-Moon 2372
Yang, Seung-Jin 2426
Yang, Shengxiang 1618
Yassine, Ali 1620
Yilmaz, Ayse S. 2384
Yu, Han 1541, 2384
Yu, Huanjun 134
Yu, Senhua 183
Yu, Tian-Li 1554, 1620
Yu, Tina 156
Yuan, Xiaohui 1624
Yuchi, Ming 638
Yue, Yading 2072
- Zalzala, Ali 2301
Zamora, Adolfo 1394
Zechman, Emily M. 1622
Zell, Andreas 610, 644
Zhang, Jian 1624
Zhang, Liping 134
Zhang, Y. 1938
Ziemke, Tom 458
Zincir-Heywood, A. Nur 2325
Zomaya, Albert Y. 1369