Lecture Notes in Computer Science 2936

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

Springer
Berlin
Heidelberg
New York
Hong Kong
London
Milan
Paris
Tokyo

Pierre Liardet Pierre Collet
Cyril Fonlupt Evelyne Lutton
Marc Schoenauer (Eds.)

Artificial Evolution

6th International Conference, Evolution Artificielle, EA 2003
Marseille, France, October 27-30, 2003
Revised Selected Papers

Springer

Volume Editors

Pierre Liardet
Université de Provence, Centre de Mathématiques,
Informatique et Mécanique, Laboratoire LATP, UMR-CNRS 6632
39, rue F. Joliot-Curie, 13453 Marseille cedex 13, France
E-mail: liardet@cmi.univ-mrs.fr

Pierre Collet
Cyril Fonlupt
Université du Littoral Côte d'Opale
Laboratoire d'Informatique du Littoral
BP 719, 62228 Calais Cedex, France
E-mail: Pierre.Collet@inria.fr,fonlupt@lil.univ-littoral.fr

Evelyne Lutton
INRIA - Rocquencourt
B.P. 105, 78153 Le Chesnay Cedex, France
E-mail: Evelyne.Lutton@inria.fr

Marc Schoenauer
Université Paris-Sud, Equipe TAO - INRIA Futurs, LRI
Bat 490, 91405 Orsay Cedex, France
E-mail: Marc.Schoenauer@inria.fr

Library of Congress Control Number: 2004103455

CR Subject Classification (1998): F.1, F.2.2, I.2.6, I.5.1, G.1.6, J.3

ISSN 0302-9743
ISBN 3-540-21523-9 Springer-Verlag Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by PTP-Berlin, Protago-TeX-Production GmbH
Printed on acid-free paper SPIN: 10992227 06/3142 5 4 3 2 1 0

Preface

This book is based on the papers presented at the 6th Conference *Evolution Artificielle*, EA 2003, held in Marseilles. Previous EA meetings took place in Toulouse (1994), Brest (1995), Nîmes (1997), Dunkerque (1999), and Le Creusot (2001), while the best presented papers were published in Springer's LNCS, volumes 1063, 1363, 1829, and 2310.
This year again, authors were invited to present original works, relevant to simulated evolution, including but not limited to evolutionary optimization and learning, theory of evolutionary computation, artificial life, population dynamics, and implementation and real-world applications of evolutionary paradigms. EA 2003 turned out to be very international, and the proposed papers were selected through a rigorous process, each of them being examined by at least three referees from the international program committee. We are greatly indebted to each of them for the hard work and time they spent to ensure a high quality of editing in this volume. We also would like to thank all the participants of the conference and the numerous authors who chose to submit their manuscripts.

We are particularly grateful to Prof. Hans-Paul Schwelfel for his invited lecture on *Genesis and Future of Evolutionary Computation* which presented a wide survey of the art with a deep analysis, promising many further interesting applications.

Contributions in this book were organized into nine parts according to their main topics, and these are briefly presented below.

1. **Theoretical Issues:** Variable-length representations in evolutionary algorithms are investigated by M. Defoin Platel et al. using parametrical models in between the royal road landscapes and the NK landscapes according to Kauffman. M. Nicolau, A. Auger and C. Ryan explore the mathematical formalization of an encoding process using grammatical evolution and give asymptotics estimates and experimental results. A basic but general method to maximize many fitness functions is to choose – at random – an initial point and apply random perturbations (following a given or updated random process) to it in order to improve the fitness. The response to the increase in the number of variables for such perturbations is studied by L. Grosset, R. Le Riche, and R.T. Haftka, both for random stochastic hill climber and univariate marginal distribution algorithms, while S. Aupetit, P. Liardet and M. Slimane applied such a method to produce binary sequences with low out-of-phase aperiodic autocorrelations.
   Asymptotic distribution laws of upper-order statistics were used by S. Puechmorel and D. Delahaye for improving the efficiency of selection and crossover operators. The last theoretical contribution, by M. Drugan and D. Thierens, was a survey on evolutionary Markov Chain Monte Carlo algorithms

related to evolutionary schemes; they proposed a new algorithm relying on elitist coupled acceptance rules.

2. **Algorithmic Issues:** Constraint satisfaction problems are considered by V. Barichard et al. in a general evolutionary framework based on the collaboration between constraint propagation techniques and local search methods. R. Baños et al. proposed a new parallel evolutionary algorithm for graph partitioning that mixed simulated annealing, tabu search and selection mechanisms. F. Lardeux, F. Saubion and J.-K. Hao investigated hybrid evolutionary algorithms involving recombination operators specially designed for SAT problems. This part ended with the presentation of B. Sareni, J. Regnier and X. Roboam on the efficiency of some crossover operators associated with self-adaptive procedures in the case of real-encoded multiobjective genetic algorithms.
3. **Applications:** The difficult technological problem of optical fiber alignment was efficiently solved using new genetic algorithms by M. Murakawa, H. Nosato and T. Higuchi. They incorporated a local-learning method to accelerate the alignment. K. Deb and A.R. Reddy dealt with a very large-sized scheduling problem often encountered in automated foundries: find an optimal sequence for casting a number of molds. A method for creating image classification is described by J. Korczak and A. Quirin. They used evolutionary operators applied to remote sensing data and derived well-adapted classification rules to recognize large objects on the image as well as the smaller ones. M. Segond et al. adapted an ant algorithm to detect vortex structures in coastal waters, using data from hydrodynamical simulations of the stream circulation. In order to speed up the selection of the flight aircraft by the controllers, D. Delahaye and S. Puechmorel presented a virtual keyboard whose optimal efficiency is related to an NP-hard assignment problem. They applied it to real instances. A.B. Garmendia-Doval, S.D. Morley and S. Juhos proposed and applied cartesian genetic programming to evolve postdocking filters automatically for removing false positives from virtual hit sets.
4. **Implementation Issues:** A number of various evolutionary algorithms are encountered in the literature. The graphic user interface (GUIDE) of P. Collet and M. Schoenauer unified all kinds of such algorithms and gave us facilities to create quite new combinations. ParaDisEO is a framework that uses evolving objects in parallel and distributed computing which is efficiently applied for large combinatorial problems. S. Cahon et al. made use of this approach in three evolutionary models involving asynchronous migration and parallel or distributed evaluation, experimenting on two real-word problems. Y. Yang, J. Vincent and G. Littlefair proposed a new model of coarse-grained parallel genetic algorithm managing clustered groupings. They tested their approach on multimodal optimization problems.
5. **Genetic Programming:** Maintaining a structural diversity and controlling the code size are the main challenges in genetic programming, a promising variant of evolutionary programming which produces solutions to problems in the form of computer programs. A measure of genotypic and phenoty-

pic diversity, based on the notions of entropy and variance, was applied by M. Tomassini et al. to three standard test problems, related to even parity, symbolic regression and artificial ants, while B. Wyns, S. Sette and L. Boullart introduced a self-improvement operator to reduce the effects of code growth.

Discrepancies, constantly encountered in genetic programming when the learning result issuing from training data is specialized over the entire distribution of instances, lead to the so-called overfitting that G. Paris, D. Robilliard and C. Fonlupt explored on two benchmarks, and they proposed guidelines to reduce its effect.

6. **Coevolution and Agent Systems:** Evolutionary processes may be issued from various paradigms. In auction theory A.J. Bagnall and I. Toft gave an adaptive agent model using a basic learning mechanism for optimal strategies in common auction scenarios. The multipopulation approach was used by F. Streichert et al. to extract global and local optima in multimodal search spaces: they proposed a niching-like method associated with cluster analysis for identifying species from an initially undifferentiated population. In their contribution R. Groß and M. Dorigo showed how to obtain an efficient cooperative transport of a pray using two simple autonomous mobile robots without any intercommunication facility and using very limited computing resources.
7. **Artificial Life:** The plant model considered by C. Lattaud is based on multiagent systems, each agent being a plant organ and each plant being composed either of a unique organ or of three organs. This approach allowed us to define interactions between plants through a diffusion process of chemical substances (allellopathy model) and to simulate evolution of artificial plant communities. M. Annunziato et al. proposed an on-line adaptive control and optimization of complex processes, based on artificial environments.
8. **Cellular Automata:** The analysis presented by M. Giacobini, M. Tomassini and A. Tettamanzi concerned the takeover time for selection mechanisms applied on a spatially structured population, involving circular cellular automata with synchronous or asynchronous propagation. Their theoretical approaches are corroborated by empirical results. A class of 2-D cellular automata are studied by E. Sapin, O. Bailleux and J.-J. Chabier to search transition rules leading to automata that can be used to simulate the logic AND and NOT gates.
9. **Machine Learning:** Methods for learning integrate evolutionary strategy and cover a large range of applications. M.C. Codrea et al. proposed a feature-learning algorithm which does an automatic identification of plant species from their fluorescence induction curves. Receiver operating characteristics (ROC) curves figure out the dependency of the true positive rate with respect to the false positive one in test interpretations, and the area under the curve (AUC) is a popular learning criterion in medical data analysis. M. Sebag, J. Azé and N. Lucas presented the ROGER algorithm, implementing an evolution strategy based on optimization of the AUC criterion. Its per-

formances were compared to those issuing from a support vector machine, namely SVMTorch. The last contribution, by D. Kazakov and M. Bartlett, dealt with a model that simulated the evolution of language on the basis of learning communication systems.

We take this opportunity to thank the different partners whose financial and material support contributed to the success of the conference: the *Université de Provence*, the *Institut National de Recherche en Informatique et en Automatique* (INRIA), and the *Centre National de la Recherche Scientifique* (CNRS), with the LATP (*Laboratoire d'Analyse, Topologie et Probabilité*) and the CNRS *Groupe de Recherche ALP* (Algorithmique Langage et Programmation), the *Association Française d'Intelligence Artificielle*, and the Association *Évolution Artificielle*.

EA 2003 took place at the *Institut Universitaire de la Formation des Maîtres* (IUFM) nicely located on *La Canebière*. We are indebted to Mme Catherine Ponsin-Costa, administrative manager, and her team Mme Shirley Chemouny, Max Laffont and Xavier Campagna, for their particular kindness.

Finally, we wish to express our gratitude to Josy Liardet and Valérie Collet for their efficiency and enthusiasm in setting up the conference. LATP and INRIA linked their sponsorship to their efficient teams: Aline Blanc and Marie-Christine Tort from Marseilles; and Nathalie Gaudechoux, Dominique Potherat, Chantal Girodon and Marie-Jo Cardet from Rocquencourt. Many thanks to all of them as well as to Mario Giacobini from Lausanne for his general and efficient help.

January 2004

Pierre Liardet
Pierre Collet
Cyril Fonlupt
Evelyne Lutton
Marc Schoenauer

Evolution Artificielle 2003 – EA 2003

October 27–30, 2003

Université de Provence, Marseilles, France

6th International Conference on Artificial Evolution

Organizing Committee

Pierre Liardet (Université de Provence)
Pierre Collet (LIL Calais)
Cyril Fonlupt (LIL Calais)
Evelyne Lutton (INRIA Rocquencourt)
Marc Schoenauer (INRIA Rocquencourt)

Program Committee

J.-M. Alliot (ENAC, Toulouse) – O. Bailleux (Univ. Bourgogne)
P. Bessière (IMAG, Grenoble) – A. Blair (Univ. New South Wales)
B. Braunschweig (IFP, Rueil-Malmaison) – J.-J. Chabrier (LIRSIA, Dijon)
P. Collard (I3S, Nice) – P. Collet (LIL, Calais)
M. Corne (Univ. of Reading) – K. Deb (IIT Kanpur)
D. Delahaye (CENA, Toulouse) – C. Dhaenens (LIFL, Lille)
M. Dorigo (ULB, Bruxelles) – N. Durand (ENAC, Toulouse)
M. Ebner (Univ. Wuerzburg) – D. Fogel (Nat. Selection Inc., La Jolla)
C. Fonlupt (LIL, Calais) – O. François (Ensimag, Grenoble)
P. Galinier (Ecole Poly., Montréal) – J. Gottlieb (SAP AG, Walldorf)
G. Granger (ENAC, Toulouse) – W. Hart (Sandia Nat. Lab.)
M. Keijzer (Vrije Univ. Amsterdam) – N. Krasnogor (Univ. West of England)
J.-K. Hao (Univ. Angers) – J.-C. Heudin (Univ. L. de Vinci, La Défense)
C. Lattaud (Paris V) – R. Le Riche (École des Mines, St.-Étienne)
P. Liardet (Univ. Provence, Marseille) – B. Leblanc (Mat. Design, Le Mans)
J. Louchet (INRIA, Rocquencourt) – E. Lutton (INRIA, Rocquencourt)
S. Luke (G. Mason Univ., Virginia) – N. Monmarché (EPU, Univ. Tours)
N. Melab (LIFL, Lille) – P. Preux (LIL, Calais)
N. Radcliffe (Quadstone Ltd., Edinburgh) – E. Ramat (LIL, Calais)
D. Robilliard (LIL, Calais) – E. Ronald (CMAP & EPFL, Lausanne)
G. Rudolph (Parsytec AG, Aachen) – M. Schoenauer (INRIA, Rocquencourt)
M. Sebag (LMS Palaiseau & LRI Orsay) – M. Slimane (EPU, Univ. Tours)
E.-G. Talbi (LIFL, Lille) – S. Tsutsui (Hannan Univ.)
G. Venturini (EPU, Univ. Tours)

Sponsoring Institutions

Université de Provence
INRIA (Institut National de Recherche en Informatique et en Automatique)
CNRS (Centre National de la Recherche Scientifique)
AFIA (Association Française d'Intelligence Artificielle)

Table of Contents

Applications

Implementation Issues

Genetic Programming

From Royal Road to Epistatic Road for Variable Length Evolution Algorithm

Michael Defoin Platel[1,2], Sebastien Verel[1], Manuel Clergue[1], and Philippe Collard[1]

[1] Laboratoire I3S, CNRS-Université de Nice Sophia Antipolis
[2] ACRI-ST

Abstract. Although there are some real world applications where the use of variable length representation (VLR) in Evolutionary Algorithm is natural and suitable, an academic framework is lacking for such representations. In this work we propose a family of tunable fitness landscapes based on VLR of genotypes. The fitness landscapes we propose possess a tunable degree of both neutrality and epistasis; they are inspired, on the one hand by the Royal Road fitness landscapes, and the other hand by the NK fitness landscapes. So these landscapes offer a scale of continuity from Royal Road functions, with neutrality and no epistasis, to landscapes with a large amount of epistasis and no redundancy. To gain insight into these fitness landscapes, we first use standard tools such as adaptive walks and correlation length. Second, we evaluate the performances of evolutionary algorithms on these landscapes for various values of the neutral and the epistatic parameters; the results allow us to correlate the performances with the expected degrees of neutrality and epistasis.

1 Introduction

Individuals in Genetic Algorithms (GA) are generally represented with strings of fixed length and each position of the string corresponds to one gene. So, the number of genes is fixed and each of them can take a fixed number of values (often 0 and 1). In variable length representation (VLR), like Messy GA or Genetic Programming, genotypes have a variable number of genes. Here, we consider VLR where a genotype is a sequence of symbols drawn from a finite alphabet and a gene is a given sub-sequence of such symbols. The main difference with fixed length representation is that a gene is identified by its form and not by its absolute position in the genotype.

Some specific obstacles come with the variable length paradigm. One of the most important is the identification of genes. Indeed, during recombination, genes are supposed to be exchanged with others that represent similar features. So the question of the design of suitable crossover operators becomes essential (see for example [1]). Another difficulty due to variable length is the tremendous amount of neutrality of the search space, as noted in [2]. Neutrality appears at different levels. First, a gene may be located at different positions in the

P. Liardet et al. (Eds.): EA 2003, LNCS 2936, pp. 3–14, 2004.

genotype. Second, some parts of genotype (called introns) do not perform any functions and so do not contribute to fitness. The last specificity is that variable length strings introduce a new dimension in the search space, which have to be carefully explored during evolution to find regions where fitter individuals prevail. The exploration of sizes seems to be difficult to handle and may lead, as in Genetic Programming, to an uncontrolled growth of individuals (a phenomenon called bloat [3]).

One of the major concerns in the GA field is to characterize the difficulty of problems. One way to achieve this is to design problems with parameters controlling the main features of the search space; to run the algorithm; and to exhibit how performances vary according to the parameters. With fixed length representations, some well known families exist, as the Royal Road functions, where inherent neutrality is controlled by the block size, or the NK-landscapes, where the tunable parameter K controls the ruggedness of the search space. With VLR, there are only a few attempts to design such academic frameworks[4]. Note, for example, the Royal Tree [5] and the Royal Road for Linear GP [1].

2 Royal Road for Variable Length Representation

In GA, Royal Road landscapes (RR) were originally designed to describe how building blocks are combined to produce fitter and fitter solutions and to investigate how the schemata evolution actually takes place [6]. Little work is related to RR in variable length EA; e.g. the Royal Tree Problem [5] which is an attempt to develop a benchmark for Tree-based Genetic Programming and which has been used in Clergue et al. [7] to study problem difficulty. To the best of our knowledge, there was no such work with linear structures.

In a previous work, we have proposed a new kind of fitness landscape [1], called Royal Road landscapes for variable length EA (VLR Royal Road). Our aim was to study the behavior of a crossover operator during evolution. To achieve this goal, we needed experiments able to highlight the destructive (or constructive) effects of crossover on building blocks.

To define VLR Royal Road, we have chosen a family of optimal genotypes and have broken them into a set of small building blocks. Formally, the set of optima is:

$$\{g \in G_\Sigma \mid \forall l \in \Sigma,\, B_b(g,l) = 1\},$$

with

$$B_b(g,l) = \begin{cases} 1 \text{ if } \exists\, i \in [0, \lambda - b] \mid \forall\, j \in [0, b-1],\, g_{i+j} = l, \\ 0 \text{ otherwise,} \end{cases}$$

and

- $b \geq 1$ the size of blocks
- Σ an alphabet of size N that defines the set of all possible letters l per locus
- G_Σ the finite set of all genotypes of size $\lambda \leq \lambda_{max}$[1] defined over Σ

[1] λ_{max} have to be greater than Nb

- g a genotype of size $\lambda \leq \lambda_{max}$
- g_k the k^{th} locus of g.

The following genotype $g \in G_\Sigma$ is an example of optimum, with $\Sigma = \{A, T, G, C\}$ and $b = 3$:

$$g = \mathbf{AAA}GTA\mathbf{GGG}TAA\mathbf{TTTCCC}TCCC\,.$$

$B_b(g, l)$ acts as a predicate accounting for the presence (or the absence) of a contiguous sequence of a single letter (i.e. a block). Note that only the presence of a block is taken into account, neither its position nor its repetition. The *number of blocks* corresponds to the number of letters $l \in \Sigma$ for which $B_b(g, l)$ is equal to one. In the previous example, only boldfaced sequences contribute to fitness[2]. The contribution of each block is fixed and so, the fitness $f_{Nb}(g)$ of genotype $g \in G_\Sigma$ having n blocks is simply:

$$f_{Nb}(g) = \frac{1}{N} \sum_{i=1}^{N} B_b(g, l_i) = \frac{n}{N}\,.$$

To efficiently reach an optimum, the EA system has to create and combine blocks without breaking existing structures. These landscapes were designed in such a way that fitness degradation due to crossover may occur only when recombination sites are chosen inside blocks, and never in case of blocks translocations or concatenations. In other words, there is no inter blocks epistasis.

3 NK-Landscapes

Kauffman [8] designed a family of problems, the NK-landscapes, to explore how epistasis is linked to the 'ruggedness' of search spaces. Here, epistasis corresponds to the degree of interaction between genes, and ruggedness is related to local optima, their number and especially their density. In NK-landscapes, epistasis can be tuned by a single parameter. Hereafter, we give a more formal definition of NK-landscapes followed by a summary review of their properties.

3.1 Definition

The fitness function of a NK-landscape is a function $f_{NK} : \{0,1\}^N \rightarrow [0,1)$ defined on binary strings with N loci. Each locus i represents a gene with two possible alleles, 0 or 1. An 'atom' with fixed epistasis level is represented by a fitness components $f_i : \{0,1\}^{K+1} \rightarrow [0,1)$ associated to each locus i. It depends on the allele at locus i and also on the alleles at K other epistatic loci (K must fall between 0 and $N-1$). The fitness $f_{NK}(x)$ of $x \in \{0,1\}^N$ is the average of the values of the N fitness components f_i:

[2] Although the last sequence of 'CCC' is a valid block, it does not contribute to fitness since it is only another occurrence.

$$f_{NK}(x) = \frac{1}{N}\sum_{i=1}^{N} f_i(x_i; x_{i_1}, \ldots, x_{i_K})$$

where $\{i_1, \ldots, i_K\} \subset \{1, \ldots, i-1, i+1, \ldots, N\}$. Many ways have been proposed to choose the K other loci from N loci in the genotype. Two possibilities are mainly used: adjacent and random neighborhoods. With an adjacent neighborhood, the K genes nearest to the locus i are chosen (the genotype is taken to have periodic boundaries). With a random neighborhood, the K genes are chosen randomly on the genotype. Each fitness component f_i is specified by extension, ie a number $y_{i,(x_i;x_{i_1},\ldots,x_{i_K})}$ from $[0, 1)$ is associated with each element $(x_i; x_{i_1}, \ldots, x_{i_K})$ from $\{0, 1\}^{K+1}$. Those numbers are uniformly distributed in the interval $[0, 1)$.

3.2 Properties

The NK-landscapes have been used to study links between epistasis and local optima. The definition of local optimum is relative to a distance metric or to a neighborhood choice. Here we consider that two strings of length N are neighbors if their Hamming distance is exactly one. A string is a local optimum if it is fitter than its neighbors.

The properties of NK-landscapes are given hereafter in term of local optima: their distribution of fitness, their number and their mutual distance. These results can be found in Kauffman[8], Weinberger[9], Fontana *et al.*[10].

- For $K = 0$ the fitness function becomes the classical additive multi-locus model, for which
  - There is single and attractive global optimum.
  - There always exists a fitter neighbor (except for global optimum).
  - Therefore the global optimum could be reach on average in $N/2$ adaptive steps.
- For $K = N - 1$, the fitness function is equivalent to a random assignment of fitnesses over the genotypic space, and so:
  - The probability that a genotype is a local optimum is $\frac{1}{N+1}$.
  - The expected number of local optima is $\frac{2^N}{N+1}$.
  - The average distance between local optima is approximately $2ln(N-1)$
- For K small, the highest local optima share many of their alleles in common.
- For K large:
  - The fitnesses of local optima are distributed with an asymptotically normal distribution with mean m and variance s approximately:

    $$m = \mu + \sigma\sqrt{\frac{2ln(K+1)}{K+1}}\ ,\ s = \frac{(K+1)\sigma^2}{N(K+1+2(K+2)ln(K+1))}$$

    where μ is the expected value of f_i and σ^2 its variance. In the case of the uniform distribution, $\mu = 1/2$ and $\sigma = \sqrt{1/12}$.
  - The average distance between local optima is approximately $\frac{Nlog_2(K+1)}{2(K+1)}$.
  - The autocorrelation function $\rho(s)$ and the correlation length τ are:

    $$\rho(s) = \left(1 - \frac{K+1}{N}\right)^s\ ,\ \tau = \frac{-1}{ln(1-\frac{K+1}{N})}\ .$$

4 Epistatic Road for Variable Length Representation

In this section, we define a problem with tunable difficulty for variable length EA, called Epistatic Road functions (ER). To do so, we propose to use the relation between epistasis and difficulty.

4.1 Definition

Individuals in a variable length representation may be viewed as sets of interacting genes. So, in order to model such a variable length search space, we have to first identify genes and second explicitly define their relations. This can be easily done by extending the VLR Royal Road thanks to dependencies between the fitness contributions of blocks. Thus, genes are designated as blocks and the contribution of a gene depends on the presence of others, exactly as in NK-landscapes.

More formally, the fitness function of an ER-landscape is a function $f_{NKb} : G_\Sigma \to [0, 1)$ defined on variable length genotypes. The fitness components f_i are defined in Sec. 3.1, and the fitness $f_{NKb}(g)$ of genotype $g \in G_\Sigma$ is the average of N fitness components f_i:

$$f_{NKb}(g) = \frac{1}{N} \sum_{i=1}^{N} f_i(B_b(g, l_i); B_b(g, l_{i_1}), \ldots, B_b(g, l_{i_K})) \,.$$

In practice, we use an implementation of NK-landscape with random neighborhood to compute f_i. We have to ensure that the set of all genotypes having N blocks corresponds to the end of the Road. For that purpose, first we exhaustively explore the space $\{0, 1\}^N$ to find the optimum value of the NK, then we permute this space in such a way that the optimum becomes 1^N.

4.2 Tunability

The properties of an ER-landscape depends on the three parameters N, K and b. Although these parameters are not entirely independent, each allows us to control a particular aspect of the landscape. Increasing the parameter N causes the size of both the search space and the neighborhood of genotype to increase. Moreover, as N determines the number of genes to find, the computational effort required to reach the optimum will be more important when high values of N are used. The parameter b controls the degree of neutrality. As b increases the size of iso-fitness sets increases. Finally, the parameter K allows to control the number of epistatic links between genes and so the number of local optima. For $K = 0$, an ER-landscape will be very closed to the corresponding VLR Royal Road since insertion of a new block in a genotype always increases the fitness. In contrast, for $K = N - 1$, with a high level of epistasis, the vast majority of the roads leads to local optima where the insertion of a new block in a genotype always decreases the fitness.

5 Fitness Landscape Analysis

Many measures have been developed to describe fitness landscapes in terms of "difficulty". Here, "difficulty" refers to the ability of a local heuristic to reach the optimum. In this section some of those metrics are applied to the ER-landscapes. In particular, we show how difficulty changes according to the three parameters N, K and b. The neighborhood of variable length genotypes is different from the neighborhood of fixed length genotypes. To define a neighborhood in ER-landscapes, we use String Edit Distance, like *Levenshtein distance* [11] which has been already used in GP to compute or control diversity [12], or to study the influence of genetic operators [13]. By definition, the Edit Distance between two genotypes corresponds to the minimal number of elementary operations (deletion, insertion and substitution) required to change one genotype into the other. So two strings in the search space are neighbors if the Edit Distance between them is equal to 1. Thus a string of length λ has $(2\lambda + 1)N$ neighbors.

In order to minimize the influence of the random creation of an NK-landscape, we take the average of the following measures over 10 different landscapes for each couple of parameters N and K. We have perform experiments for $N = 8$, 10 and 16, for K between 0 and $N - 1$ and for b between 1 and 5.

5.1 Random Walks, Autocorrelation Function, and Correlation Length

Weinberger[9,14] defined *autocorrelation function* and *correlation length* of random walks to measure the epistasis of fitness landscapes.
A random walk $\{g_t, g_{t+1}, \ldots\}$ is a series where g_t is the initial genotype and g_{i+1} is a randomly selected neighbor of g_i. Then the autocorrelation function ρ of a fitness function f is the autocorrelation function of the time series $\{f(g_t), f(g_{t+1}), \ldots\}$:

$$\rho(s) = \frac{\langle f(g_t) f(g_{t+s}) \rangle_t - \langle f \rangle^2}{var(f)} .$$

The correlation length τ measures how the correlation function decreases and so how rugged the landscape is. More rugged the landscape the shorter the correlation length.

$$\tau = -\frac{1}{ln(\rho(1))} .$$

Empirical measures on ER landscapes were performed on 20.10^3 random walks of length 35 for each triplet of parameters N, K, b and for each of 10 instances of NK-landscapes. The initial genotypes were generated by randomly choosing its length between 0 and λ_{max} and then randomly choosing each letter of the genotype. For those random walks, λ_{max} is equal to $2Nb$. For small values of b, the correlation length decreases quickly (when the parameter K

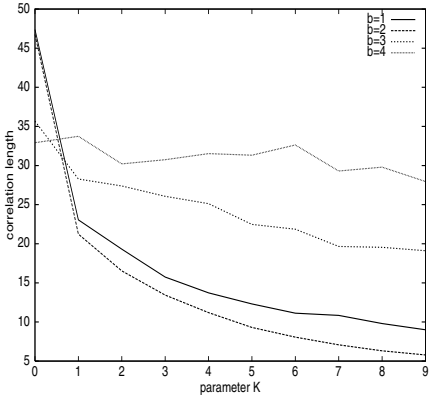


Fig. 1. Mean correlation length of ER-landscapes for $N = 10$

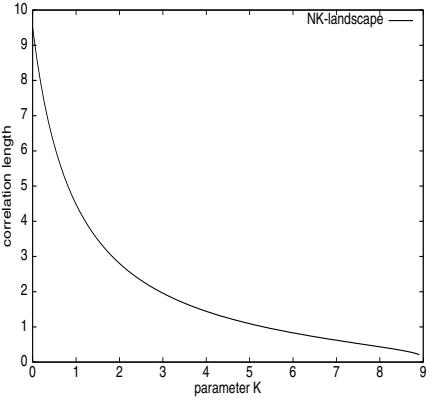


Fig. 2. Theoretical correlation length of NK-landscapes for $N = 10$

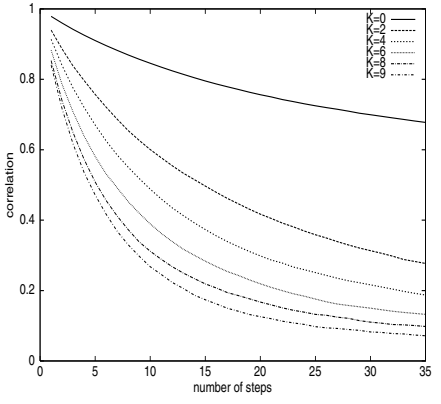


Fig. 3. Autocorrelation function of ER-landscape for $N = 10$

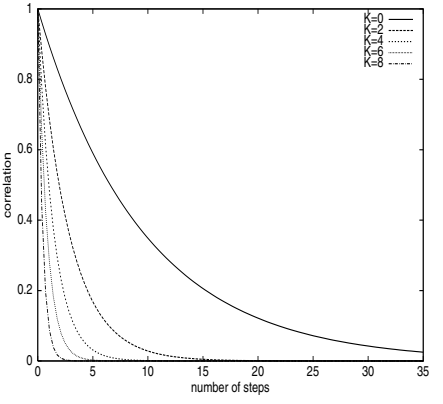


Fig. 4. Theoretical autocorrelation function of NK-landscape for $N = 10$

increases, see fig. 1 and 3). As expected, the correlation of fitness between genotypes decreases with the modality due to the parameter K. We can compare this variation with the theoretical correlation length of NK-landscapes, given in 3.2 (see fig. 2 and 4). As b increases, the influence of K on the correlation length decreases. Neutrality keeps a high level of correlation in spite of the increase in modality.

5.2 Adaptive Walks and Local Optima

Several variants of adaptive walk (often called myopic or greedy adaptive walk) exists. Here we use the series $\{g_t, g_{t+1}, \ldots, g_{t+l}\}$ where g_t is the initial genotype

and g_{i+1} is one of the fittest neighbor of g_i. The walk stops on g_{t+l} which is a local optimum. By computing several adaptive walks, we can estimate:

- The fitness distribution of local optima by the distribution of the final fitnesses $f(g_{t+l})$.
- The distance between local optima which is approximately twice the mean of the length l of those adaptive walks.

Empirical measurements on ER landscapes were performed on 2.10^3 random walks for each triplet of parameters N, K, b and for each of 10 instances of NK-landscapes. We used the same initialization procedure as the random walk. The parameter λ_{max} is set to 50. The distribution of local optima fitnesses is close to

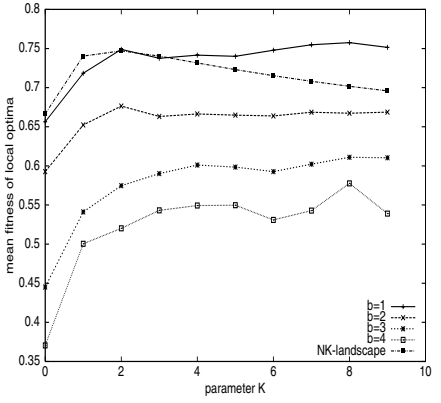


Fig. 5. Mean fitness of local optima of ER-landscapes obtained with adaptive walks for $N = 10$

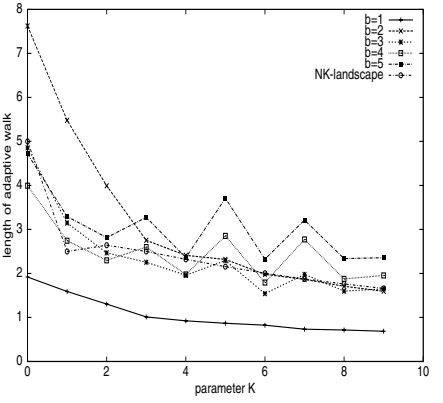


Fig. 6. Mean length of adaptive walks on ER-landscape for $N = 10$

normal distribution. The mean fitness of local optima is represented for $N = 10$ on Figure 5; it decreases with b. The variations of the fitness of local optima are great for small values of K but become almost insignificant for medium and high values of K. In Figure 6, the mean length of adaptive walks is represented for $N = 10$. As expected, it decreases with K for small values of b. So, the parameter K increases the ruggedness of the ER-landscape. On the other hand, when b is higher, K has less influence on the length of the walk. Indeed, the adaptive walk breaks off more often on neutral plateaux.

5.3 Neutrality

A random walk is used to measure the neutrality of ER-landscapes. At each step, the number of neighbors with lower, equal and higher fitness is counted. We perform 2.10^3 random walks of length 20 for each triplet of parameter N, K and b. The Table 1 gives the proportions of such neighbors for N=8, K=4

(they depend slightly on N and K) and for several values of b. The number of equally fit neighbors is always high and is maximum for b=4. So, neutral moves are a very important feature of ER-landscapes.

Table 1. Proportion of Lower, Equal and Higher neighbor

	$N = 8$, $K = 4$		
Block size	Lower	Equal	Higher
$b = 2$	7.2	85.8	7.0
$b = 3$	2.8	94.4	2.8
$b = 4$	0.5	98.9	0.6

6 EA Performances

In this section, we want to compare the performances of an evolutionary system on ER-landscapes for various settings of the three parameters N, K and b. The performances are measured by the success rate and the mean number of blocks found. In order to minimize the influence of the random creation of NK-landscapes, we take the average of these two measures over 10 different landscapes. 35 independent runs are performed with mutation and crossover rates of respectively 0.9 and 0.3 (as found in [1]). The standard one point crossover, which blindly swaps sub-sequences of parents, was used. Let us notice that a mutation rate of 0.9 means that each program involved in reproduction has a 0.9 probability to undergo one insertion, one deletion and one substitution. Populations of 1000 individuals were randomly created according to a maximum creation size of 50. The evolution, with elitism, maximum program size of 100(λ_{max}), 4-tournament selection, and steady-state replacement, took place during 400 generations.

6.1 Results

We have performed experiments for N=8, 10 and 16, for K between 0 and $N/2$ and for b between 2 and 5. We note that the case b=1 is not relevant because the optimum is always found at the first generations for all values of K. In Figure 8, we have reported the success rate (over 35×10 runs) as a function of K for N=8. As expected, we see that for K=0, the problem is easy to solve for all values of b. Moreover, increasing K decreases the success rate and this phenomenon is amplified when high values of b are used. For N=10 and 16, too few runs find the optimum and so the variations of the success rate are not significant. The Figure 7 gives the evolution of the average number of blocks of the best individual found for N=10, b=4 and K between 0 and 5. At the beginning of the runs, the number of blocks found increases quickly then halts after several generations. The higher is K, the sooner ends evolution. This behavior looks

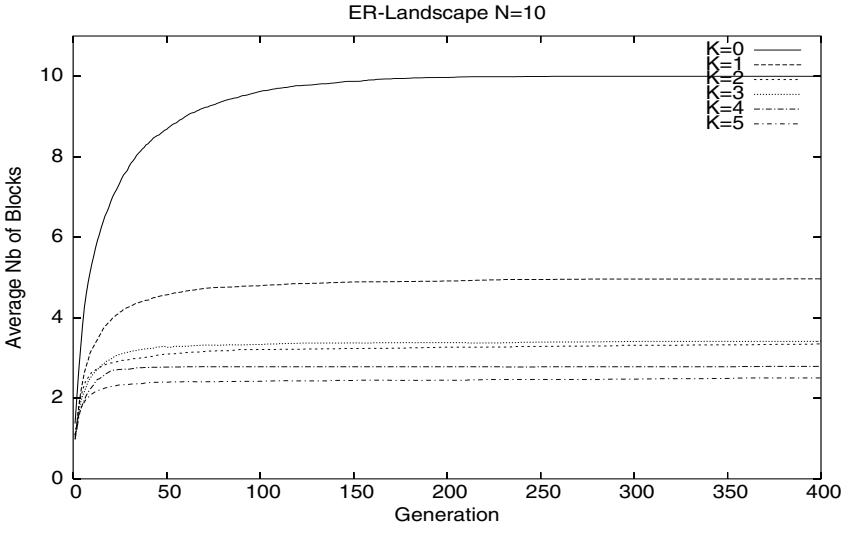


Fig. 7. Evolution of average number of blocks found on ER N=10 and b=4

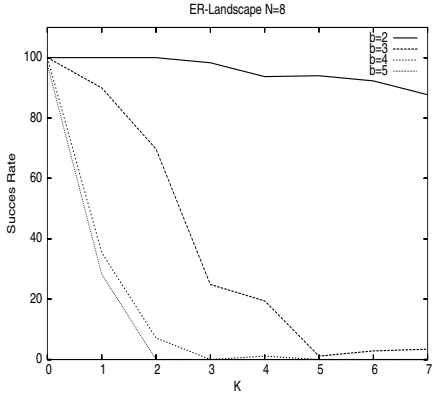


Fig. 8. Success rate as a function of K on ER N=8

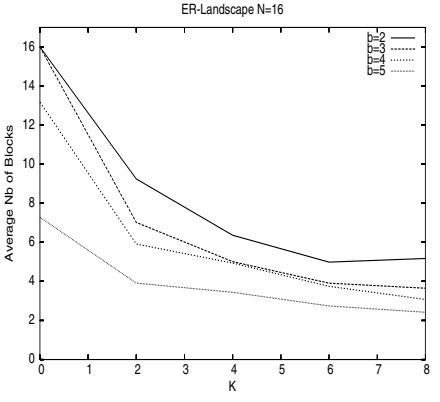


Fig. 9. Average number of blocks found as a function of K on ER N=16

like premature converge and confirms experimentally that the number of local optima increases with K. We have also plotted the average number of blocks of the best individual found as a function of K for N=16 (see Fig. 9). We see that this number decreases as K or b increases. These two parameters undoubtedly modify the performances and can be used independently to increase problem difficulty.

In [15], random and adaptive walks have been used to measure problem difficulty in GP. The author has shown that only the adaptive walk gives significant

results on classical GP benchmarks. We have computed the correlation between these two measures and the average number of blocks found on ER, for all settings of N, K and b. We note that the correlation is 0.71 between the length of the adaptive walk and the number of blocks. Conversely, the length of the random walk seems to be completely uncorrelated to performance.

Conclusion

We think that a better understanding of the implications of variable length representations on Evolutionary Algorithms would allow researchers to use these structures more efficiently. In this paper, our goal is to investigate which kind of property could influence the difficulty of such problems. We have chosen two features of search spaces, the neutrality and the ruggedness. So, we have designed a family of problems, the Epistatic Road landscapes, where those features can be tuned independently.

Statistical measures computed on ER-landscapes have shown that, similarly to NK-landscapes, tuning the epistatic coupling parameter K increases ruggedness. Moreover, as for Royal Roads functions, tuning the size block parameter b increases neutrality.

The experiments that we have performed with a VLR evolutionary algorithm, have demonstrated the expected difficulty according to parameters b and K. Although our results can not be directly transposed to real world problems, mainly because our initial hypotheses are too simple, in particular about the nature of building blocks, we have a ready-to-use VLR problem of tunable difficulty, which allows us to study the effects of genetic operators and the dynamics of the evolutionary process.

References

1. Platel, M.D., Clergue, M., Collard, P.: Maximum homologous crossover for linear genetic programming. In: Genetic Programming, Proceedings of EuroGP'2003. Volume 2610 of LNCS., Essex, UK, Springer-Verlag (2003) 194–203.
2. Banzhaf, W., Frankone, F.D., Nordin, P.: Some emergent properties of variable size EAs. Position paper at the Workshop on Evolutionary Computation with Variable Size Representation at ICGA-97 (1997).
3. Langdon, W.B., Poli, R.: Fitness causes bloat. In Chawdhry, P.K., Roy, R., Pan, R.K., eds.: Second On-line World Conference on Soft Computing in Engineering Design and Manufacturing, Springer-Verlag London (1997) 13–22.
4. Daida, J.M., Polito, J.A., Stanhope, S.A., Bertram, R.R., Khoo, J.C., Chaudhary, S.A.: What makes a problem GP-hard? analysis of a tunably difficult problem in genetic programming. 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. Volume 2., Orlando, Florida, USA, Morgan Kaufmann (1999) 982–989.
5. Punch, W.F., Zongker, D., Goodman, E.D.: The royal tree problem, a benchmark for single and multiple population genetic programming. In: Advances in Genetic Programming 2. MIT Press, Cambridge, MA, USA (1996) 299–316.

6. Forrest, S., Mitchell, M.: Relative building-block fitness and the building-block hypothesis. In: Foundation of Genetic Algorithms 2. Morgan Kaufman (1993) 109–126.
7. Clergue, M., Collard, P., Tomassini, M., Vanneschi, L.: Fitness distance correlation and problem difficulty for genetic programming. In: GECCO 2002: Proceedings of the Genetic and Evolutionary Computation Conference, New York, Morgan Kaufmann Publishers (2002) 724–732.
8. Kauffman, S.A.: "The origins of order". Self-organization and selection in evolution. Oxford University Press, New-York (1993).
9. Weinberger, E.D.: Local properties of kauffman's N-k model, a tuneably rugged energy landscape. Physical Review A **44** (1991) 6399–6413.
10. Fontana, W., Stadler, P.F., Bornberg-Bauer, E.G., Griesmacher, T., Hofacker, I.L., Tacker, M., Tarazona, P., Weinberger, E.D., Schuster, P.: RNA folding and combinatory landscapes. Physical review E **47** (1993) 2083–2099.
11. Levenshtein., V.I.: Binary codes capable of correcting deletions, insertions, and reversals. Soviet Physics-Doklady (1966).
12. Brameier, M., Bhanzhaf, W.: Explicit control of diversity and effective variation distance in linear genetic programming (2001).
13. O'Reilly, U.: Using a distance metric on genetic programs to understand genetic operators (1997).
14. Weinberger, E.D.: Correlated and uncorrelated fitness landscapes and how to tell the difference. Biological Cybernetics **63** (1990) 325–.
15. Kinnear, Jr., K.E.: Fitness landscapes and difficulty in genetic programming. In: Proceedings of the 1994 IEEE World Conference on Computational Intelligence. Volume 1., Orlando, Florida, USA, IEEE Press (1994) 142–147.

Functional Dependency and Degeneracy: Detailed Analysis of the GAuGE System

Miguel Nicolau[1], Anne Auger[2,3], and Conor Ryan[1]

[1] Department Of Computer Science And Information Systems
University of Limerick, Ireland
{Miguel.Nicolau, Conor.Ryan}@ul.ie
[2] CERMICS – ENPC
Cité Descartes, 77455 Marne-La-Vallée, France
[3] INRIA Rocquencourt, Projet Fractales
BP 105, 78153 LE CHESNAY Cedex, France
auger@cermics.enpc.fr

Abstract. This paper explores the mapping process of the GAuGE system, a recently introduced position-independent genetic algorithm, that encodes both the positions and the values of individuals at the genotypic level. A mathematical formalisation of its mapping process is presented, and is used to characterise the functional dependency feature of the system. An analysis of the effect of degeneracy in this functional dependency is then performed, and a mathematical theorem is given, showing that the introduction of degeneracy reduces the position specification bias of individuals. Experimental results are given, that backup these findings.

1 Introduction

GAuGE [16] (Genetic Algorithms using Grammatical Evolution) is a position independent Genetic Algorithm that encodes both the position and value of each variable of a given problem. It suffers from neither under nor over-specification, due to the use of a genotype-to-phenotype mapping process; that is, a position specification is mapped onto a set of available positions in each phenotype string, ensuring that a fixed-length genotype string generates a single value for each element of the phenotype string.

Two of the main biologically inspired features present in GAuGE are the functional dependency and code degeneracy at the genotypic level. This work is an investigation into both features, and their combined effect on the position specifications of the genotype strings, in the initial generation of individuals. To this end, a mathematical analysis of these features is performed, and a theorem is presented, which shows that an increase of degeneracy leads to a weaker functional dependency across the position specifications in each genotype string. These results are supported by experimental evidence.

This paper is organised as follows. Section 2 gives a brief presentation of the Grammatical Evolution system, and Section 3 presents the GAuGE system and its mapping process. Section 4 presents and analyses the experiments conducted, and Section 5 draws some conclusions and possible future work directions.

P. Liardet et al. (Eds.): EA 2003, LNCS 2936, pp. 15–26, 2004.

2 Grammatical Evolution

Grammatical Evolution (GE) [15,14,13] is an automatic programming system that uses grammars to generate code written in any language. It uses a genetic algorithm to generate variable-length binary strings, which are interpreted as a series of integer values (each value being encoded with 8 bits). These values are then used to choose productions from the rules of a given grammar, to map a start symbol onto a program (the phenotype), consisting solely of terminal symbols extracted from the grammar. This genotype-to-phenotype mapping, based on the analogous process in molecular biology, provides a division between the search space (binary strings) and the solution space (evolved programs) [2].

Another interesting feature of GE is the functional dependency across the genes on each genotype string. Specifically, the function of a gene is dependent on those genes preceding it, as previous choices of grammar productions will dictate which symbol is to be mapped next, and therefore which rule is to be applied. This creates a dependency across the genotype string, ranging from left to right, and has been termed the "ripple" effect [12].

Finally, the use of degenerate code plays an important role in GE: by using the mod operator to map an integer to the choices of a grammar rule, neutral mutations [8] can take place, as different integers can choose the same production from a given rule. This means that the genotype can be modified without changing the phenotype, allowing variety at the genotypic level.

3 GAuGE

The GAuGE system is based on most of the same biologically inspired features present in GE, and as GE, it also uses a genotype to phenotype mapping process, thus separating the search space (the genotype space) and the solution space (the phenotype space).

The main principle behind GAuGE is the separate encoding of the position and value of each variable, at the genotypic level. This allows the system to adapt its representation of the problem; experiments conducted previously [10] have shown that this feature allowed GAuGE to evolve its representation, to match the salience hierarchy of a given set of problems.

Previous work has used similar approaches and techniques as the ones employed in GAuGE. Some of Bagley's [1] computer simulations already used an extended string representation to encode both the position and the value of each allele, and used an inversion operator to affect the ordering of genes. Holland [6] later presented modifications to the schema theorem, to include the approximate effect of the inversion operator. To tackle the problems associated with the combination of the inversion and crossover operators, these were later combined into a single operation, and a series of reordering operators were created [11].

The so-called messy genetic algorithms applied the principle of separating the *gene* and *locus* specifications with considerable success ever since their introduction [4], and have since been followed by many competent GAs.

Work by Bean [3] with the Random Keys Genetic Algorithm (RKGA) hinted that a tight linkage between genes would result in both a smoother transition between parents and offspring when genetic operators are applied, and an error-free mapping to a sequence of ordinal numbers. More recently, Harik [5] has applied the principles of functional dependency in the Linkage Learning Genetic Algorithm (LLGA), in which a chromosome is expressed as a circular list of genes, with the functionality of a gene being dependent on a chosen interpretation point, and the genes between that point and itself.

3.1 Formal Definition of the Mapping Process

For a given problem composed of ℓ variables, a population $G = (G_i)_{0 \leq i \leq N-1}$, of N binary strings, is created, within the genotype space $\mathcal{G} = \{0,1\}^L$, where L will be fixed later on. Through a genotype to phenotype mapping process, these strings will be used to encode a population $P = (P_i)_{0 \leq i \leq N-1}$, of N potential solutions, from the phenotypic space $\mathcal{P}$, which can then be evaluated.

As each string G_i encodes both the position and the value of each variable of a corresponding P_i string, the length of G_i will depend on a chosen *position field size* (pfs) and *value field size* (vfs), i.e., the number of bits used to encode the position and the value of each variable in P_i. The required length of each string G_i can therefore be calculated by the formula:

$$L = (pfs + vfs) \times \ell \tag{1}$$

The mapping process consists in three steps denoted here by

$$\Phi : G \xrightarrow{\Phi_1} X \xrightarrow{\Phi_2} R \xrightarrow{\Phi_3} P\,.$$

The first mapping step (Φ_1) consists in interpreting each G_i as a sequence of *(position,value)* specification pairs, using the chosen pfs and vfs values. In this way, G_i can be used to create a string X_i, where:

$$X_i = \left((X_i^j, \tilde{X}_i^j)\right)_{0 \leq j \leq l-1} = (X_i^0, \tilde{X}_i^0), (X_i^1, \tilde{X}_i^1), \ldots, (X_i^{l-1}, \tilde{X}_i^{l-1})$$

with X_i^j being an integer random variable (encoded with pfs bits from G_i) that takes its values from $\{0, \ldots, 2^{pfs} - 1\}$, and $\tilde{X}_i^j$ an integer random variable (encoded with vfs bits from G_i) that takes its values from $\{0, \ldots, 2^{vfs} - 1\}$.

The second mapping step (Φ_2) then consists in using X_i to create a string of positions $R_i = (R_i^0, \ldots, R_i^{l-1})$, and a string of values $\tilde{R}_i = (\tilde{R}_i^0, \ldots, \tilde{R}_i^{l-1})$, where $R_i^j \in \{0, \ldots, \ell - 1\}$ is the position in the phenotype string (P_i) that will take the value specified by $\tilde{R}_i^j$.

The elements of these strings are calculated as follows. Assuming $R_i^0, \ldots, R_i^{j-1}$ are known, the set $\mathrm{A}_i^j = \{0, \ldots, \ell - 1\} \backslash \{R_i^0, \ldots, R_i^{j-1}\}$ is considered. Then the elements of the set A_i^j are ordered (from the smaller to the

greater) into a list of $\ell - j$ elements, A_i^j, and

$$R_i^j = \left(\mathrm{A}_i^j\right)_{X_i^j \bmod (l-j)} \tag{2}$$

where $(\mathrm{A}_i^j)_{X_i^j \bmod (\ell-j)}$ is the element in (A_i^j) at position $X_i^j \mod (\ell - j)$. Eq. (2) clearly shows that the random variable R_i^j depends on the previous random variables $R_i^0, \ldots, R_i^{j-1}$ and on j.

The string of values can be constructed using the following formula[1]:

$$\tilde{R}_i^j = \tilde{X}_i^j \mod range \tag{3}$$

where *range* is the value range of the variables for the given problem. For a binary problem, for example, $range = 2$.

Finally, through the third mapping step (Φ_3), the phenotype string P_i, from the phenotypic space $\mathcal{P} = \{0, \ldots, range\}^\ell$ can be calculated by using the formula:

$$P_i^{R_i^j} = \tilde{R}_i^j \,. \tag{4}$$

In other words, through a permutation defined by R_i, the elements of $\tilde{R}_i$ are placed in their final positions.

3.2 Example

As an example of the mapping process employed in GAuGE, consider the (simplified) case where one would wish to solve a binary problem of 4 variables (so $range = 2$ and $\ell = 4$). The first step would be to create a population G of size N; if the values $pfs = 6$ bits and $vfs = 2$ bit were chosen, then, according to Eq. (1), $L = (6 + 2) \times 4 = 32$, that is, each member of G is a binary string of 32 bits.

Here is how the mapping process translates each of those strings onto a phenotype string P_i. Consider the following individual:

$$G_i = 000110\,0\,1000010\,10\,000100\,11\,000101\,01\,.$$

The first step is to create the X_i string, by using the chosen pfs and vfs values:

$$X_i = \big((6,1),(2,2),(4,3),(5,1)\big)\,.$$

From this string, the positions string R_i and the values string $\tilde{R}_i$ can be created. To create R_i, Eq. (2) is used; for the first element R_i^0, this gives us:

$$\mathrm{A}_i^0 = \{0,1,2,3\}, \qquad R_i^0 = \left(\mathrm{A}_i^0\right)_{6 \bmod 4} = \left(\mathrm{A}_i^0\right)_2 = 2\,.$$

[1] This isn't always the case. Previous variations of the GAuGE system [9] have used a different mapping process at this stage.

For the second element, R_i^1,the set A_i^1 is $\{0,1,3\}$, as the element $R_i^0 = 2$ was removed from it. Therefore the second element of R_i becomes:

$$\mathrm{A}_i^1 = \{0,1,3\}, \qquad R_i^1 = \left(\mathrm{A}_i^1\right)_{2 \bmod 3} = \left(\mathrm{A}_i^1\right)_2 = 3\,.$$

The third element is calculated in the same fashion:

$$\mathrm{A}_i^2 = \{0,1\}, \qquad R_i^2 = \left(\mathrm{A}_i^2\right)_{4 \bmod 2} = \left(\mathrm{A}_i^2\right)_0 = 0\,.$$

And finally the fourth element:

$$\mathrm{A}_i^3 = \{1\}, \qquad R_i^3 = \left(\mathrm{A}_i^3\right)_{5 \bmod 1} = \left(\mathrm{A}_i^3\right)_0 = 1\,.$$

So the string of positions becomes:

$$R_i = (2,3,0,1)\,.$$

The string of values $\tilde{R}_i$ can then be calculated using Eq. (3):

$$\begin{aligned}
\tilde{R}_i^0 &= \tilde{X}_i^0 \mod 2 = 1 \mod 2 = 1,\\
\tilde{R}_i^1 &= \tilde{X}_i^1 \mod 2 = 2 \mod 2 = 0,\\
\tilde{R}_i^2 &= \tilde{X}_i^2 \mod 2 = 3 \mod 2 = 1,\\
\tilde{R}_i^3 &= \tilde{X}_i^3 \mod 2 = 1 \mod 2 = 1\,.
\end{aligned}$$

Finally, by using Eq. (4), the phenotype string P_i can be constructed in the following manner:

$$P_i^2 = 1, \qquad P_i^3 = 0, \qquad P_i^0 = 1, \qquad P_i^1 = 1$$

So the phenotype string P_i, ready for evaluation, is:

$$P_i = 1110\,.$$

4 Functional Dependency and Degenerate Code

In this section, the functional dependency occurring within the elements of the genotype string is analysed, along with the effect of degenerate code on that dependency. By functional dependency, the manner in which the function of a gene is (possibly) affected by the function of previous genes is meant. As shown in Eq. 2, each position specification derived from the genotype string depends on previous position specifications, and on its location within the genotype string. The degenerate code feature refers to the many-to-one mapping process from genotype to phenotype. As seen in Section 3.1, the use of the mod function maps the value specified by X_i^j onto the set of available positions left on the phenotype string.

In order to characterise the dependency of each R_i^j on the elements $R_i^0 \ldots R_i^{j-1}$ and on j itself, the mean value of each R^j is computed. If this dependency did not exist, then this mean value would be constant in j. Section 4.1 shows that this mean value depends on j, and Section 4.2 shows the effect of the introduction of degeneracy on this mean value.

4.1 Computing the Mean of R^j

For every $j \in \{0, \ldots, \ell - 1\}$, the mean (or expectation) of R^j is denoted $E(R^j)$. This expectation can be computed exactly in $\ell!$ operations (see Remark 1). It is thus crucial to use another method to compute the expectation $E(R^j)$ of all R_i^j. The method chosen here is the numerical simulation of the expectation with the Monte Carlo method [7], whose main interest is to give a confidence interval for the approximation of the mean.

The principle of the Monte Carlo simulation is as follows: in order to calculate the average position $E(R^j)$ of all R_i^j, a population of N individuals (with N large, typically equal to 10000) is generated. The N individuals are denoted $(R_i^j)_{1 \leq i \leq N}$. Each mean of R^j, $E(R^j)$ is approximated by

$$\frac{1}{N} \sum_{i=1}^{N} R_i^j \ . \tag{5}$$

A consequence of the Central Limit Theorem is that there is a confidence interval for this approximation. More precisely, for the 99% interval confidence, there is a probability equal to 0.99 that the true values of each average position $E(R^j)$ are within the interval

$$\left[\frac{1}{N} \sum_{i=1}^{N} R_i^j - \alpha_i(N), \frac{1}{N} \sum_{i=1}^{N} R_i^j + \alpha_i(N) \right]$$

where $\alpha_i(N)$ is equal to

$$2.58 \frac{\sqrt{\frac{1}{N} \sum_{i=1}^{N} (R_i^j)^2 - \left(\frac{1}{N} \sum_{i=1}^{N} R_i^j \right)^2}}{\sqrt{N}} \ . \tag{6}$$

The values used in the simulations are given in Table 1. Figure 1 shows a plot of those averages, along with the 99% interval.

Table 1. Experimental setup. 7 bits were used for the position field, as this is the minimum number of bits required to encode 128 positions

Problem length (l):	128
Population size (N):	100000
Position field size (pfs):	7 bits
Value field size (vfs):	1 bit

Experimental results. The graph illustrates the functional dependency across the elements of all the strings of positions R, from the left to the right side. As each number specified by X_i^j is considered modulo smaller values (128, 127, ...),

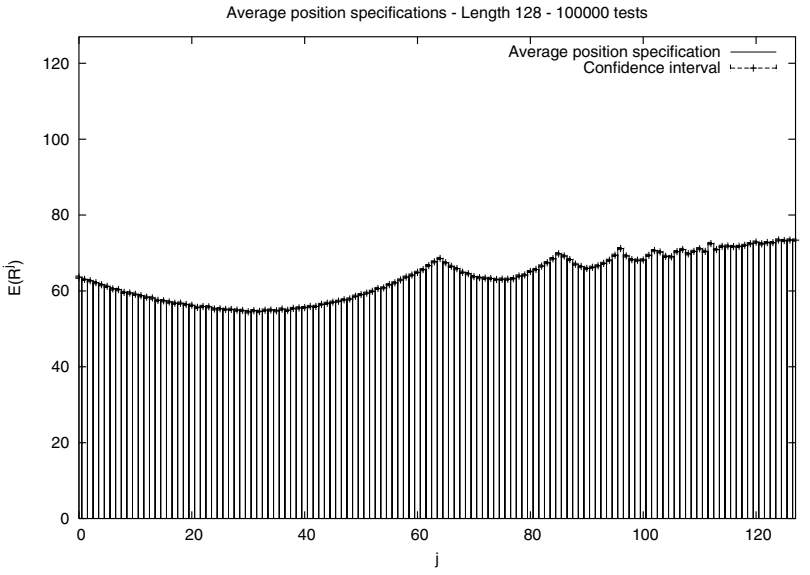


Fig. 1. Average position specification per element of X_i, for a genotype generating a phenotype of length 128. The *x-axis* shows each position (index j) of the string X_i, and the *y-axis* the average position it specifies, $E(R_i^j)$ (results averaged over 100000 randomly created individuals)

the index it specifies in the set $\mathrm{A_i^j}$ is wrapped around to the beginning of that set (due to the use of the mod operator), which means that smaller values will be specified on average. However, as the number to mod X_i^j by reaches the value 64, the distribution of values specified is again uniform, which explains the return of the average position specified to a more average value again (close to 63.5).

This balancing phenomenon continues on, creating a *ripple effect* [12] across the positions specified in each R_i. The slight upward slope in the positions specified is explained by the fact that between the points where the positions return to *close to* average values (0,64,85,96,102,106, ... [2]), the positions specified are smaller than the average.

4.2 Introducing Degeneracy

In this section, the effect of the introduction of degeneracy at the genotype level is investigated. Degeneracy in the position specifications is introduced by using bigger values for pfs; indeed, this introduces a redundant mapping, as different values for X_i^j will map to the same R_i^j value (through the use of the mod operator). This will have the effect of balancing the position specifications

[2] This sequence of *peaks* represents the points where the mod operator wraps around (e.g., 128 mod 64 = 0, and 128 mod 43 = 42, but 128 (mod 42) = 2 (128 − 43 = 85))

across each R_i string, as each element of that string will be averaged across a larger set of values, thus *flattening* the ripples observed.

A theoretical proof of the fact that introduction of degeneracy flattens the ripples, and that, consequently, the position each R^j specifies becomes less dependent from j, can be given. Specifically we shall prove in Appendix 5 the following Theorem

Theorem 1. *Let i be an integer in $\{0, ..., N-1\}$ and let $R_i = (R_i^0, ..., R_i^{\ell-1})$ defined in Section 3 by Eq. (2), then for all $j \in \{0, \ldots, \ell-1\}$*

$$E(R_i^j) = \frac{\ell-1}{2} + \mathcal{O}(\ell^3 2^{-pfs}) \,. \tag{7}$$

where the notation $\mathcal{O}(\ell^3 2^{-pfs})$ means that $\frac{\mathcal{O}(\ell^3 2^{-pfs})}{\ell^3 2^{-pfs}}$ is bounded when $pfs \to \infty$ by a constant independent of ℓ. A consequence of Eq. (7) is thus

$$E(R_i^j) \xrightarrow[pfs \to \infty]{} \frac{\ell-1}{2} \text{ for all } j \,,$$

meaning that the expectation $E(R_i^j)$ becomes constant and thus independent of the index j when *pfs* is high. Moreover $\mathcal{O}(\ell^3 2^{-pfs})$ gives a quantitative idea of the value of *pfs* in order to consider that $E(R_i^j)$ can be approximated by $\frac{\ell-1}{2}$: this is precisely when $\frac{2^{pfs}}{\ell^3} \gg 1$.

To illustrate these findings, a similar experimental setup to that used in Section 4.1 is used; in these experiments, the value of pfs varied from 7 bits to 14 bits. Figure 2 plots the results of the simulation.

Experimental results. The introduction of degeneracy has an interesting effect. By using a larger number of bits for pfs, the range of numbers each X_i^j can specify is increased (e.g., using 14 bits, the range will be 0-16383, rather than 0-127 with 7 bits per gene); when modded by the number of available positions (128, 127, ...), this reduces significantly the differences between the position specified by each field, thus effectively *flattening* the ripples. Figure 2 illustrates this effect: by gradually increasing the value of pfs, the differences in the average of each R_i^j are reduced, as is the slight increasing average position specified toward the end of each R_i string.

5 Conclusions and Future Work

This paper has presented a detailed investigation of the mapping process employed in the GAuGE system. A formal definition of that process has been given, and is used to characterise the functional dependency occurring between the position specifications of the genotype string, and the effect that the introduction of degeneracy has on that dependency. The theorem presented, backed up by experimental evidence, shows that the introduction of degeneracy reduces the functional dependency between the position specifications and their placement

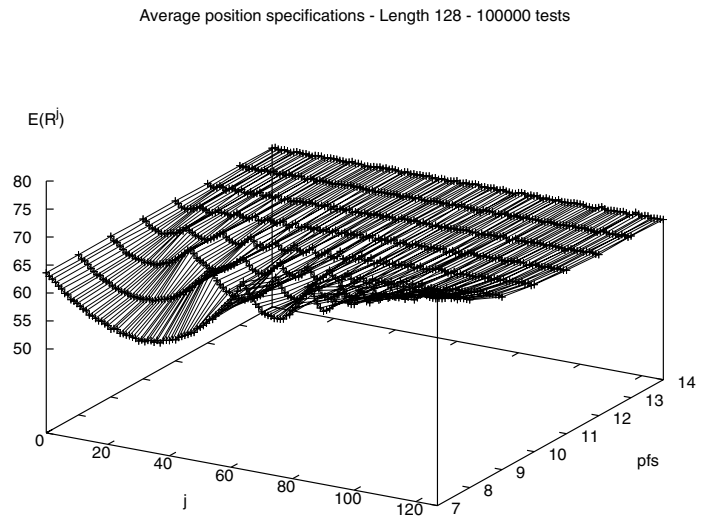


Fig. 2. Effect of degeneracy on the average position specification per element of X_i. The *x-axis* shows each position (j) of the string X_i, the *y-axis* (depth) shows the pfs value chosen, and the *z-axis* (vertical) shows the average position specified, $E(R_i^j)$ (results averaged over 100000 randomly created individuals)

on the real specifications string, and illustrates some of the implications of using degenerate code at the genotype level.

The models presented can be of use when analysing alternative mapping processes, not only for the GAuGE system, but also for other systems where a detailed analysis of functional dependency should be required.

As can be expected, when the evolutionary process starts, the distribution of position specifications across the genotype is no longer uniform. Future work will therefore analyse the implications that the results reported have during the evolution process.

Appendix: Proof of Theorem 1

Before giving the proof of Theorem 1, an alternative view of the mapping described in Section 3.1 is proposed. The mapping is seen as a permutation of the set of values $\{\tilde{R}_i^0, \ldots, \tilde{R}_i^{\ell-1}\}$. Indeed, the string of positions $R_i = (R_i^0, \ldots, R_i^{\ell-1})$ where for all $j \in \{0, \ldots, \ell-1\}$, $R_i^j \in \{0, \ldots, \ell-1\}$ is nothing but a classical manner to write a permutation of ℓ elements, *i.e.* a bijective function of a set of ℓ elements into itself. Formally the permutation $\rho : \{0, \ldots, \ell-1\} \to \{0, \ldots, \ell-1\}$ induced by the mapping is defined as follows:

$$\rho(j) = R_i^j \text{ for } 0 \leq j \leq \ell - 1\,.$$

And the mapping consists then in permuting the elements $\{\tilde{R}_i^0, \ldots, \tilde{R}_i^{\ell-1}\}$ into $\{\tilde{R}_i^{\rho(0)}, \ldots, \tilde{R}_i^{\rho(\ell-1)}\}$. This alternative view is used in the proof of the following Theorem: denoted $\mathcal{S}_l$.

Theorem 1. *Let i be an integer in $\{0, ..., N-1\}$ and let $R_i = (R_i^0, ..., R_i^{\ell-1})$ defined in Section 3 by Eq. (2), then for all $j \in \{0, \ldots, \ell-1\}$,*

$$E(R_i^j) = \frac{\ell - 1}{2} + \mathcal{O}(\ell^3 2^{-pfs})\,.$$

Proof. Considering each R_i^j defined by Eq. 2 as a random variable, one sees that the distributions of R_i^j are the same for any i. Thus, for sake of simplicity, the subscript i is omitted in this proof. For consistency with the formalisation of the mapping in terms of permutation, where $R = (R^0, \ldots, R^{\ell-1})$ is identified with a permutation, for any given permutation $\rho \in \mathcal{S}_l$ ($\mathcal{S}_\ell$ being the permutation group on $\{0, \ldots, \ell-1\}$) the notation $R = \rho$ denotes in the sequel the event:

$$R^j = \rho(j) \text{ for all } j\,.$$

From the construction of R^j one sees that, for an acceptable value a (among the $\ell - j$ possible values) for R^j, one has

$$P(R^j = a | R^{j-1}, \ldots, R^0) = 2^{-pfs}\big(\lfloor \frac{2^{pfs}}{\ell - j} \rfloor + \mathbb{1}_{\{a < 2^{pfs} \pmod{\ell} - j\}}\big) \tag{8}$$

where $\lfloor \frac{2^{pfs}}{\ell - j} \rfloor$ denotes the integer part of $\frac{2^{pfs}}{l-j}$ and $\mathbb{1}_{\{a<2^{pfs} \pmod{\ell} - j\}} = 1$ if $a < 2^{pfs} \pmod{\ell} - j$ and 0 otherwise. Simplifying Eq. (8), one has,

$$P(R^j = a | R^{j-1}, \ldots, R^0) = \frac{1}{\ell - j} + r_{j,pfs}(a)$$

with $|r_{j,pfs}(a)| \leq 2^{-pfs}$. Therefore, for a given permutation ρ, $P(R = \rho)$ is the product of the above probabilities and a straightforward computation leads to

$$P(R = \rho) = \frac{1}{\ell!}(1 + \mathcal{O}(\ell^2 2^{-pfs})).$$

where the notation $\mathcal{O}(\ell^2 2^{-pfs})$ means that $\frac{\mathcal{O}(\ell^2 2^{-pfs})}{\ell^2 2^{-pfs}}$ is bounded when $pfs \to \infty$ by a constant independent of ℓ. Now the expectation of the random variable R^j is

$$E(R^j) = \sum_{0 \leq a < \ell} a \sum_{\rho \in \mathcal{S}_\ell} P(R = \rho \;\&\; \rho(j) = a) \tag{9}$$

but the number of permutations ρ such that $\rho(j)$ is fixed and is equal to $(\ell-1)!$, hence

$$E(R^j) = (\ell - 1)! \frac{1}{\ell!}(1 + \mathcal{O}(\ell^2 2^{-pfs}))(0 + 1 + \cdots + (\ell - 1)).$$

Finally

$$E(R^j) = \frac{\ell - 1}{2}(1 + \mathcal{O}(\ell^2 2^{-pfs}))$$

or

$$E(R^j) = \frac{\ell - 1}{2} + \mathcal{O}(\ell^3 2^{-pfs}).$$

□

Remark 1 (Complexity for the computation of $E(R^j)$). As one can see on Eq. (9), the complexity to compute $E(R^j)$ is $\ell!$.

Acknowledgments. The authors would like to thank all anonymous reviewers, whose suggestions led to several improvements of this work; in particular, the reviewer whose clever interpretation of the mapping in term of permutations led to a more precise estimation in the theorem given and to simplifications in the proof. We would also like to thank the feedback provided by Zbigniew Skolicki on the explanation of the *ripple* effect.

References

1. Bagley, J. D.: The behaviour of adaptive systems which employ genetic and correlation algorithms. Doctoral Dissertation, University of Michigan (1967).
2. Banzhaf, W.: Genotype-Phenotype-Mapping and Neutral Variation - A case study in Genetic Programming. In: Davidor et al., (eds.): Proceedings of the third conference on Parallel Problem Solving from Nature. Lecture Notes in Computer Science, Vol. 866. Springer-Verlag (1994), 322-332.
3. Bean, J.: Genetic Algorithms and Random Keys for Sequencing and Optimization. ORSA Journal on Computing, Vol. **6**, No. 2 (1994), 154-160.
4. Goldberg, D. E., Korb, B., and Deb, K.: Messy genetic algorithms: Motivation, analysis, and first results. Complex Systems, Vol. **3** (1989), 493-530.
5. Harik, G.: Learning Gene Linkage to Efficiently Solve Problems of Bounded Difficulty Using Genetic Algorithms. Doctoral Dissertation, University of Illinois (1997).
6. Holland, J. H.: Adaptation in Natural and Artificial Systems. University of Michigan Press. (1975).
7. Kalos, M. H., and Withlock, P. A.: Monte Carlo Methods, Vol 1, Wiley (1986).
8. Kimura, M.: The Neutral Theory of Molecular Evolution. Cambridge University Press (1983).
9. Nicolau, M., and Ryan, C.: LINKGAUGE: Tackling hard deceptive problems with a new linkage learning genetic algorithm. In: Langdon et al., (eds.): Proceedings of the Genetic and Evolutionary Computation Conference GECCO-2002. Morgan Kaufmann Publishers, San Francisco (2002), 488-494.
10. Nicolau, M., and Ryan, C.: How Functional Dependency Adapts to Salience Hierarchy in the GAuGE System. In: Ryan et al, (eds.): Proceedings of EuroGP-2003. Lecture Notes in Computer Science, Vol. 2610, Springer-Verlag (2003), 153-163.
11. Oliver, I. M., Smith, D. J., and Holland, J. R. C.: A Study of Permutation Crossover Operators on the Traveling Salesman Problem. In: Proceedings of the Second International Conference on Genetic Algorithms (1987), 224-230.

12. O'Neill, M., Ryan, C., Keijzer, M., and Cattolico, M.: Crossover in Grammatical Evolution. Genetic Programming and Evolvable Machines, Vol. **4**, No. 1 (2003), 67-93.
13. O'Neill, M. and Ryan, C.: Grammatical Evolution - Evolving programs in an arbitrary language. Kluwer Academic Publishers (2003).
14. O'Neill, M., and Ryan, C.: Grammatical Evolution. IEEE Transactions on Evolutionary Computation, Vol. **5**, No. 4, (2001), 349-358.
15. Ryan, C., Collins, J. J., and O'Neill, M.: Grammatical Evolution: Evolving Programs for an Arbitrary Language. In: Banzhaf et al., (eds.): Proceedings of the First European Workshop on Genetic Programming, EuroGP'98. Lecture Notes in Computer Science, Vol. 1391. Springer-Verlag (1998), 83-95.
16. Ryan, C., Nicolau, M., and O'Neill, M.: Genetic Algorithms using Grammatical Evolution. In: Foster et al, (eds.): Proceedings of EuroGP-2002. Lecture Notes in Computer Science, Vol. 2278, Springer-Verlag (2002), 278-287.

A Study of the Effects of Dimensionality on Stochastic Hill Climbers and Estimation of Distribution Algorithms

Laurent Grosset[1,2], Rodolphe Le Riche[2], and Raphael T. Haftka[1]

[1] Mechanical and Aerospace Engineering Department, University of Florida, USA
[2] CNRS URA 1884 / SMS, École des Mines de Saint Étienne, France

Abstract. One of the most important features of an optimization method is its response to an increase in the number of variables, n. Random stochastic hill climber (SHC) and univariate marginal distribution algorithms (UMDA) are two fundamentally different stochastic optimizers. SHC proceeds with local perturbations while UMDA infers and uses a global probability density. The response to dimensionality of the two methods is compared both numerically and theoretically on unimodal functions. SHC response is $\mathcal{O}(n \ln n)$, while UMDA response ranges from $\mathcal{O}(\sqrt(n) \ln(n))$ to $\mathcal{O}(n \ln(n))$. On two test problems whose sizes go up to 7^{200}, SHC is faster than UMDA.

1 Introduction

Random stochastic hill climber (SHC) and univariate marginal distribution algorithms (UMDA, Mühlenbein and Paaß (1996)) are two fundamentally different stochastic optimizers. SHC proceeds with local perturbations while UMDA infers and uses a global probability density. It is important to understand how these two different search processes scale with the number of variables n.

Previous contributions have analyzed stochastic hill climbers and population-based evolutionary algorithms on specific objective functions. Garnier et al. (1999) computed the expected first hitting time and its variance for two variants of stochastic hill-climbers. Droste et al. (2002) extended the estimation of the running time of $\mathcal{O}(n \ln n)$ for a (1+1)-evolution strategy (ES) to general linear functions. Several studies have investigated the benefits of a population. For example, SHC and genetic algorithms have been compared in Mitchell et al. (1994) on the Royal Road function. Jansen and Wegener (2001) presented a family of functions for which it can be proven that populations accelerate convergence of an evolutionary algorithm (EA) to the optimum even without crossover. He and Yao (2002) used a Markov chain analysis to estimate the time complexity of EAs for various problems and showed that a population is beneficial for some multimodal problems. Comparisons between single point and population-based evolution strategies on Long-Path problems are given in Garnier and Kallel (2000). There has been little work on the time complexity of estimation of distribution algorithms. The convergence of estimation of distribution algorithms has been

P. Liardet et al. (Eds.): EA 2003, LNCS 2936, pp. 27–38, 2004.

studied in Mühlenbein et al. (1999). Experimental comparisons were conducted by Pelikan et al. (2000). However, there is a lack of theoretical results on the relative performances of SHC and UMDA.

The present paper is a numerical and analytical investigation of the effect of the number of non-binary variables on the performance of SHC and UMDA for two separable, unimodal functions. The topology of the functions is purposely simple so that the effect of dimensionality can be isolated. The performance of the methods for a multimodal, separable function is also discussed.

2 Presentation of the Algorithms

We consider the problem of maximizing some fitness function F over a design space $D = A^n$, where A is an alphabet of cardinality c. SHC searches the space by choosing an initial point $x = (x_1, x_2, \dots, x_n)$ at random and applying random perturbations to it: the algorithm changes the value of one variable chosen at random to an adjacent value and accepts the new point only if it improves the fitness function. Perturbations are applied until no more progress is observed.

UMDA is a simple form of estimation of distribution algorithms (EDA). EDAs use populations of m points to infer the distribution $p(x)$ of good points. By a succession of sampling and selection steps, the distribution converges toward regions of increasing fitness evaluation, eventually yielding the optimum. In UMDA, distributions are expressed as products of univariate marginal distributions, leading to the following update rule for the distributions:

$$p(x, t+1) = \prod_{i=1}^{n} p^s(x_i, t) \ , \tag{1}$$

where $p(x, t+1)$ refers to the search distribution at time $t+1$ and $p^s(x_i, t)$ designates the univariate distribution of the variable x_i in selected points at time t. In this work, truncation selection of ratio τ was used.

The algorithm can be summarized as follows:

1. set $t = 0$,
2. initialize the search distribution $p(x, 0)$,
3. create m points by sampling from $p(x, t)$,
4. select the τm best individuals based on the fitness function,
5. estimate the univariate marginal distributions $p^s(x_i, t)$, and calculate $p(x, t+1)$ from (1),
6. go to 3.

3 Numerical Experiments

Three test problems are considered. The first two problems, "Max A_{11}" and "Vibration", are separable and unimodal functions. The third, "Min A_{66}", is a multimodal and separable function. All three problems have a physical meaning in terms of composite laminate design.

A composite laminate is made of layers of fiber reinforced material (plies), and its response is determined by many factors including the number of plies, the fiber volume fraction, the fiber and matrix properties, etc. In this work, we consider only laminates that have a fixed number of plies n, as shown in Fig. 1. In each ply, the fibers are aligned in the same direction. The goal of laminate optimization is to determine the fiber angles x_1 to x_n that maximize some objective function. Even though fiber angles can take any value between $0°$ and $90°$, the set of acceptable values is typically limited to a small number of discrete values for manufacturing requirements. In the problems considered in this paper, the angles can take seven values from $0°$ to $90°$ in $15°$ steps. Note that the angles values are ordered, i.e. they are not symbolic, hence the notion of distance underlying SHC is well defined.

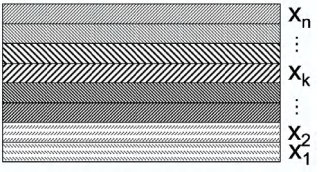


Fig. 1. Laminate cross-section: in the k^{th} layer, the fibers have an angle x_k w.r.t. a reference direction

3.1 Max A_{11} Problem

The first problem consisted in maximizing the longitudinal in-plane stiffness A_{11} of a symmetric balanced laminate[1], which is expressed by:

$$A_{11} = U_1\, h + 4U_2 \sum_{k=1}^{n} t_k \cos 2x_k + 4U_3 \sum_{k=1}^{n} t_k \cos 4x_k \,, \tag{2}$$

where U_1, U_2 and U_3 are material constants, n is the total number of plies, t_k the thickness of the k^{th} ply, and h the total thickness of the laminate.

The fitness function of this problem is a sum of functions of one variable only, so that they do not interact with one another, and UMDA is expected to converge to the global optimum $x_i^* = 0°$, $i = 1, n$. The function A_{11} is unimodal, so that SHC will also yield x^*.

The algorithms were applied to the problem for five different numbers of variables $n = 12, 20, 50, 100, 200$. The selection ratio τ of the UMDA was kept constant at $\tau = 0.3$ (as recommended in Mühlenbein et al. (1999)). Several

[1] A laminate is symmetric if the ply orientations are symmetric about its mid-plane. It is balanced when for each $+\theta$ ply, there is a $-\theta$ in the laminate. Therefore, a balanced and symmetric laminate has n unknown ply orientations and $4n$ constitutive plies.

population sizes (50, 100, 500 and 1,000) were tried in order to obtain an efficient scheme and allow a fair comparison with SHC.

Two criteria were used to compare the algorithms' performance: the number of analyses required to reach 80% reliability (defined as the probability of finding the optimum, estimated over 50 independent runs), and the number of analyses needed until the average best fitness reaches 98% of the optimal fitness. Clearly, these two criteria provide only a partial picture of the algorithms' relative performances, and other criteria may be used. However they measure two quantities that are critical in optimization algorithms: the maximum fitness criterion is an indication of the velocity of the convergence in the vicinity of the optimum, while the reliability criterion measures the algorithm effectiveness at finding the true optimum.

Figure 2 presents the number of evaluations to 98% of the maximum fitness against the number of variables for SHC and four different population sizes of UMDA. Clearly, SHC converges faster than UMDA for all the numbers of variables investigated. The evolution of the cost for SHC appears to be linear[2], which confirms the results reported in Sect. 4 and in Pelikan et al. (2000) for the *Onemax* problem. For UMDA with a given population size, the number of evaluations needed to reach 98% of the optimal fitness increases sub-linearly. Larger populations are more expensive, but smaller population can fail to converge for large n. This is the case when a population of 50 individuals is used to solve the Max A_{11} problem and $n \geq 50$, where the average maximum fitness never reaches 98% of the maximum fitness. This can be explained by the fact that when smaller populations are used, the chance of losing particular values of the variables is higher, which prevents the algorithm form finding the optimum, as will be discussed in Sect. 4.2.

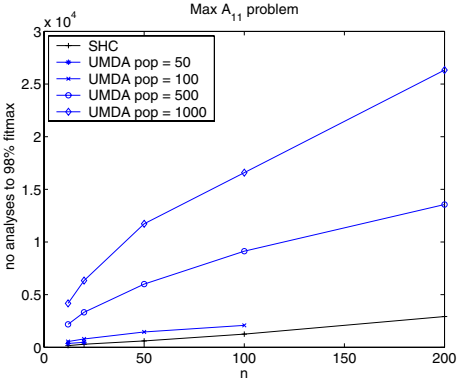


Fig. 2. Number of analyses until the average maximum fitness reaches 98% of the optimal fitness, Max A_{11} problem

[2] We will see that it is in fact $\mathcal{O}(n \ln n)$.

The effect of the loss of variable values for small populations is visible in the reliability: for each problem size n, there exists a minimum population size below which 80% reliability is never reached because of premature convergence of the distributions. This minimum population size was $m^* = 100$ for $n = 12$, $m^* = 500$ for $n = 20$, 50 and 100, $m^* = 1,000$ for $n = 200$.

In the Max A_{11} problem, all the variables are interchangeable, as the order of the variables does not affect the fitness function. Consequently, the expected value of the univariate distribution of all the variables is the same. Figure 3 shows the evolution of the probability distributions of the variable corresponding to the innermost ply for the case $n = 100$, $m = 500$. Starting from a uniform distribution over the seven values, the optimum value ($p(x_1 = 0°, t) = 1$, $p(x_1 = i, t) = 0$, $i \neq 0°$) gradually takes over the whole distribution.

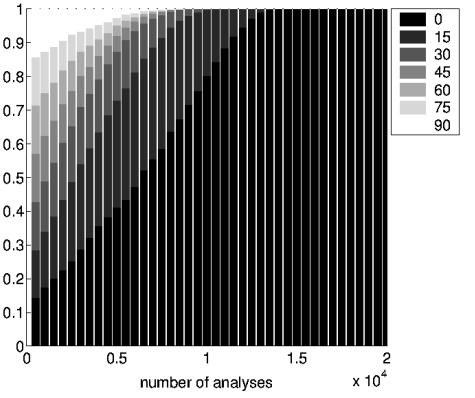


Fig. 3. Evolution of the probability distribution for the innermost ply

3.2 Vibration Problem

The second problem consisted in maximizing the first natural frequency of a simply supported rectangular laminated plate. This problem has independent variables, but exhibits a hierarchical structure in which the contribution of each variable to the fitness function depends on its position.

The first natural frequency of a simply supported rectangular plate is proportional to the square root of the following expression:

$$F = \frac{D_{11}}{L^4} + \frac{2(D_{12} + 2D_{66})}{L^2 W^2} + \frac{D_{22}}{W^4}\,, \tag{3}$$

where L and W are the length and width of the plate.

The bending stiffness coefficients D_{ij} are obtained by:

$$D_{11} = U_1 \frac{h^3}{12} + \frac{4}{3} U_2 \sum_{k=1}^{n} t_k (3z_k^2 - t_k^2) \cos 2x_k + \frac{4}{3} U_3 \sum_{k=1}^{n} t_k (3z_k^2 - t_k^2) \cos 4x_k \tag{4}$$

$$D_{22} = U_1 \frac{h^3}{12} - \frac{4}{3} U_2 \sum_{k=1}^{n} t_k (3z_k^2 - t_k^2) \cos 2x_k + \frac{4}{3} U_3 \sum_{k=1}^{n} t_k (3z_k^2 - t_k^2) \cos 4x_k \quad (5)$$

$$D_{66} = U_5 \frac{h^3}{12} - \frac{4}{3} U_3 \sum_{k=1}^{n} t_k (3z_k^2 - t_k^2) \cos 4x_k \quad (6)$$

$$D_{12} = U_4 \frac{h^3}{12} - \frac{4}{3} U_3 \sum_{k=1}^{n} t_k (3z_k^2 - t_k^2) \cos 4x_k \quad (7)$$

where the U terms are material constants and z_k refers to the position of the k^{th} ply in the laminate.

The effect of z is to give a hierarchical structure to the problem because plies located in outer layer have more weight than those located in inner layers. As a result, the distributions corresponding to the outermost plies are expected to converge faster than those corresponding to inner plies. The optimum laminate for this problem was $x_i^* = 60°$, $i = 1, n$.

The two algorithms were applied to the vibration problem for the five problem sizes n used in the previous section. The number of evaluations necessary for the average maximum fitness function to reach 98% of the optimal fitness is shown in Fig. 4. The results are similar to those obtained for the Max A_{11} problem. The cost of SHC still appears linear in the number of variables. However, the number of evaluations needed by the two algorithms to reach 98% of the maximum fitness is smaller than for the Max A_{11} problem. For instance, for $n = 12$, SHC needs 64 evaluations on the vibration problem, against 160 for the Max A_{11} problem. In the case $n = 200$, it requires 1,402 analyses for the vibration problem, against 2,923 for the Max A_{11} problem. Similarly, the number of analyses needed by UMDA with a population of 1,000 individuals decreases from 4,175 analyses to 2,028 analyses for $n = 12$ and from 26,322 analyses to 17,531 analyses for $n = 200$. The faster convergence toward high fitness regions for the second problem can be explained by the fact that a large part of the response is governed by outermost plies, so that most of the fitness improvement can be achieved by determining the value of these influential plies. In addition, the fact that the optimum angle (60°) is close to the center of the domain helps SHC by reducing the average number of steps it has to take.

If the hierarchical structure of the problem allows a rapid convergence to high fitness regions, it also causes numerical difficulties for UMDA. The convergence of the probability distribution for the innermost and outermost plies for the case $n = 100$, $m = 500$ are presented in Figs. 5 and 6, respectively. On this hierarchical problem, it appears very clearly that the algorithm proceeds from the outside to the inside, starting with the more influential variables, and determining the inner variables only at the very end. This mechanism is responsible for the loss of variable values for the less influential inner plies: in the early stages of the search, the selection of good individuals is mainly determined by the outermost variables. As a result, individuals which contain the optimum value of the outermost plies but not of the innermost plies get selected, potentially leading to the disappearance of these values in the distribution if too small a population

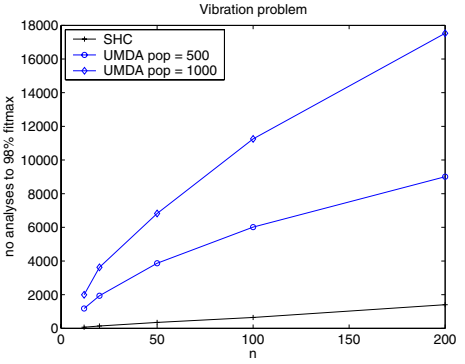


Fig. 4. Number of analyses until the average maximum fitness reaches 98% of the optimal fitness, vibration problem

is used. In order to prevent the loss of values, we observed that larger populations have to be used. The minimum population sizes for this problem were $m^* = 500$ for $n = 12$ and $n = 20$, $m^* = 1000$ for $n = 50$. In the cases $n = 100$ and $n = 200$, the algorithm did not reach 80% reliability for the population sizes tested within the maximum number of 40,000 analyses used in this work.

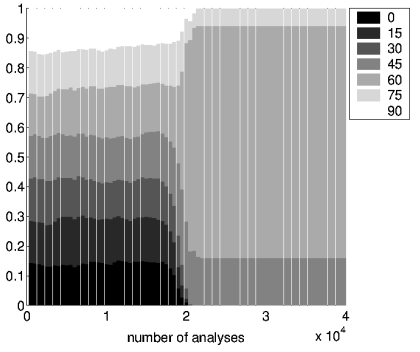


Fig. 5. Evolution of the probability distribution for the innermost ply

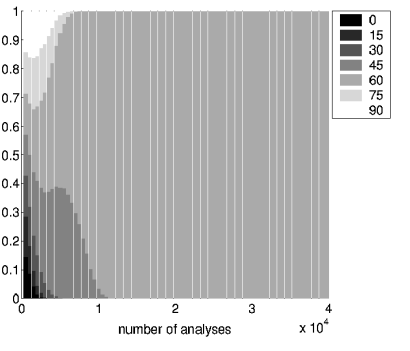


Fig. 6. Evolution of the probability distribution for the outermost ply

SHC, however, is unaffected by the problem's hierarchical structure because it merely compares two neighboring points. Fig. 7 shows the two performance measures on the two problems. Both the number of evaluations to 98% of the maximum fitness and the number of evaluations until 80% reliability is achieved are lower for the vibration problem than for the Max A_{11} problem.

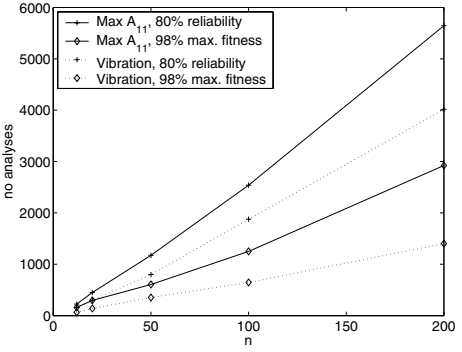


Fig. 7. Comparison of the performance of SHC on the Max A_{11} problem and the vibration problem

3.3 A Multimodal, Separable Function

SHC is misled by local minima, while the relationship between UMDA and the objective function is more complex. It is nevertheless known that, if the objective function is separable[3] and the population size is larger than m^*, UMDA converges to the global optima. An example of such an objective function where the reliability of UMDA tends to 1 while that of SHC is nearly 0 is the minimization of the in-plane shear stiffness of a composite laminate, A_{66}, over ply orientations that are bound between 0° and 75°,

$$\min_{0^\circ \leq x_i \leq 75^\circ} A_{66}\,, \tag{8}$$

where

$$A_{66} = U_5 h - U_3 \sum_{k=1}^{n} t_k \cos 4x_k\,. \tag{9}$$

The global optimum is $x_i^* = 0^\circ$, $i = 1, n$. Replacing any of the x_i^* with 75° creates a local optimum whose basin of attraction starts at $x = 45^\circ$. For an n-dimensional case, there are 2^n local optima. Numerical experiments with $n = 12$ averaged over 50 runs confirm that the SHC reliability is 0 (it is theoretically $(45/75)^{12} = 2.10^{-3}$) while that of an UMDA with $m = 500$ reaches 1 after 3,500 analyses. Fig. 8 shows the evolution of the probability distribution of the outermost ply $p(x_n, t)$. It is interesting to note that in the early stages of the search, both the probability of 75° and the probability of 0° (the two local optima) increase. But after about 2,000 analyses, the probability of 75° starts to decrease and the algorithm converges to the global optimum.

[3] A more general result is given in Mühlenbein et al. (1999) where the convergence of "Factorized Decomposition Algorithm" (FDA) to the optima is proved for additively decomposed functions. Separable functions are a special case of additively decomposed functions and UMDA is the corresponding simplified FDA.

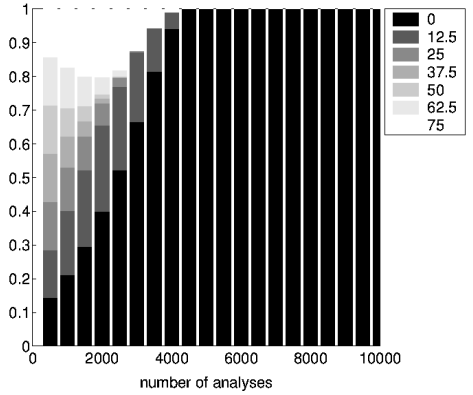


Fig. 8. Evolution of the probability distribution for the outermost ply, Min A_{66} problem

4 Elements of Theoretical Explanations

The numerical experiments that were just described are now theoretically analyzed by calculating expected convergence times. There are n variables that can take c discrete values.

4.1 Convergence Time of SHC

The following analysis considers a stochastic hill climber operating on a unimodal function. If the SHC is at a point where k out of the n variables are correctly set, the expected time before one of the non optimal variable is perturbed is $n/(n-k)$. The random perturbation can then take the variable closer to the optimum or not, with probabilities 1/2 (neglecting distortions due to limits on the values). The expected time for one beneficial step is then $2n/(n-k)$. Let d_i denote the average distance between the i-th variables of a random point and the optimum,

$$d_i = \frac{1}{c} \sum_{j=1}^{c} |x_i^j - x_i^*| \,, \tag{10}$$

where x_i^j is the j-th possible value of the i-th variable. In the cases presented here, all variables values at the optimum are the same ($x_i^* = 0°$ for Max A_{11} and Min A_{66} or $x_i^* = 45°$ for the vibration problem), therefore the average distance to the optimum is the same for all variables i, $d_i = d$ (but it varies with the problem). At each variable that is not correctly set, an average of d steps in the right direction is needed to reach the optimum. By summing the expected times of each beneficial step, one obtains the expected time to locate the optimum

from a random starting point that has k optimal variables

$$T_k = \sum_{i=k}^{n-1} \frac{2dn}{n-i} \,. \tag{11}$$

T_k can now be averaged over all random starting points, which yields the expected convergence time of an SHC on a unimodal function,

$$T_{SHC} = \frac{1}{2^n} \sum_{k=0}^{n} C_n^k T_k = \frac{nd}{2^{n-1}} \sum_{k=0}^{n} C_n^k \sum_{i=k}^{n-1} \frac{1}{n-i} \,. \tag{12}$$

It can be shown Garnier et al. (1999) that (12) yields a convergence time of order $\mathcal{O}(n \ln n)$ for large n. Estimated and measured convergence times are compared for both the Max A_{11} and the vibration problems in Fig. 9. The dependency of the number of function evaluations in terms of the dimension n is correctly predicted. Because d is smaller in the vibration problem than in the Max A_{11} problem, convergence is faster in the first case, which is also correctly predicted.

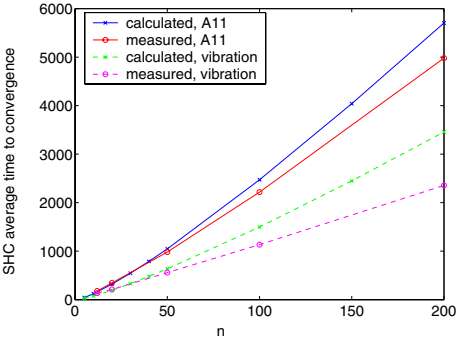


Fig. 9. Estimated and measured convergence times of SHC for the Max A_{11} and the vibration problems

4.2 Convergence Time of UMDA

A univariate marginal distribution algorithm is now considered. In Mühlenbein et al. (1999), the behavior of an UMDA with truncation selection is studied on two model functions, the *Onemax* and the *Int*, that have common features with the Max A_{11} and the vibration problems. In the *Onemax* problem, the number of 1's in a binary string is maximized. Like in Max A_{11}, the function is separable, and each variable has the same contribution to the objective function. The *Int* function,

$$Int = \sum_{i=1}^{n} 2^{i-1} x_i \,, \tag{13}$$

is also maximized on binary strings. Like in the vibration problem, the function is separable and there is a gradual influence of the variables on the function. In the vibration problem however, the difference in sensitivity of the objective function to each variable is lower than in Int. If the population size m, is larger than a critical value m^*, Mühlenbein shows that the expected number of generations to convergence[4] N_g, is

$$N_g \approx \mathcal{O}(\sqrt{n}) \qquad \text{for } Onemax \tag{14}$$

$$N_g \approx \mathcal{O}(n) \qquad \text{for } Int \,. \tag{15}$$

The larger number of generations seen on Int is due to the different weights variables have on the objective function. For a truncation rate $\tau = 0.5$, the first selection is exclusively based on x_n, the second selection is based on x_{n-1}, etc. The discovery of the optimum is sequential in variable values, whereas some level of parallelism can be achieved on less hierarchical objective functions.

The expected number of objective function evaluations to convergence is

$$N_f = N_g m^* \,. \tag{16}$$

No analytical expression for m^* was given by Mühenbein et al. An approximation to m^*, $\widehat{m^*}$ is now proposed based on the initial random population sampling, and neglecting variable values lost during selection. The probability that a given variable value is not represented in the population is $((c-1)/c)^m$. The probability that the values making up the optimum, x^*, have at least one sample in the initial population is

$$P_{pop} = \left(1 - \left(\frac{c-1}{c}\right)^m\right)^n \,. \tag{17}$$

For a given P_{pop} (typically close to 1), the critical population size is estimated from (17),

$$\widehat{m^*} = \frac{\ln(n/(1-P_{pop}))}{\ln(c/(c-1))} \;\approx\; \mathcal{O}(\ln(n)) \,. \tag{18}$$

From (14) to (18), the order of magnitude of the number of evaluations to convergence is

$$N_f \approx \mathcal{O}(\sqrt{n}\ln(n)) \qquad \text{for } Onemax \text{ and} \tag{19}$$

$$N_f \approx \mathcal{O}(n\ln(n)) \qquad \text{for } Int \,. \tag{20}$$

These orders of magnitude agree well with the UMDA convergence trends seen on the Max A_{11} and vibration tests in Sect. 3. Note however that, because the vibration problem is not as hierarchical as Int in terms of variables, its experimental convergence time behaves more like $\sqrt{n}\ln(n)$ than $n\ln(n)$.

[4] Following Mühlenbein's work, convergence time is defined here as the time when $p(x^*, N_g) = 1$.

5 Concluding Remarks

The asymptotic time to convergence of the SHC on a unimodal function is $n \ln n$, while it is bound between $\sqrt{n}\ln(n)$ and $n\ln(n)$ for the UMDA. Asymptotic behavior of UMDA is better than that of SHC on the Max A_{11} test case. Numerical experiments show that this asymptotic behavior does not take place until after $n = 200$ variables since SHC has always converged to the optimum faster than UMDA. It should be stressed that this study is primarily intended at characterizing the dimensionality effects on SHC and UMDA, irrespectively of other optimization difficulties. When multimodality is introduced through the separable function A_{66}, the reliability of the SHC is logically proportional to the percentage of the design space spanned by the basin of attraction of the global optimum while UMDA robustly finds the global optimum.

References

Mühlenbein, H., Paaß, G.: From recombination of genes to the estimation of distributions, I. Binary parameters. Lecture Notes in Computer Science 1141: Parallel Problem Solving from Nature IV (1996) 178–187.

Garnier, J., Kallel, L., Schoenauer, M.: Rigorous hitting times for binary mutations. Evolutionary Computation **7** (1999) 173–203.

Droste, S., Jansen, T., Wegener, I.: On the analysis of the (1+1) evolutionary algorithm. Theoretical Computer Science **276** (2002) 51–81.

Mitchell, M., Holland, J., 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 Publishers, Inc. (1994) 51–58.

Jansen, T., Wegener, I.: On the utility of populations in evolutionary algorithms. In: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001), San Francisco, California, USA, Morgan Kaufmann (2001) 1034–1041.

He, J., Yao, X.: From an individual to a population: An analysis of the first hitting time of population-based evolutionary algorithms. IEEE Transactions on Evolutionary Computation **6** (2002) 495–511.

Garnier, J., Kallel, L.: Statistical distribution of the convergence time of evolutionary algorithms for long-path problems. IEEE Transactions on evolutionary computation **4** (2000) 16–30.

Mühlenbein, H., Mahnig, T., Rodrigez, A.: Schemata, distributions and graphical models. J. of Heuristics **5** (1999) 215–247.

Pelikan, M., Goldberg, D., Sastry, K.: Bayesian optimization algorithm, decision graphs, and Occam's razor. Technical Report 2000020, IlliGAL (2000).

Evolutionary Search for Binary Strings with Low Aperiodic Auto-correlations

Sebastien Aupetit[1], Pierre Liardet[2], and Mohamed Slimane[1]

[1] École Polytechnique de l'Université de Tours, Laboratoire d'Informatique
64, Av. Jean Portalis, 37200 Tours, France
sebastien.aupetit@etu.univ-tours.fr
slimane@univ-tours.fr

[2] Université de Provence, Centre de Mathématiques Informatique et Mécanique
Laboratoire ATP, UMR-CNRS no. 6632
39, rue F. Joliot-Curie, 13453 Marseille cedex 13, France
liardet@cmi.univ-mrs.fr

Abstract. In this paper we apply evolutionary methods to find finite binary ±1-sequences with low out-of-phase aperiodic auto-correlations. These sequences have important applications in communication or statistical mechanics, but their construction is a difficult computational problem. The Golay Factor of Merit is studied from a probabilistic point of view, in order to explain the poor efficiency of evolutionary algorithms. Various genetic algorithms are then proposed and tested.

1 Introduction

Let B denote the binary alphabet $\{+1, -1\}$ and let $x = x_0 \cdots x_{N-1}$ $(\in B^N)$ be a B-string of length $|x| = N$. The so-called aperiodic auto-correlation of x is the finite $\mathbf{Z}$-valued sequence $\Gamma(x)$ defined by

$$\Gamma(x)_k = \sum_{0 \le j < N-k} x_j x_{j+k}, \quad 0 \le k < N.$$

Readily, $\Gamma(x)_0 = N$.

Binary sequences (or B-strings) x with flat out-of-phase aperiodic auto-correlations have applications in communication engineering [12] or statistical mechanics [2], for example. Classically, two interesting factors of merit have been introduced in the literature. The oldest one, that we call the extreme merit factor $\mathtt{EMF}(x)$, is defined for any $x \in B^N$ by

$$\mathtt{EMF}(x) = \max\{|\Gamma(x)_k| \,;\, 1 \le k < N\}.$$

The main problem is both to find the quantity

$$M_N = \min\{\mathtt{EMF}(x) \,;\, x \in B^N\}$$

P. Liardet et al. (Eds.): EA 2003, LNCS 2936, pp. 39–50, 2004.

and the so-called *optimal* or extreme sequences which realize this minimum. It is an old but unsolved problem in Discrete Mathematics to find exact values of M_N in general and to construct binary optimal sequences. The situation is even worse: none trivial asymptotic upper bound for M_N is known, but it seems reasonable to conjecture that M_N is of order $\sqrt{N}$.

The second factor of merit, first intensively studied by Golay [3], [4], and later on by J. Bernasconi [2] and Mertens [9] in connection with some Ising models, is the (aperiodic) Merit Factor, defined by

$$\mathtt{MF}(x) = \frac{N^2}{2E(\Gamma(x))}$$

where $E(\Gamma(x))$, viewed as an out-of-phase energy, is obtained from the map $E : \mathbf{C}^N \to \mathbf{C}^N$ given by

$$E(u) = \sum_{1 \le k < N} |u_k|^2 \ .$$

A fundamental problem is then to minimize the energy *i.e.*, to find

$$E_N = \min\{E(\Gamma(x))\,;\ x \in B^N\}$$

or to maximize the Merit Factor:

$$F_N = \max\{\mathtt{MF}(x)\,;\ x \in B^N\},$$

and also, to find sequences which realize such a maximum. At the present time, no efficient algorithm for finding sequences with the best MF is available, except the obvious one by exhaustive enumeration. This has been done first by Golay [3], for $N \le 32$, who has conjectured that F_N is bounded above by 12 or so. S. Mertens [9], using a *branch and bound* method exhibits a more efficient search compare to the exhaustive one, and give, up to $N = 48$, a complete table of values for E_N. His results yields a minimal Merit Factor around 9.3. On the other hand, at least up to N=199, heuristic searches [1] suggest that lower bound for F_N is about 6 for long skew-symmetric sequences x. Recall that, by definition, a skew-symmetric sequence verifies the relations

$$x_{n-1-s} = (-1)^s x_{n-1+s} \tag{1}$$

for $N = 2n-1$ and $1 \le s \le n-1$. The value 6 is also the highest asymptotic lower bound of MF which is known for infinite families of sequences, built from Legendre symbol or Modified-Jacobi symbol, well known in number theory (see [6] and [7]). Other infinite families of sequences are derived from cyclotomic classes in Galois fields $\mathbf{Z}/p\mathbf{Z}$. They lead to sequences of Merit Factor that converge to 6 [10].

For N large ($N > 100$), there is no hope to find exact optimal sequences by exhaustive calculation. Consequently, various approaches involving stochastic search algorithms have been proposed, like simulated annealing [2], [3], evolutionary algorithms [5] or hybrid local search algorithm [11]. In fact, all these algorithms perform quite poorly. One main reason for that is due to a very scattered distribution of optimal sequences in the search space [9], with no efficient

evolutionary methods driving toward optimal sequences, despite of the obvious fact that a single modification of any sequence x by replacing one component x_k by its opposite $-x_k$, just change EFM of a quantity bounded in absolute value by 2.

In this paper, we first compute theoretically the mean and variance of the random variables $E \circ \Gamma(\cdot)/N^2$ and $E \circ C(\cdot)/N^2$. It appears that, when N tends to infinity these random variables converge in probability to constants $1/2$ and 1 respectively. After that, we introduce several new evolutionary algorithms for estimating E_N and F_N by using various strategies of searching. Since all these algorithms are stochastic, we devote section 2 to a simple statistical analysis of the pseudo-random generator in use. To do that, we compute by trials the distribution of the random variable $\mathtt{MF}(\cdot)$ defined on the space B^N endowed with the uniform probability P. We also consider the periodic case which is simpler from a mathematical point of view. The so-called Cyclic Merit Factor CMF is defined for $x \in B^N$ by

$$\mathtt{CMF}(x) = \frac{N^2}{2E(C(x))}$$

where $C(x)$ denote the cyclic auto-correlation sequence of x which, at phase k, is given by

$$C(x)_k = \sum_{0 \le i < N} x_i x_{i+k}$$

(replacing in the sum x_j by x_{j-N} if $j \ge N$). Notice that the CMF has been already widely explored, theoretically and by experiments.

In section 2, we consider very elementary but quite natural algorithms which are used as basic references for evaluating more sophisticated algorithms inherited from classical genetic schemes. They are one-to-one evolutionary algorithms in the sense that they start with only one individual u on which a one single mutation-selection is performed step by step. In the first algorithm (1-RandomSequential) no assumption is assumed on the structure of u. Next, in the algorithm RandomSkewSequence, u runs in the set of skew-symmetric sequences and used also a one single step by step mutation. The third algorithm goes back to the first one but, at each step, on the running position, t nearby mutations are performed and the issues are evaluated. These algorithms are repeated on M individuals, chosen at random, and best results are collected.

Section 3 deals with more classical evolutionary algorithms (EA). After initialization of a population with M individuals ($M = 2000$), a basic round is iterated until a stop-round test applies. The round in the first algorithm is essentially a mutation-selection. The round in the two other algorithms consists in coupling-selection-mutation operators.

Section 4 draws conclusions, proposes conjectural properties and select evolutionary algorithms to find long binary sequences with good Merit Factors.

2 Random Tests and 1-1 Evolutionary Algorithms

Obviously, we cannot avoid the dependency of any evolutionary algorithm with a pseudo-random generator. For our purpose, we have used the `rand()` function available in C++ library, which corresponds to the left truncation reduced mod 2^{15} of the linear congruential generator $g_{n+1} = ag_n + b \pmod{2^{32}}$ with $a = 214013$ and $b = 2531011$. Despite of regularities in binary representation of the generated number (see [8]) produced by such a generator and the relatively short period (in fact, equal to 2^{15}), it will be good enough for our experiments which usually require series of 2000 trails. Suggested by Referees, we have compare these experiments with those obtained by using the Mersenne Twister (MT) pseudo-random generator: the empirical distribution functions in both cases are quite identical (see Figures 1 and 2).

The search space B^N is endowed with the uniform probability P. Therefore, the maps $x \mapsto x_k$ $(0 \leq k < N)$, defined on B^N, are i.i. random variables with law $P(x_k = 1) = P(x_k = -1) = 1/2$. Maps $E \circ \Gamma(x)$, $E \circ C(x)$, $\mathtt{MF}(x)$ and $\mathtt{CMF}(\cdot)$ defined above are also considered as random variables. The expectation of $E \circ \Gamma(x)$ can be computed exactly. In fact, for $1 \leq k < N$,

$$\begin{aligned}\int (\Gamma(x)_k)^2 P(dx) &= \sum_{0 \leq i,j < N-k} \int x_i x_{i+k} x_j x_{j+k} P(dx) \\ &= (N-k) + 2 \sum_{0 \leq i < j < N-k} \int x_i x_{i+k} x_j x_{j+k} P(dx) = N - k\,.\end{aligned}$$

Hence,
$$\int E \circ \Gamma(x) P(dx) = \sum_{0 < k < N} (N-k) = N(N-1)/2\,.$$

Now we compute the expectation of $E \circ C$:

$$\begin{aligned}\int (C(x)_k)^2 P(dx) &= \sum_{0 \leq i,j < N} \int x_i x_{i+k} x_j x_{j+k} P(dx) \\ &= N + 2 \sum_{0 \leq i < N} \int x_i x_{i+k}^2 x_{i+2k} P(dx) \\ &= N - k + e(N,k),\end{aligned}$$

where $e(N,k) = 0$ if $k \neq N/2$ and, for N even, $e(N, N/2) = 1$. Therefore

$$\int E \circ C(x) P(dx) = N^2 + e_N$$

with $e_N = 1$ if N is even and $e_N = 0$ otherwise. These results show, in particular, that the expectation of $E(\Gamma(x))$ is about half the expectation of $E(C(x))$.

We also compute the variance. One has

$$\mathrm{Var}(E \circ \Gamma) = \int (E \circ \Gamma(x))^2 P(dx) - \left[\int E \circ \Gamma(x) P(dx) \right]^2$$

$$= \sum_{1\le r,s<N} \int (\Gamma(x)_r \Gamma(x)_s)^2 P(dx) - \left[\frac{N(N-1)}{2}\right]^2$$
$$= \sum_{1\le k<N} \int (\Gamma(x)_k)^4 P(dx) + 2 \sum_{1\le r<s<N} \int \Gamma(x)_r \Gamma_s(x) P(dx)$$
$$- \left[\frac{N(N-1)}{2}\right]^2 .$$

But

$$\int (\Gamma(x)_k)^4 P(dx) = \sum_{0\le i,j,r,s<N-k} \int x_i x_{i+k} x_j x_{j+k} x_r x_{r+k} x_s x_{s+k} P(dx)$$

where partial integral in the sum is null except if $i = j$, $r = s$, or $i = r$, $j = s$, or $i = s$ with $j = r$, but all cases $i = j = r = s$ have to be considered only once. Therefore

$$\int (\Gamma(x)_k)^4 P(dx) = 3(N-k)^2 - 2(N-k)$$

and consequently

$$\sum_{1\le k<N} \int (\Gamma(x)_k)^4 P(dx) = \frac{N(N-1)(2N-1)}{2} - N(N-1)\,. \tag{2}$$

The computation of $I_{rs} = \int \Gamma(x)_r \Gamma(x)_s P(dx)$ $(r < s)$ is a little bit harder. Expanding the product of correlations, one gets

$$I_{rs} = \sum_{\substack{0\le i,i'<N-r \\ 0\le j,j'<N-s}} I_{rs}(i,i',j,j')$$

with

$$I_{rs}(i,i',j,j') = \int x_i x_{i+r} x_{i'} x_{i'+r} x_j x_{j+s} x_{j'} x_{j'+s} P(dx).$$

It is clear that $I_{rs}(i,i,j,j) = 1$ while $I_{rs}(i,i,j,j') = 0$ if $j \ne j'$ and similarly $I_{rs}(i,i',j,j) = 0$ for $i \ne i'$. Hence

$$\sum_{\substack{0\le i<N-r \\ 0\le j<N-s}} I_{rs}(i,i,j,j) = (N-r)(N-s). \tag{3}$$

Since i and i' (resp. j and j') play the same role in the summation, we may assume that $i < i'$, $j < j'$ in the remaining part, the result being multiplied by 4. Now, $I_{rs}(i,i',j,j') = 0$ if $i < j$ or $j < i$ and for $i = j$, in order to have $I_{rs}(i,i',i,j') \ne 0$, it is necessary that $i' + r = j' + s$, which implies $i' = i + s$ and $j' = i + r$, so that

$$I_{r,s}(i,i+r,i+s,i+s+r,i,i+s,i+r,i+r+s) = 1 \tag{4}$$

with $0 \le i < N - r - s$, and all the other integrals, not considered above, are null. Using (3) and (4) we get

$$I_{rs} = (N-r)(N-s) + 4(N-r-s)\,.$$

Standard computations lead to

$$2\sum_{1\le r<s<N} I_{rs} = \left[\frac{N(N-1)}{2}\right]^2 - \frac{N(N-1)(2N-1)}{6} + 4N(N-1)$$

and finally

$$\mathrm{Var}(E\circ\Gamma) = \frac{N(N-1)(2N-1)}{2} + 3N(N-1) - \frac{N(N-1)(2N-1)}{6},$$

or

$$\mathrm{Var}(E\circ\Gamma) = \frac{2}{3}N(N-1)(N+4)\,.$$

The cyclic case is more complicated, but the following estimate is easy to compute:

$$\mathrm{Var}(E\circ C) = 6N^3 + \mathcal{O}(N^2)\,.$$

By the classical Bienaymé-Tchebychev inequality, we derive

$$P\Big(\Big|\frac{E\circ\Gamma(x)}{N^2} - \frac{1}{2}\Big| \ge \frac{t}{\sqrt{2N/3}}\Big) \le \frac{1}{t^2} \tag{5}$$

for $N \ge 6$, and

$$P\Big(\Big|\frac{E\circ C(x)}{N^2} - 1\Big| \ge \frac{t}{\sqrt{6N}}\Big) \le \frac{1}{t^2} \tag{6}$$

for N large enough. Taking $t = \varepsilon\sqrt{2N/3}$ with $\varepsilon > 0$ small in formula (5), we get

$$P\Big(\Big|\frac{E\circ\Gamma(x)}{N^2} - \frac{1}{2}\Big| \ge \varepsilon\Big) \le \frac{3}{2\varepsilon^2 N}$$

and, using (6), a similar estimate follows for $E\circ C(x)/N^2$.

Going back to the Merit Factors, we see that, from the probabilistic point of view, $\mathtt{MF}(x)$ and $\mathtt{CMF}(x)$ are concentrated around the value 1 and 1/2 respectively. We can be more precise: the random variables $\mathtt{MF}(x)$ and $\mathtt{CMF}(x)$ converge in probability to constants 1 and 1/2 respectively, that is to say:

Theorem 1. For all $\eta > 0$, $\lim_{N\to\infty} P(\big|\mathtt{MF}(x) - 1\big| \ge \eta) = 0$.

Theorem 2. For all $\eta > 0$, $\lim_{N\to\infty} P(\big|\mathtt{CMF}(x) - \frac{1}{2}\big| \ge \eta) = 0$.

In Figure 1 we have plotted classical empirical distribution functions of the Merit Factor MF, obtained by simulation with $N = 199$ and $M = 2000$ trials, using the rand() C++ function and the Mersenne Twister (MT) PRNG. Fig 2 shows empirical distribution functions of the Cyclic Merit Factor CMF issuing from another analogous independent series of $M = 2000$ trials. The corresponding histogram of density obtained with the Mersenne Twister PRNG is also plotted.

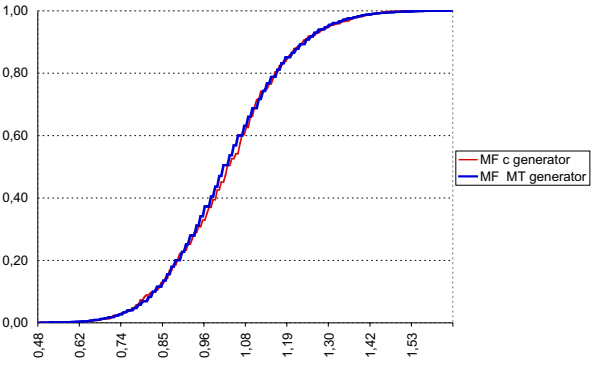


Fig. 1. Empirical distributions of Merit Factor `MF` ($N = 199$, $M = 2000$ trials) obtained with the rand() C++ function (resp. MT): min=0,479628 (0,479628), max=1.782063 (1.769038), mean=1.03155 (1.08175626); Maximal deviation (between both curves): 0.05790

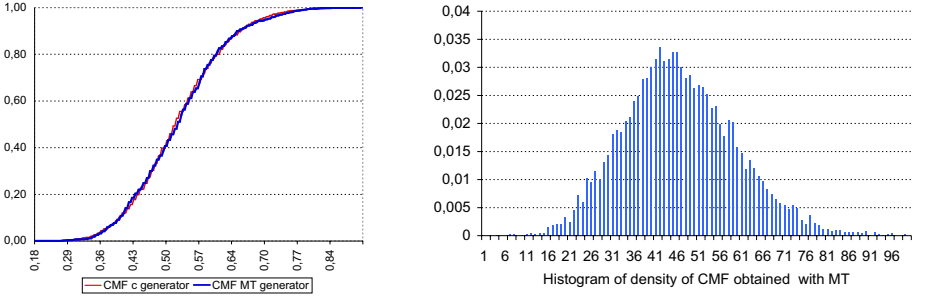


Fig. 2. Empirical distributions of Cyclic Merit Factor `CMF` (N=199, M=2000 trials) obtained with the rand() C++ function (resp. MT): min=0.182060 (0.182060), max=0.937257 (0.929705), mean=0.522708 (0.562830), maximal deviation: 0.044400

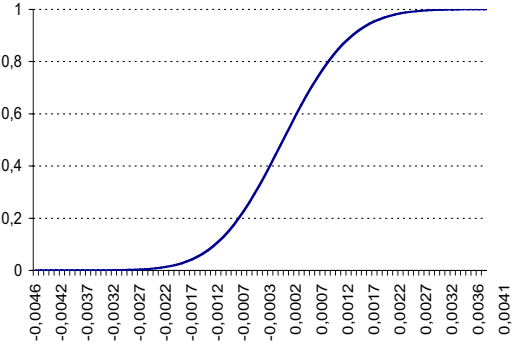


Fig. 3. Empirical distribution of the Merit Factor `MF` after normalization (N=199, M=100000 trials, min=-0.004641, max=0.005109), using Rand() C++ function

In Fig 3, we have plotted the empirical distribution of $\mathtt{MF}$ computed with 10^5 trials, but after normalization (so that the mean is 0 and the variance equal 1). All values computed are between -0.004641 and 0.005109, furnishing an empirical evidence (if it is required) of Theorem 1.

In our next stochastic algorithms, the input data is a random sequence x in B^N and the output is a sequence x^* ($= \mathtt{S}_1(x)$, $\mathtt{S}^*(x)$, and $\mathtt{S}_t(x)$) in B^N such that $\mathtt{EMF}(x^*) \leq \mathtt{EM}(x)$ (or $\mathtt{MF}(x^*) \geq \mathtt{MF}(x)$). Let $\mathtt{Flip}_{k,1} : B^N \to B^N$ be the map such that $(\mathtt{Flip}_{k,1}(x))_j = x_j$ if $j \neq k$ and $(\mathtt{Flip}_{k,1}(x))_k = -x_k$, consider the map $S_{k,1} : B^N \to B^N$ given by

$$S_{k,1}(x) = \begin{cases} x & \text{if } \mathtt{EMF}(x) \leq \mathtt{EMF}(\mathtt{Flip}_{k,1}(x)), \\ \mathtt{Flip}_{k,1}(x) & \text{otherwise,} \end{cases}$$

and define $S_1 : B^N \to B^N$ by

$$S_1(x) = S_{N-1,1} \circ \cdots \circ S_{1,1} \circ S_{0,1}(x).$$

The first algorithm, called $\mathtt{RandomSequential}_1()$ ($\mathtt{RS}_1()$ for short), is furnished by

Algorithm `1-RandomSequential` [$\mathtt{RS}_1()$]
begin
$x =$ empty string;
for $k = 0$ to $N - 1$ do $x \leftarrow x{\cdot}\mathtt{Rand}_0()$;
output $\mathtt{S}_1(x)$
end.

where $x{\cdot}\mathtt{Rand}_0()$ is the concatenation of the string x and the output of a suitable B-valued pseudo-random generator.

The next algorithm is applied on the set $(SB)_N$ of skew-symmetric sequences x with $N = 2n - 1$ (see (1)). For $1 \leq k < n - 1$, let $S_k : (SB)_N \to (SB)_N$ be the map

$$S_k^*(x) = \begin{cases} x & \text{if } \mathtt{EMF}(x) \leq \mathtt{EMF}(\mathtt{Flip}_{n-1-k}(\mathtt{Flip}_{n-1+k}(x))), \\ \mathtt{Flip}_{n-1-k}(\mathtt{Flip}_{n-1+k}(x)) & \text{otherwise,} \end{cases}$$

and set

$$S^* = S_{N-1}^* \circ \ldots \circ S_0^*.$$

Then, we define

Algorithm `RandomSkewSequential` [$\mathtt{RS}^*()$]
begin
$x = 1$;
for $s = 1$ to $n - 1$ do
$R \leftarrow \mathtt{Rand}_0()$ and $x \leftarrow R{\cdot}x{\cdot}(-R)$;
output $\mathtt{S}^*(x)$;
end.

The last algorithm of this section, built for $1 \leq t < N$ and called `t-Random-Sequential`, is a natural generalization of $\mathtt{RS}_1$. For any $w = w_0 \cdots w_{t-1} \in B^t$ and $0 \leq k < N - t$ we introduce $\mathtt{Flip}_{k,w} : B^N \to B^N$ by setting $(\mathtt{Flip}_{k,w}(x))_j = x_j$ if $0 \leq j < k$ or $k + t \leq j < N$, and $(\mathtt{Flip}_{k,w}(x))_j = w_{k-j}x_j$ for $k \leq j < k + t$. For convenience, put $\mathrm{Val}(w) = \sum_{0 \leq k < t}(1 - w_k)2^{k-1}$ and notice that $\mathrm{Val}(\cdot)$ is a bijection from B^N to $\{0, \ldots, 2^t - 1\}$. The map $S_{k,1}$ is generalized by $S_{k,t} : B^N \to B^N$ as follows. The string $y = S_{k,t}(x)$ is the one given by $y = \mathtt{Flip}_{k,w}(x)$ where w is the unique string in B^N which realizes the minimum of $\mathrm{Val}(\cdot)$ among the strings $v \in B^N$ such that $\mathtt{EMF}(\mathtt{Flip}_{k,v}(x)) \leq \mathtt{EMF}(\mathtt{Flip}_{k,u}(x))$ for all $u \in B^t$. Finally

$$S_t = S_{N-1-t,t} \circ \cdots \circ S_{0,t}$$

and we define:

Algorithm `t-RandomSequential` $[\mathtt{RS}_t()]$
begin
$x =$ empty string;
for $k = 0$ to $N - 1$ do $x \leftarrow x \cdot \mathtt{Rand}_0()$;
output $\mathtt{S}_t(x)$
end.

Experiments. We have performed these algorithms M times, independently. Results are collected in Tables 1, 2 and 3 below. They give the best and worst `EMF` (resp.`FM`) values among $M = 2000$ trials (with the total number of extrema values obtained in the series), the mean μ, and the standard deviation σ of the output series. The input column corresponds to best and worst EMF values of random sequences built by using the Rand() C++ function. Results with the MT pseudorandom generator are quite similar.

Table 1: RS_1					output							
input					EMF				MF			
min	nb	max	nb	μ	σ	min	nb	max	nb	μ	σ	max	nb	min	nb	μ	σ
22	7	62	1	33.849	5.338	16	6	29	2	20.73	1.861	1.912	1	0.822	1	1.311	0.173

The best `EMF` is equal to 16; it is obtained with 6 sequences, whose the one which has the best `MF` (=1.9120) is given in long-run notation by

1122341461121211121111112122212151212112613131132222131111211351111132111511241141235113112114242134 61

Table 2:RS^*					output							
input					EMF				MF			
min	nb	max	nb	μ	σ	min	nb	max	nb	μ	σ	max	nb	min	nb	μ	σ
25	1	83	1	43.381	7.997	17	1	37	2	25.572	2.660	2.695	1	0.826	1	1.509	0.264

The optimal sequence with `EMF`=17 realizes also the best Merit Factor `MF`= 2.695, which is much better than the one obtained above. This result is in accordance with the constant observation, reported in the literature, that stochastic algorithms on this problem give better results if they are restricted to the search space of skew-symmetric sequences.

The latest test gives

Table 3:RS_5						output									
input						EMF						MF			
min	nb	max	nb	μ	σ	min	nb	max	nb	μ	σ	max	nb	min	nb	μ	σ
21	2	59	1	33.895	5.384	16	9	29	3	20.656	1.880	2.008	1	0.683	1	1.323	0.179

In this series, the best EMF is 16 which realizes the maximal MF (=2.008), a performance which is better that the one furnished by $\mathtt{RS}_1$ (as it was expected) but is worse that the one obtain with $\mathtt{RS}^*$. The sequence is issued from an input having EMF=34 and MF= 0,922.

These experiments show that these algorithms move EMF downward, MT upward, and decrease the standard deviation by a factor of at least 1/2.

3 Evolutionary Machineries

The preceding algorithms proceed only by mutation-selection acting on one individual. Therefore we are far from classical genetic operators which act simultaneously on several individuals and involve crossing-over operators or general combinations, producing children that have to pass a selection process. In order to estimate the effect of these combinations, we start with a "pure mutation" algorithm.

Algorithm EA0

Input: initialization of a population P_0 whose individuals are chosen randomly in B^N.
Evolution $P_{n+1} \leftarrow P_n$ by mutation:
for each individual x in P_n, chose randomly $k \in \{0, \ldots, N-1\}$ and replace x by $S_{k,1}(x)$.
Evaluation: $m_{n+1} = \min\{EMF(x); x \in P_{n+1}\}$.
Repeat until $n > 10$ and $m_n = m_{n-10}$.
Output $n - 10$ and m_n.

For the sequel it is convenient to introduce the map $M_k = (B^N)^k \to B^N$ $(k \geq 1)$, with $M_k(u^{(1)}, \ldots, u^{(k)}) = u^{(j)}$ if j is the greatest integer such that $\mathtt{EMF}(u^{(j)}) = \min\{\mathtt{EMF}(u^{(i)}\,;\ 1 \leq i \leq k\}$. The second algorithm is derived from the above by changing the *Evolution* round by the following:

Algorithm EA01

...
Evolution $P_{n+1} \leftarrow P_n$ by combination:
choose at random M couples $(u, v) \in P_n \times P_n$;
replace u by $M_3(u, v, ((u_0 \times v_0) \cdots (u_{N-1} \times v_{N-1}))$;
do Algorithm EA0.
...

The last algorithm is also a modification of the *Evolution* round in `EA0`:

Algorithm `EA03`

...
Evolution $P_{n+1} \leftarrow P_n$ by crossing-over:
choose at random M *couples* $(u, v) \in P_n \times P_n$;
choose at random $k \in \{0, \ldots N-1\}$ *and*
set $u = u'u''$, $v = v'v''$ *with* $|u'| = |v'| = k$;
replace u *by* $M_4(u, v, u'v'', v'u'')$;
do Algorithm `EA0`.
...

We have performed 20 times these algorithms with $M = 2000$ and have collected in Table 4 the minimal, maximal values obtained for EMF, with the numbers of strings which reach such values, the maximal number of rounds, minus the 10 extra rounds for stopping, and the classical statistics (mean μ and standard deviation σ).

Table 4 (with EMF)	EA0	EA01	EA03
min	17	15	15
nb	1	9	12
max	20	16	16
nb	2	11	8
μ	18	15.55	15,4
σ	0.7539	0.5104	0.5026
$Max(n) - 10 =$	24	5	10

Now, we modify algorithms `EA0`, `EA01`, `EA03` by taking the Merit Factor `MF` for evaluating the strings (in place of the Extrema Merit Factor `EMF`). We run 20 times these new algorithms, respectively denoted by `EA'0`, `EA'01` and `EA'03`, with initial populations of 2000 individuals. Table 5 summarizes the results. Recall that the best strings are those which have the greatest Merit Factor.

Table 5 (with MF)	EA'0	EA'01	EA'03
max	2.973	3.705	3.826
nb	1	1	1
min	2.056	3.309	3.356
nb	1	1	1
μ	2.713	3.468	3.526
σ	0.216	0.117	0.135
$Max(n) - 10 =$	112	18	25

One observes that `EA03` (resp. `EA'03`) is more efficient that `EA0` while `EA01` (resp. `EA'01`) seems to be comparable to `EA03` (resp. `EA'03`). This allows us to conclude that the introduction of *combination* or *crossing-over* operators improves the performance, but we are far from the asymptotic lower bound 6 for `MF`.

4 Conclusion

The main theoretical observation proved in this studies is that the classical factors of merit `MF` and `CMF` used to evaluate strings having flat out-of-phase energy converge in probability to 1 and 1/2 respectively when the length N of string tends to infinity. This explains why stochastic algorithms are relatively inefficient to find optimal sequences. The `RS` algorithms can be viewed as stochastic operators which modifies the initial distribution of `MF` by concentrating the values around a larger mean. It should be of interest to prove that iterations of these operators lead to a constant, and to compute it. The experiments confirm the interest to choose crossing-over operators but they are not enough powerful, due to a very scattered distribution of optimal strings (see [2]) with no evident relation between the Merit Factor of a given string and the values of `MF`() in its neighborhood (according to the Hamming distance). Nevertheless, Algorithms `EA03` or `EA'03` can be used to produce quickly very long strings with good `EMF` or good `MF`, respectively.

References

1. Beenker G.F.M., Claasen T.A.C.M and Hermens P.W.C., *Binary sequences with maximally flat amplitude spectrum*, Philips J. Res. **40** (1985), 289–304.
2. Bernasconi J., *Low autocorrelation binary sequences: statistical mechanics and configuration space analysis*, J. Physique, **48** (1987), 559.
3. Golay M.J.E., *The merit factor of long low autocorrelation binary sequences* , IEEE Trans. Inf. theory, **IT-23** (1982) 43–51.
4. Golay M.J.E., *The merit factor of Legendre sequences*, IEEE Trans. Inf. theory, **IT-29**, 1983, 934-936.
5. de Groot C., Würtz D. and Hoffmann K.H., *Low Autocorrelation Binary Sequences: Exact Enumeration and Optimization by Evolutionary Strategies*, Optimization, **23** (1992), Gordon & Breach Science Publish. S. A., 369–384.
6. Høholdt T., *The Merit Factor of Binary sequences. Difference Sets, Sequences and their Correlation Properties*, A. pott and all (eds.), Series C: mathematical and Physical Sciences, Kluwer Acad. Publish. **542** (1999), 227–237.
7. Jensen J.M., Elbrønd Jensen H., Høholdt T., *The Merit Factor of Binary Sequences Related to Difference Sets*, IEEE Trans. Inform. Theory, **37**-3 (1991), 617–626.
8. D.E. Knuth, *The Art of Computer Programming*, vol. 2: Seminumerical Algorithms, Addison-Weslet, Reading,MA, 2nd edition (1981).
9. Mertens S.,*Exhaustive search for low-autocorrelation binary sequences*, J. Phys. A: Math. Gen. **29** (1996), 473–481
10. Parker M.G., *Even Length Binary Sequence Families with Low Negaperiodic Autocorrelation*, LNCS 2227 (2001), pp 200–210.
11. Prestwich S., *A Hybrid Local Search Algorithm for Low-Autocorrelation Binary Sequences*, Tech. Report TR-00-01, Dept. Comp. Sci., National Univ. Ireland at Cork, pp. 14.
12. Schroeder M.R., *Number Theory in Science and Communication*, Springer, Berlin (1984).

Order Statistics in Artificial Evolution

Stephane Puechmorel[1] and Daniel Delahaye[2]

[1] ENAC
7, Avenue Edouard Belin
31055 TOULOUSE CEDEX
[2] CENA

Abstract. This article deals with the exploitation of statistical information from extremes values of an evolutionary algorithm.
One can use the fact that upper order statistics of a sample converge to known distributions for improving efficiency of selection and crossover operators.
The work presented in this paper is restricted to criteria defined on real vector spaces. It relies on an underlying canonical model of genetic algorithm, namely tournament selection and uniform crossover. Nevertheless, the results obtained so far encourage further investigations.

1 Introduction

Evolutionary computation is recognized to be highly successful in solving real world optimization problems. Its robustness and ability to locate efficiently global optima among many local ones allows treatment of cases for which other methods fail. However, most of these properties rely on stochastic exploration of the search space and if the complexity of one iteration is low, the overall cost of the algorithm can be high due to the number of samples needed for locating the global optimum. Moreover, a compromise must be done between exploring the search space in order to find new candidate points and exploiting the already computed ones. Careful balancing is the key of speed-up in evolutionary computation and is a major area of research. The selection operator is the one which has greatest influence on this and several procedures has been designed, so that exploration or exploitation is enforced. There is two main classes of selection operators:

- Selection based on fitness. Operators belonging to this class use the value of the criterion to compute selection probabilities. The historical 'spin wheel selection' is one of them, like the 'stochastic remainder' which has better properties.
- Rank based selection. In this case, only the relative value of the criterion is used. All operator from this class may be thought as a two stage procedure: first sort the population, then draw with a given law inside the sorted sample. The popular tournament selection is rank based, and has a nice behavior on a broad range of problems. Furthermore, it is possible to tune the selection pressure by merely changing the size of the competitors pool.

P. Liardet et al. (Eds.): EA 2003, LNCS 2936, pp. 51–62, 2004.

In the following, we will restrict our attention to rank based selection, which will be the start point of our analysis (it is a quite consensual fact that rank based operators and specially tournament selection are among the best selection procedure available for evolutionary computation. Many of our own simulation agree with that point of view).

Another aspect of evolutionary algorithms is the design of mutation and crossover operators. Generally, mutation is an operator which avoids being trapped in local solutions and allows to enlarge the exploration scale. On the other hand, crossover are frequently used. In the following, we will address the problem of finding the global optimum of a criterion furnished by a real valued function f over an hyper-rectangle $\prod_{i=1}^{n}[a_i, b_i]$ so that operator will act on real vectors. Barycentric crossover is by far the most commonly used in this case. It is easy to see that the effect of crossing two parents is a uniform random trial on a segment containing them. The so called uniform crossover, that is drawing a new random linear combination for each component of the parent vectors is more efficient on many problems. Here again, it is easy to figure out that this operator samples points from a uniform deviation in the hyper-rectangle of $\mathbb{R}^n$ defined by the parents (the hyper-rectangle defined by two points x and y of $\mathbb{R}^n$ is the closed set $H(x, y) = \{u \in \mathbb{\$}^n | \forall\, i = 1 \dots n, x_i \leq u_i \leq y_i\}$).

Gathering the previous notes, the process underlying a whole generation inside an evolutionary algorithm may be described by a sampling distribution based on mixed uniform deviates on hyper-rectangle and depending only on the previous population. This point of view has been adopted in [DMPJ01]. An evolutionary algorithm may then been seen has a measure valued dynamical system. Using asymptotic properties of upper order statistics, it is possible to compute a threshold above which samples may be considered as belonging to a neighborhood of a maximum, while samples below will be assumed to belong to an exploratory pool. The sampling distribution obtain at each generation will then be a mix of a uniform deviate on the complement set of the hyper-rectangle containing all extreme samples and an exploitation distribution defined on this hyper-rectangle.

Simulation results on standard test problems will be presented and a comparison done with the standard tournament selection - uniform crossover algorithm.

2 Asymptotic Law of Extremes

In this section we will collect some classical results about limiting distributions of order statistics. Fundamental references are [Zhi91], [dH81]. Let E be a compact of $\mathbb{R}^d$ and let $\{X_1, \dots X_N\}$ a size n sample of common distribution P on E. Finally, let $f : E \to \mathbb{R}$ a P-measurable function. For a measurable subset $U \subset E$, the essential supremum of f on U, denoted $essup f_U$, is defined as follows:

- Define the function $F_U : t \in \mathbb{R} \to P\left(\{u \in U | f(u) < t\}\right)$;
- $esssup f_U = \inf\{t | F_U(t) = 1\}$.

The essential supremum is the value that will be searched by a stochastic algorithm : high values taken by f on zero measure subsets will not be taken into

account. With the notations above, let M be the essential supremum of f on E and F be F_E. Put

$$V : x \in \mathbb{R}^+ \to 1 - F(M - x^{-1}) ;$$

V is said to be of regular variation at infinity if it exists a real number $\alpha > 0$ such that for all $t > 0$:

$$\lim_{x \to +\infty} \frac{V(tx)}{V(x)} = t^{-\alpha} .$$

The exponent α is called the tail index of the distribution F. When V is of regular variation, there exists a limiting law for the sample maximum $\eta_N = \sup\{X_1, \ldots X_N\}$ in the following sense. there exists real sequences $(c_n)_{n \in \mathbb{N}}, (d_n)_{n \in \mathbb{N}}$ such that:

$$\lim_{n \to +\infty} F^n(c_n x + d_n) = \psi_\alpha(x) .$$

With F the distribution function and ψ_α defined by

$$\psi_\alpha(x) = \begin{cases} \exp(-(-x)^\alpha) , x < 0 , \\ 1 \qquad\qquad , x \geq 0 . \end{cases}$$

One choice of such sequences is $d_n = M, \forall n$ and $c_n = F^{-1\leftarrow}(1 - n^{-1})$ (the notation $F^{-1\leftarrow}$ denotes the right limit). Many cumulative density functions are of regular variation. This is obviously the case if we can find $\alpha > 0, \beta > 0)$ such that :

$$F(x) = 1 - \beta(M - x)^\alpha + o((M - x)^\alpha) .$$

It may be noticed that the definition of regular variation can be extended by relaxing the assumption on the limit expression and imposing only that there exists an mapping $h : \mathbb{R}^+ \to \mathbb{R}^+$ such that:

$$\lim_{x \to +\infty} \frac{V(tx)}{V(x)} = h(t) .$$

However, in this case we have for $(t_1, t_2) \in \mathbb{R}^{+2}$:

$$h(t_1 t_2) = \lim_{x \to +\infty} \frac{V(t_1 t_2 x)}{V(t_2 x)} \frac{V(t_2 x)}{V(x)} = h(t_1) h(t_2) .$$

It is a well known fact that continuous solutions of this functional equation are precisely of the for t^α, so that it turns out that the first form has full generality. Regular variation is not a stringent assumption. Most cumulative distribution functions obtained from maximization problems belong to this class. Furthermore, it can be shown [Zhi91] that $\alpha = d/2$ in the case of a class C^2 criterion with maxima inside E and uniform sampling in E. Following [Pic75], we define, for a threshold values ν, the conditional excess distribution function F_ν as:

$$F_\nu(t) = P(\{u \in E | \nu \leq f(u) < t\}) / P(\{u \in E | f(u) \geq \nu\}) = \frac{F(t) - F(\nu)}{1 - F(\nu)}, \; t > \nu .$$

If F_ν has regular variation of tail index α, putting $\nu = M - \theta^{-1}$ and $\lambda < 1$, one has

$$F_\nu(M - \lambda\theta^{-1}) = \frac{V(\theta) - V(\lambda^{-1}\theta)}{V(\theta)}\,,$$

so that

$$\lim_{\theta \to +\infty} F_\nu(M - \lambda\theta^{-1}) = 1 - \lambda^\alpha\,.$$

Except for the cases which can be analytically solved, the value of the tail index α can be estimated from the sample. Hill [Hil75] designed such a conditional maximum-likelihood estimate of α, which is easily computable:

$$\widehat{\alpha}_{k,N} = \left(\frac{\sum_{i=0}^{k-1} \log(\eta_{N-i})}{k} - log(\eta_{N-k}) \right)^{-1}$$

where $\eta_1 \geq \ldots \geq \eta_N$ are the order statistics of the sample and k is chosen from N so that $\lim_{N \to +\infty} kN^{-1} \to 1$.

3 Tournament Selection Analysis

3.1 Basic Results

Let n, m be integers such that $n \geq m \geq 1$. A (n, m)-tournament selection from a population $(X_i)_{i=1\ldots N}$ of size N produces individuals, first by uniformly sampling n individuals from the population (the pool of competitors), then selecting the m individuals with the highest (or lowest for minimization) criterion value. Increasing n increases the selection pressure and propension to elitism. Since pools of competitors are obtained by independent uniform sampling, noting will change if we apply a permutation σ to the population and apply tournament selection to the new population $(X_{\sigma(i)})_{i=1,\ldots,N}$. Among possible permutations, one may choose to order population in increasing order. Inside a pool of n individuals, those m with highest criterion value will be simply those with m highest indices. Formally, once the population has been order, one may describe the process of selecting m individual out of a pool of n by sampling the m highest order statistics from a discrete uniform law in the set $\{1, \ldots, N\}$, then selecting individual with matching indices. Join density of the m upper order statistics $\eta_{n-m+1}, \ldots, \eta_n$ of n independent uniform random variables is given by [Rei89]:

$$p_{\eta_{n-m+1},\ldots,\eta_n}(x_1, \ldots, x_m) = n! \frac{x_1^{n-m}}{(n-m)!}, \quad x_1 < \ldots < x_m\,.$$

For algorithmic implementation, it is convenient to realize sampling by successive draws from conditional laws. This can be done with the following procedure:

– Draw the real random variable y_m by sampling a real uniform random variable t in the interval $[0, 1]$, then computing $y_m = t^{n^{-1}}$.

- Assuming y_{k+1} has been obtained, y_k is obtained by sampling t and computing $y_k = y_{k+1} t^{(n-m+k)^{-1}}$.
- When all the y_i has been obtained, multiply each by N and round the result towards the nearest integer to obtain selected indices.

Some useful results may be obtained from this simple computation. First of all, probability of selecting the maximum of the sample increases nearly linearly with n in the case of large populations. In fact the probability of selecting the maximum in a population of size N and a $(n, 1)$-tournament with the rounding procedure described above is given by :

$$1 - \left(1 - \frac{1}{2N}\right)^n .$$

which can be expanded as

$$\frac{n}{2N} + o(N^{-1}) .$$

Second, in the case of a general (n, m)-tournament and if m is large (thus n), the selection of last individuals is close to uniform sampling (conditionally to the least index value obtained so far). A survey of properties of tournament selection may be found in [Bli97].

3.2 Limiting Distributions in Tournament Selection

As most evolutionary operators, selection may be viewed not as an operator evolving individuals but probability distributions [DMPJ01]. The most attractive feature about that is the ability to use classical convergence proof (fixed point theorems for example) instead of complicated probabilistic arguments. From that point of view, an evolutionary algorithm is a measure valued dynamical systems, which evolves empirical (that is to say sum of point distributions) measures. It may be noted that a weak law of large numbers exists within this frame, when the population size go to infinity. Our purpose is to restrict our attention to the tournament selection operator and to show how to use results on the asymptotic law of extremes. Let E be a Banach space (which will be $\mathbb{R}^n$ in our application), $\mathcal{B}(E)$ its Borel field and let $\mathcal{P}(E)$ the set of probability measures on E. The total variation distance on $\mathcal{P}(E)$ is defined by:

$$d(\mu, \nu) = \|\mu - \nu\| = \sup_{A \in \mathcal{B}(E)} |\mu(A) - \nu(A)| .$$

In the case of densities (with respect to the Haar measure on E), and confusing the notation of density and measure, we have the well known equality:

$$\|\mu - \nu\| = \frac{1}{2} \int_E |\mu(x) - \nu(x)| dx.$$

Now, let μ be a density and let $f : E \to \mathbb{R}$ be the criterion to be maximized that will be assumed of being twice continuously differentiable. The density of the best individual on a tournament of size n is given by:

$$n\mu(x)\mu\left(\{u | f(u) < f(x)\}\right)^{n-1} .$$

Let ϕ be the operator on probability measures associated with tournament selection. Taking densities μ, ν, we have:

$$\|\phi(\mu) - \phi(\nu)\| = \frac{1}{2} \int_E |\mu(x)\mu(L_{f(x)})^{n-1} - \nu(x)\nu(L_{f(x)})^{n-1}| dx$$

with $L_{f(x)} = \{u \in E | f(u) < f(x)\}$ the sub-level set of f on value $f(x)$. We can then write :

$$\begin{aligned}
&\|\phi(\mu) - \phi(\nu)\| \\
&\quad = \frac{1}{2} \int_E |(\mu(x) - \nu(x))\mu(L_{f(x)})^{n-1} - \nu(x)(\nu(L_{f(x)})^{n-1} - \mu(L_{f(x)})^{n-1})| dx \\
&\quad \leq \frac{1}{2} \int_E |\mu(x) - \nu(x)| dx + \frac{1}{2} \int_E \nu(x) |\nu(L_{f(x)})^{n-1} - \mu(L_{f(x)})^{n-1}| dx
\end{aligned}$$

Now, since$\mu(L_{f(x)})^{n-1} \leq 1$ and $\nu(L_{f(x)})^{n-1} \leq 1$, we get

$$|\nu(L_{f(x)})^{n-1} - \mu(L_{f(x)})^{n-1})| \leq (n-1)|\nu(L_{f(x)})^{n-1} - \mu(L_{f(x)})^{n-1}| \leq (n-1)\|\mu - \nu\| .$$

Finally

$$\|\phi(\mu) - \phi(\nu)\| \leq \frac{n+1}{2} \|\mu - \nu\| ,$$

so that ϕ is Lipschitz.

From now on we assume that the set of critical points of f is finite and f reaches its maximum. Let $\mu \in \mathcal{P}(E)$; the $\phi\mu$-measure of $A \in \mathcal{B}(E)$ can be computed:

$$\begin{aligned}
\phi\nu(A) &= n \int_A \mu(x)\mu(L_{f(x)})^{n-1} dx \\
&= n \int_{-\infty}^{M} \mu(L_t)^{n-1} \int_{\partial L_t \cap A} \|f'(z)\|^{-1} \mu(z) dvol_{L_t}(z) dt ,
\end{aligned}$$

where M is the maximum of f and $dvol_{L_t}$ is the canonical volume form on the level set L_t. To analyse further the behaviour of this measure, assume first that μ belongs to the domain of attraction of the weibull law, that is to say there exists a sequence (a_n) such that

$$\lim_{n \to +\infty} \mu(L_{M + a_n t})^n = e^{-(-t)^\alpha}, \, t < 0 ,$$

and assume that f has a unique maximum at x_0, and is λ-convex. Let A be limited by a level set at the value t_0. Then, for n large enough, the $\phi\mu$ measure of A may be approximated by

$$I(A) = n \int_{t_0}^{M} e^{-\theta_{n-1}(M-t)^\alpha} \int_{\partial L_t \cap A} \|f'(z)\|^{-1} \mu(z) dvol_{L_t}(z) dt$$

with $\theta_n > 0$. Since f is C^2 and λ-convex, there exists $K > 0$ such that outside L_{t_0} those inequalities hold:

$$\lambda \|x - x_0\| \leq \|f'(x)\| \leq K \|x - x_0\| .$$

Therefore

$$K^{-1}n\int_{t_0}^{M} e^{-\theta_n(M-t)^\alpha}\int_{\partial L_t\cap A}\|z-x_0\|^{-1}\mu(z)dvol_{L_t}(z)dt \le I(A),$$
$$I(A) \le \lambda^{-1}n\int_{t_0}^{M} e^{-\theta_n(M-t)^\alpha}\int_{\partial L_t\cap A}\|z-x_0\|^{-1}\mu(z)dvol_{L_t}(z)dt\,.$$

4 Algorithm

Asymptotic distribution of order statistics allows some statistical inference about the value of the true maximum of a function given a sorted sample. Furthermore, rank based selection process like tournament naturally uses order statistics since relative value of individuals are used. To demonstrate the interest of statistical inference based on extreme values distribution, we have modified a standard tournament selection based on genetic algorithm so that sample values that may be considered as extremes will be specially treated.

4.1 Tail Index Estimation

Some runs of the standard GA has been done in order to test for adjustment of the distribution of population order statistics to a Weibull law.Since empirical distribution function can be easily computed from the sorted population the Kolmogorov-Smirnov test was used . For most generations, conformance of the upper order statistics to the asymptotic law is accepted. However, during some transition phases this property is lost and upper values no longer obey to a Weibull law. Hill tail index estimator yields values that have the same order of magnitude than the theoretical value for a C^2 criterion, but variation is high from a generation to another.

Maximum-likelihood estimation of both the tail index and the essential supremum of the criterion has been tried too, but increased computational cost (implicit equation solving) is a major drawback. Furthermore, tail index obtained by this procedure is close to the Hill estimator.

4.2 Genetic Operators on Extreme Values

It is easy to see that uniform real crossover is in fact uniform sampling in the hyper-rectangle defined by the two parents. That observation shows that some speed-up in convergence may be obtained by restricting the use of uniform crossover to individuals representing extreme values of the population. The only remaining point is to defined which values may be considered as extreme. The problem has been addressed in [GO03] but the solution found is computationally expensive (successive Kolmogorov-Smirnov goodness of fit tests). Following the same idea, we make adjustment tests but starting from the upper fifth distinct values of the population the increase by a fixed amount until KS test failed. We enforce that the number of tests be under some given number.

5 Results

Different test functions have been used in order to compare our method with a classical GA :

Sphere	$f_1(\boldsymbol{x}) = \sum_{i=1}^{i=N} x_i^2$	$-10000 \leq x_i \leq 10000$
Ackley	$f_2(\boldsymbol{x}) = -c_1.\exp(S_1(\boldsymbol{x})) - \exp(S_2(\boldsymbol{x})) + c_1 + e$ with $S_1 = -c_2\sqrt{\frac{1}{N}\sum_{i=1}^{N} x_i^2}$, $S_2 = \frac{1}{N}\sum_{i=1}^{N}\cos(c_3.x_i)$, $c_1 = 20\ c_2 = 0.2\ c_3 = 2\pi$	$-10000 \leq x_i \leq 10000$
Griewank	$f_3(\boldsymbol{x}) = \frac{1}{400*N}\sum_{i=1}^{N} x_i^2 - \prod_{i=1}^{N}\cos(\frac{x_i}{\sqrt{i}}) + 1$	$-10000 \leq x_i \leq 10000$
Rosenbrook	$f_4(\boldsymbol{x}) = \sum_{i=0}^{N-1} 100*(x_i^2 - x_{i+1})^2 + (1 - x_i)^2$	$-30 \leq x_i \leq 30$

All the functions have to be minimized and have their minimum at 0 unless the Lenard-Jones function for which only an experimental mininum is used (best known min=-128.287 for 30 atoms). It must be noticed that both algorithms use the same selection scheme (stochastic remainder without replacement which is not the best) and do not use any scaling or sharing operators.

Our goal being to compare the influence of domain chromosome and order statistics we wanted them to work exactly the same way from the selection point of view.

The number of evaluations being different at each generation for those two algorithms, the number of generations has been adapted in order to maintain the same number of evaluations for all experiments.

Notice that the following curves have been adjusted in order to represent both results on the same graph. Those adjustments have been done on both axes. The "x" axis address the number of evaluations for our GA and must be scaled for the standard GA ($\times$ 20). The "y" axis represent the fitness given by both algorithms. The given results are so different that a logarithm scale has been used to see both curves.

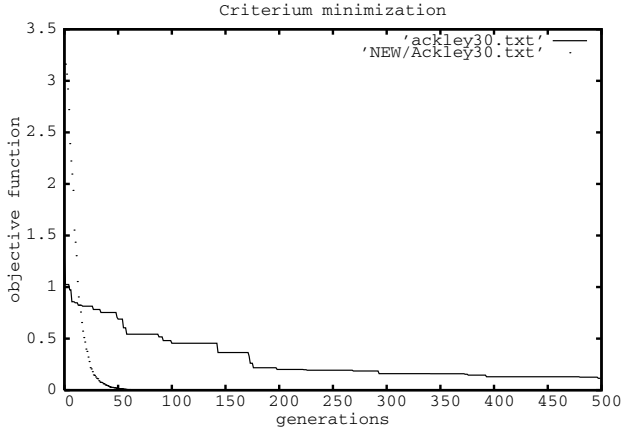
The parameters used for our GA are the following:

individuals 100	generations 500
probability of crossover 0.4	probability of mutation 0.3

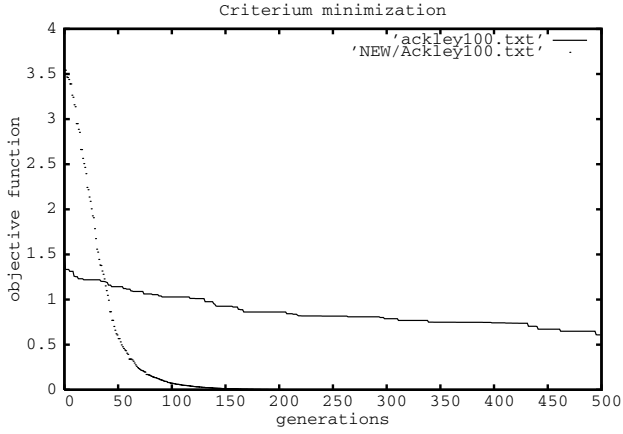
For the Rosenbrook, Lennard-Jones the number of generations has been extended to 1500 and 2500 respectively. The experiments have been done on a PentiumII 300 MHz and last 7 minutes for N=200 and 14 minutes for N=2000 (N is the dimension of the state space). It must be noticed that other experiments has been done for the same functions with the optimum moved in the state space (without symmetries) and the given results are quite similar.

Function	f_1	f_2	f_3	f_4
Standard AG - N=200	10621	11598	11.98	20.11
Domain AG - N=200	2.28	0	0.32	0.96
Standard AG - N=2000	910^5	8.710^5	20.45	165.8
Domain AG - N=2000	622	222	3.38	1.11

Function	f_5 N=200	f_6 N=90 (30 Atoms)
Standard AG	15.610^6	-77.21
Domain AG	254	-125.9 [1]

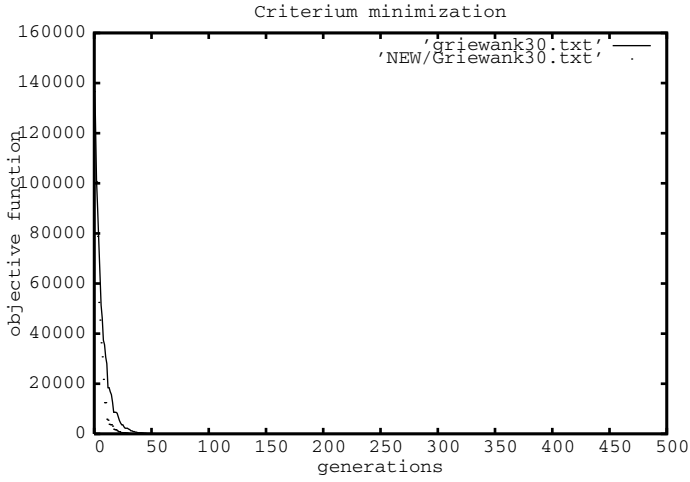


(a) Ackley Dimension 30

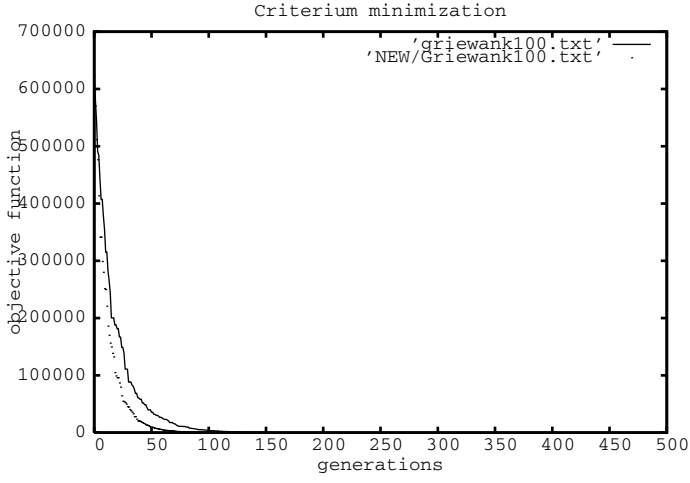


(b) Ackley Dimension 100

Fig. 1. Objective evolution for the Ackley function



(a) Griewank Dimension 30

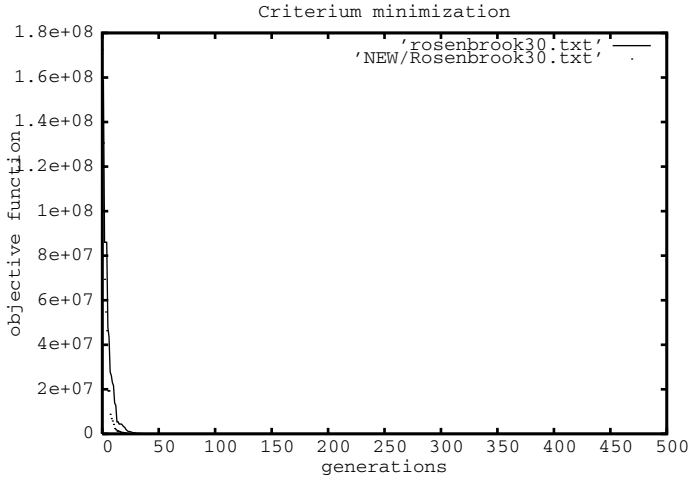


(b) Griewank Dimension 100

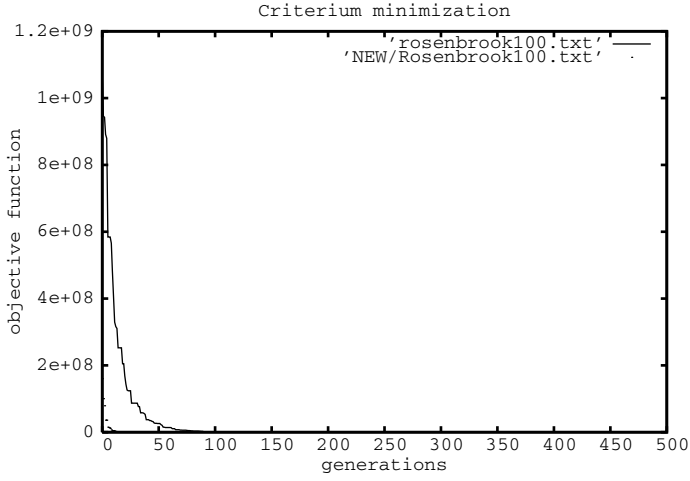
Fig. 2. Objective evolution for the Griewank function

6 Conclusion

This paper shows the gain given by the mix of extreme values inference and artificial evolution. On one side, the main advantage of order statistics for optimization is their abilities to summarize the properties of an entire domain with a "small" sample. On the other side, the evolution process of GA is able to build



(a) Rosenbrook Dimension 30



(b) Rosenbrook Dimension 100

Fig. 3. Objective evolution for the Rosenbrook function

the most adapted chromosome to environment given by the fitness landscape. The mixing of both methods really increase the performances of GA by guiding the exploration and exploitation phases. For all tests, results produced by this new GA, are better than those given by a classical GA.

Notice that this algorithm may be still improved in the following way:

- use of a better selection scheme;
- the order statistics may control the drawing of individuals;

– pools of samples may be stored to reduce the number of functions of evaluation.

References

[Bli97] T. Blickle. *Handbook of Evolutionary Computation*, chapter Tournament selection. IOP Publishing Ltd., 1997.

[dH81] L. de Haan. Estimation of the minimum of a function using order statistics. *Journal of the American Statistical Association*, 1981.

[DMPJ01] Kallel L. Del Moral P. and Rowe J. Modeling genetic algorithms with interacting particle systems. *Revista de Matematica, Teoria y aplicaciones*, 2001.

[GO03] J. GONZALO and J. OLMO. Which extreme values are really extremes. *Preprint*, 2003.

[Hil75] B.M. Hill. A simple approach to inference about the tail of a distribution. *Annals of Statistics*, 1975.

[Pic75] J. Pickhands. Statistical inference using extreme order statistics. *Annals of Statistics*, 1975.

[Rei89] R.D. Reiss. *Approximate distributions of order statistics*. Springer-Verlag, 1989.

[Zhi91] A.A. Zhighljavsky. *Theory of random search*. Kluwer Academic, 1991.

Evolutionary Markov Chain Monte Carlo

Mădălina M. Drugan and Dirk Thierens

Institute of Information and Computing Sciences, Utrecht University
P.O. Box 80.089, 3508 TB Utrecht, The Netherlands
{madalina,dirk}@cs.uu.nl

Abstract. *Markov chain Monte Carlo* (MCMC) is a popular class of algorithms to sample from a complex distribution. A key issue in the design of MCMC algorithms is to improve the proposal mechanism and the mixing behaviour. This has led some authors to propose the use of a population of MCMC chains, while others go even further by integrating techniques from evolutionary computation (EC) into the MCMC framework. This merging of MCMC and EC leads to a class of algorithms, we call *Evolutionary Markov Chain Monte Carlo* (EMCMC). In this paper we first survey existing EMCMC algorithms and categorise them in two classes: *family-competitive EMCMC* and *population-driven EMCMC*. Next, we introduce the *Elitist Coupled Acceptance* rule and the *Fitness Ordered Tempering* algorithm.

1 Introduction

Markov Chain Monte Carlo (MCMC) algorithms provide a framework for sampling from complicated target distributions that cannot be sampled with simpler, distribution specific, methods. MCMC algorithms are applied in many fields, and their use in Machine Learning has recently been advocated in [1]. Usually, MCMC uses a single chain which runs for a long time. However, to improve the convergence rate, there are some MCMC variants that work with a population of MCMCs. The use of a population makes these algorithms somewhat similar to Evolutionary Algorithms (EA). Indeed some authors have proposed algorithms that integrate techniques from the EC field into the MCMC framework. Here we survey these EMCMC algorithms and classify them into two basic categories: *family-competitive EMCMC* algorithms that operate through an acceptance rule at the family level, and *population-driven EMCMC* algorithms that operate through a population-driven, adaptive proposal distribution. One property of EMCMC algorithms is that they are not necessarily a set of parallel MCMC chains, but that they are a single MCMC at the population level. Besides surveying existing EMCMC algorithms we also propose two alternative techniques: the *Elitist Coupled Acceptance* (ECA) rule and the *Fitness Ordered Tempering* (FOT) algorithm.

The paper is organised as follows. Section 2 discusses basic concepts and algorithms from MCMC. Section 3 describes parallel MCMC algorithms, while Section 4 surveys EMCMC algorithms. We introduce the ECA and FOT techniques in Section 5, and report some experimental results in Section 6.

P. Liardet et al. (Eds.): EA 2003, LNCS 2936, pp. 63–76, 2004.

2 The Markov Chain Monte Carlo Framework

MCMC is a general framework to generate samples X_t from a probability distribution $P(\cdot)$ while exploring its search space $\Omega(X)$ using a *Markov chain*. MCMC does not sample directly from $P(\cdot)$ but only requires that the density $P(X)$ can be evaluated within a multiplicative constant $P(X) = P'(X)/Z$, where Z is a normalisation constant and $P'(\cdot)$ is the unnormalized target distribution. A Markov chain is a discrete-time stochastic process $\{X_0, X_1, \ldots\}$ with the property that the state X_t given all previous values $\{X_0, X_1, \ldots, X_{t-1}\}$ only depends on X_{t-1}: $P(X_t \mid X_0, X_1, \ldots, X_{t-1}) = P(X_t \mid X_{t-1})$. We call $P(\cdot \mid \cdot)$ the transition matrix of the Markov chain. $P(\cdot \mid \cdot)$ is a *stationary* - this is, independent of time t - *transition matrix* with the following properties: (i) all the entries are non-negative, and (ii) the sum of the entries in a row is 1. We assume that $P(\cdot) > 0$. MCMC converges, in infinite time, to the probability distribution $P(\cdot)$, thus it samples with higher probability from more important states of $P(\cdot)$. An finite state MCMC which has an *irreducible* and *aperiodic* stationary transition matrix converges to a unique *stationary distribution* [1]. A MCMC chain is irreducible if, and only if, every state of the MCMC chain can be reached from every other state in several steps. A MCMC is aperiodic if, and only if, there exists no cycles to be trapped into. A sufficient, but not necessary, condition to ensure that $P(\cdot)$ is the stationary distribution is that MCMC satisfies the *detailed balance condition* [1]. A MCMC satisfies the detailed balance condition if, and only if, the probability to move from X_t to X_{NEW} multiplied by the probability to be in X_t is equal to the probability to move from X_{NEW} to X_t multiplied by the probability to be in X_{NEW}: $P(X_{NEW} \mid X_t) \cdot P(X_t) = P(X_t \mid X_{NEW}) \cdot P(X_{NEW})$.

Metropolis-Hastings algorithms. Many MCMC algorithms are Metropolis-Hastings (MH) algorithms [2,3]. Since we cannot sample directly from $P(\cdot)$, MH algorithms consider a simpler distribution $S(\cdot \mid \cdot)$, called the *proposal distribution* for sampling the next state of a MCMC chain. $S(X_{NEW} \mid X_t)$ generates the candidate state X_{NEW} from the current state X_t, and the new state X_{NEW} is accepted with probability:

$$A(X_{NEW} \mid X_t) = \min\left(1, \frac{P'(X_{NEW}) \cdot S(X_t \mid X_{NEW})}{P'(X_t) \cdot S(X_{NEW} \mid X_t)}\right).$$

If the candidate state is accepted the next state becomes $X_{t+1} = X_{NEW}$. Otherwise, $X_{t+1} = X_t$. The transition probability for arriving in X_{NEW} when the current state is X_t is $T(X_{NEW} \mid X_t) = S(X_{NEW} \mid X_t) \cdot A(X_{NEW} \mid X_t)$, if $X_{NEW} \neq X_t$, and $T(X_t \mid X_t) = 1 - \sum_{Y, Y \neq X_t} S(Y \mid X_t) \cdot A(Y \mid X_t)$, otherwise.

A MH algorithm is aperiodic, since the chain can remain in the same state with a probability greater than 0, and by construction it satisfies the detailed balance condition. If, in addition, the chain is irreducible, then it converges to the stationary distribution $P(\cdot)$. The rate of convergence depends on the relationship between the proposal distribution and the target distribution: the closer the proposal distribution is to the stationary distribution, the faster the

chain converges. Two popular Metropolis-Hastings algorithms are the *Metropolis algorithm* and the *independence sampler*. For the Metropolis algorithm, the proposal distribution is symmetrical $S(X_{NEW} \mid X_t) = S(X_t \mid X_{NEW})$ and the acceptance rule becomes $A(X_{NEW} \mid X_t) = \min(1, \frac{P'(X_{NEW})}{P'(X_t)})$. Because it accepts the candidate states often - and thus the state space is well sampled - the Metropolis rule generally performs well. The proposal distribution of the independence sampler does not depend on the current state $S(X_{NEW} \mid X_t) = S(X_{NEW})$. The independence sample's acceptance probability can be written as $A(X_{NEW} \mid X_t) = \min(1, w(X_{NEW})/w(X_t))$, where $w(\cdot) = P'(\cdot)/S(\cdot)$. Candidate states with low $w(X_{NEW})$ are rarely accepted, while states with high $w(X_{NEW})$ are very often accepted, and the process could get stuck for a long time in states with very high $w(X)$. Obviously, the choice of $w(\cdot)$ greatly influences the convergence rate.

Simulated annealing (SA). SA is a minor modification of a single chain MH algorithm used for optimisation. Instead of sampling from the entire distribution $P(\cdot)$, SA samples at step t from $P'_t(\cdot) = P'(\cdot)^{\frac{1}{Temp[t]}}$, where $Temp[t]$ decreases according to a cooling scheduler to 0. With $Temp[\cdot]$ close to ∞, the chain accepts almost any candidate state according to MH acceptance rule A, whereas, when $Temp[\cdot]$ is close to 0, the chain rejects almost all states that have lower fitness than the current one. Note that, for constant temperature $Temp[t]$, SA is a MCMC which converges to the distribution $P_t(\cdot)$. However, every time the SA chain is cooled, the transition matrix is changed and the detailed balance is not satisfied. Yet, in infinite time, SA converges to the optimum and, more general, if $Temp[i]$ decreases to 1 SA converges to the stationary distribution $P(\cdot)$. SA is a *non-homogeneous MCMC* which converges to a given stationary distribution. The time to convergence depends on the cooling schedule. In practice, a fast cooling schedule is preferred to a slower one, increasing the risk of poor performance.

3 Parallel MCMC Algorithms

MCMC algorithms are usually applied in a sequential way. A single chain is run for a long time until it converges to the stationary distribution $P(\cdot)$. The states visited during the initial phase of the run are considered to be unreliable and further ignored. For reasons of computational efficiency this burn-in phase should be as short as possible. MCMCs with a short burn-in phase are said to mix well (note that this is unrelated to the mixing of building blocks in the EC literature). There exist various variations on the standard MCMC algorithm to speed up this mixing process. In this paper we are particularly interested in techniques that use multiple chains in parallel as opposed to a single chain.

Multiple independent chains (MIC). The most straightforward technique to make use of multiple chains is simply to run N independent MCMCs in parallel. The chains are started at different initial states and their output is observed at the same time. It is hoped that this way a more reliable sampling of the

target distribution $P(\cdot)$ is obtained. It is important to note that no information exchange between the chains is taking place. Recommendations in the literature are conflicting regarding the efficiency of parallel independent chains. Yet there are at least theoretical advantages of multiple independent chains MCMC for establishing its convergence to $P(\cdot)$ [4].

Parallel tempering (PT). Parallel tempering [5] is a parallel MCMC with N chains each having a different stationary distribution $P'_i(\cdot) = P'(\cdot)^{\frac{1}{Temp[i]}}$, $i = 1, \ldots, N$. The temperatures have an increasing magnitude $Temp[1] < \ldots < Temp[N]$ with $Temp[1] = 1$. The stationary distribution of the lowest temperature chain is therefore equal to the target distribution, or $P'_1(\cdot) = P'(\cdot)$. The temperatures $Temp[i], (2 \leq i \leq N)$ are given a constant value, typically according to a geometrically increasing series. Note that this is similar to the temperature values proposed by the cooling scheduler of simulated annealing, though for SA the different temperatures are generated sequentially in time, while for PT the different temperatures are generated in space - this is, the population - and remain unchanged during the run.

The candidate states are generated using mutation and accepted with the standard Metropolis-Hasting acceptance rule. Chains in PT exchange information by swapping states. Two chains i and j interact by trying to exchange their current states $X_t[i]$ and $X_t[j]$ using the *swapping acceptance rule*:

$$A_S(X_t[i], X_t[j]) = \min\left(1, \frac{P'_i(X_t[j]) \cdot P'_j(X_t[i])}{P'_i(X_t[i]) \cdot P'_j(X_t[j])} \cdot \frac{S(X'_t \mid X''_t)}{S(X''_t \mid X'_t)}\right)$$

where $X'_t = (\ldots, X_t[i], \ldots, X_t[j], \ldots)$ and $X''_t = (\ldots, X_t[j], \ldots, X_t[i], \ldots)$. Note that A_S is a MH acceptance rule, satisfying the detailed balance condition; A_S accepts with probability 1 an exchange of states if the more important state is inserted into the chain with the lower temperature. To increase the acceptance rate the two chains usually have adjacent temperatures ($|i-j| = 1$). Heuristically, PT improves mixing: better states of a warmer chain can be inserted in a colder chain that is mixing slower.

Parallel sintering (PS). Parallel sintering [6] can be viewed as a generalisation of parallel tempering where the proposal distributions of each chain in the population is a member of some family of distributions $\{P_i(\cdot) \mid i = 1, \ldots, N\}$ defined over spaces of different dimensions (resolutions), that are related to the target distribution by the highest dimensional distribution (or largest resolution) $P_1(\cdot) = P(\cdot)$. As in PT parallel sintering exchanges information between adjacent chains by exchanging states through the swapping acceptance rule. Here too the idea is to increase the mixing of the slowly mixing highest dimensional chain towards the target distribution by inserting states of the lower dimensional - and faster mixing - chains in the population. Parallel sintering is a multiple chain MCMC version of the single chain MCMC called simulated sintering [7].

4 Evolutionary MCMC Algorithms

In the previous section, we have outlined parallel MCMCs. In the following we survey existing EMCMC algorithms, and distinguish between two categories: *family-competitive EMCMC* and *population-driven EMCMC*.

Family-competitive EMCMC algorithms let two chains from the population exchange information to sample two new states. This interaction can be implemented by the proposal mechanism and/or by the acceptance rule. Recombining the states of the two chains is an example of the former approach, while in the latter two states are selected from the two proposed states and their 'parent' states. We call any MH acceptance rule which selects two states from the family of four states a *coupled acceptance rule*. Note that in this view parallel tempering (and parallel sintering) belong to the class of family-competitive EMCMC algorithms. Family-competitive EMCMC algorithms correspond to family-competitive EAs where two individuals create offspring by mutation and recombination, and selection takes places at this family level - examples are the elitist recombination GA [8], the deterministic crowding GA [9], the genetic invariance GA [10].

Population-driven EMCMC algorithms adapt their proposal distribution according to the entire, current population of states. The adaptation is such that the algorithm maintains a single MCMC at the population level - this is, a MCMC whose states are populations. Population-driven EMCMC algorithms correspond to the probabilistic model-building evolutionary algorithms in the sense that new solutions are proposed on the basis of information taken from the entire, current population. In the probabilistic model-building evolutionary algorithms a distribution is learned from the current population, and offspring is generated by sampling from this distribution [11].

Table 1 compares the EMCMC algorithms surveyed here according to their perturbation operators, their methods of the communication between chains, their EMCMC category, and their use as a sampler or optimizer. Call $X_t = (X_t[1], \ldots, X_t[N])$ the population at time t of N states (or individuals) $X_t[\cdot]$. To each individual $X[\cdot]$ we associate a *fitness function* $f : \Omega(X[\cdot]) \to \mathbb{R}$, where $X[\cdot]$ samples from $P'(X[\cdot]) = g(f(X[\cdot]))$ (g is a monotonic function, $g : \mathbb{R} \to \mathbb{R}^+ \setminus \{0\}$). Here, we consider an individual $X[\cdot]$ as a string of l characters $X[\cdot][1]X[\cdot][2] \ldots X[\cdot][l]$. We use $X[\cdot][\cdot]^a$ to denote the $a-$th value $\Omega(X[\cdot][\cdot])$, the set of all possible values of $X[\cdot][\cdot]$. We call $X[\cdot][j]^a$ an *allele* of $X[\cdot][j]$. The position of $X[\cdot][j]$ in $X[\cdot]$ is called the *locus* of $X[\cdot][j]$. X'_t, X''_t are two intermediate populations.

Evolutionary Monte Carlo (EMC). Liang and Wong [12], [13] propose the evolutionary Monte Carlo algorithm, which incorporates recombination into the parallel tempering (PT) algorithm to speed up the search and preserve good building blocks of the current states of the chains. Like PT, EMC has a population of MCMC chains with constant and (geometrically) increasing temperatures. Chains interact through the swapping acceptance rule A_S that attempts to exchange states between chains with adjacent temperature. This way, the good individuals sampled in the warmer chain are transferred to a colder chain

Table 1. Comparison of the presented EMCMC algorithms

algorithm	perturbation op.	communication	type EMCMC optim. / sampl.(distrib.)	
EMC	mut, recomb	A_C, A_S	fam-comp	$\prod_{i=1}^{N} P_i'(\cdot)$
MRGA	mut, recomb	A_C, A_S	fam-comp	$\prod_{i=1}^{N} P_i'(\cdot)$
popSA		Boltzmann trials	fam-comp	maxim
PRSA	mut, recomb	A_C	fam-comp	maxim
mparSA	mut, neigh. recomb	neighbours	fam comp	maxim
popMCMC	mut	$S(\cdot \mid \cdot)$	pop-driven	$\prod_{i=1}^{N} P'(\cdot)$
eMCMC	mut, recomb	recomb, $S(\cdot \mid \cdot)$	pop-driven	maxim
ECA	mut, recomb	ECA	fam-comp	$\prod_{i=1}^{N} R'(\cdot)$

where they may be preserved longer. The lowest temperature chain $Temp[1] = 1$ converges to the target distribution $P(\cdot)$, where $P_i'(\cdot) = \exp(\frac{-f(X[i])}{1})$.

The candidate states are generated in two different ways and accepted by two different rules. With probability q_m each chain in the population generates a new state by mutation and accepts it with the standard Metropolis-Hastings acceptance rule. With probability $1 - q_m$ new states are generated by recombination of two states from different chains, and the offspring states are accepted with the *coupled acceptance rule*:

$$A_C((X_{NEW}[i], X_{NEW}[j]) \mid (X_t[i], X_t[j])) = \min(1, \frac{P_i'(X_{NEW}[i]) \cdot P_j'(X_{NEW}[j])}{P_i'(X_t[i]) \cdot P_j'(X_t[j])} \cdot \frac{S(X_t' \mid X_{NEW}')}{S(X_{NEW}' \mid X_t')}),$$

with $X_t' = (\ldots, X_t[i], \ldots, X_t[j], \ldots)$, $X_{NEW}' = (\ldots, X_{NEW}[i], \ldots, X_{NEW}[j], \ldots)$. Note that when $q_m = 1$ EMC reduces to PT. Liang and Wong discussed experimental results for a model selection problem, a time series change-point identification problem, and for a protein folding problem. These experiments showed the effectiveness of EMC as compared to PT.

Multi-resolution Genetic Algorithm-style MCMC (MRGA). Holloman, Lee and Higdon [6] also proposed to integrate recombination in their multi-resolution parallel sintering algorithm. MRGA runs several chains for each resolution, sampling from the distribution at that resolution. Each individual has: (i) a variable size part, which represents information specific to its resolution, updated with a MCMC and (ii) a fixed dimensional (e.g. lowest dimension) vector with common interpretability used to move between chains (e.g. for image processing this could be the mean of neighbourhood cells). MRGA samples at the individual level using mutation, while it samples at the population level using recombination between the fixed dimensionality vector of two chains of the same or different resolutions. Both new individuals are accepted/rejected using the swapping acceptance rule A_S, which, in this case is equivalent with the coupled acceptance rule A_C. Like in parallel sintering, to improve mixing, chains of different resolutions periodically fully exchange the fixed dimensionality vectors

using the swapping acceptance rule A_S. MRGA proposes with probability p_{swap} a full exchange, or with probability $1 - p_{swap}$ a crossover exchange.

Population MCMC (popMCMC). Laskey and Myers [14] introduced popMCMC where a population of MCMC chains exchange information through a population-based, adaptive proposal distribution. As before the Boltzmann distribution is used: $P'(X_t[\cdot]) = \exp^{-f(X_t[\cdot])}$,where f is a fitness function. Each generation a locus $X_t[i][j]$ is randomly picked to be mutated. An allele $X_t[i][j]^g$ on the j-th locus of the candidate individual $X_{NEW}[i]$ is generated using a distribution $\hat{\alpha}_{ija} = \frac{N(X_t[i][j]^a)+1}{N+|\Omega(X[\cdot][\cdot])|}$, where $N(X_t[i][j]^a)$ is the number of individuals that have the allele $X_t[i][j]^a$ on the j-th locus and $|\Omega(X[\cdot][\cdot])|$ is the total number of alleles. $X_{NEW}[i]$ is accepted using the MH acceptance rule $A(X_{NEW}[i] \mid X_t[i])$.

Since the proposal distribution depends on the current population, the transition matrix of a single individual is not stationary, yet the transition matrix of the whole population is stationary. Whenever good individuals from the population have a specific allele on a certain locus, new individuals will have with high probability the same allele on the same locus. Since popMCMC is a population of independently sampling MCMC chains, the allele remains in the population for a while. It is interesting to observe the similarity between popMCMC and univariate EDAs [11] in generating the candidate individuals from the structure of the current population. PopMCMC converges to the stationary distribution of N independently sampled points from $P(\cdot)$. The authors show experimentally that popMCMC finds highly likely Bayesian network structures faster than multiple independent MCMC chains.

Evolutionary MCMC (eMCMC). Zhang and Cho [15] proposed the eMCMC algorithm which is designed to find the optimum of a distribution by generating L samples from a population of N MCMC chains and then selecting the best N states of them ($L > N$). The initial population is sampled according to a prior Gaussian distribution. Each generation, each chain from the current population generates several new candidate individuals using mutation and recombination which are accepted according to a Metropolis acceptance rule $A(\cdot \mid \cdot)$. The best N individuals from the L individuals are then selected for the next generation. Their posterior distribution - estimated with a Gaussian distribution model - represents the proposal distribution $S(\cdot \mid \cdot)$ for the next generation. We observe that eMCMC is also related with EDA [11] since the candidate individuals are generated from $S(\cdot \mid \cdot)$ which is adapted each generation. The eMCMC algorithm is an EMCMC algorithm which samples using mutation and recombination and where $S(\cdot \mid \cdot)$ is adapted each generation to sample from promising regions of the target distribution. Experimentally, eMCMC outperformed single chain MCMC and the standard GA for a system identification task.

Population-based simulated annealing (popSA). Goldberg [16] proposed a population-based simulated annealing algorithm which creates a Boltzmann distribution over the population. Instead of MH acceptance rule, it uses the *logistic acceptance rule* which has the probability to accept a candidate individual: $1/(1 + \exp \frac{f(X_t[\cdot])-f(X_{NEW}[\cdot])}{Temp[t]})$, where $Temp[t]$ is the temperature for generation

t. When $Temp[t]$ is constant, the unnormalized stationary distribution of this algorithm is the Boltzmann distribution $P'_t(X_t[\cdot]) = \exp \frac{f(X_t[\cdot])}{Temp[t]}$, where f is the fitness function. Each individual $X_t[i]$ from the current population chooses two other individuals from the population for coupling: one $X_t[j]$ has a fitness value different from $X_t[i]$ by a threshold θ and one $X_t[k]$ has a fitness value different from $X_t[i]$ or from both $X_t[i]$ and $X_t[j]$ by a threshold θ. An anti-acceptance competition is held between $X_t[j]$ and $X_t[k]$, where $X_t[j]$ is accepted with probability $1/(1 + \exp \frac{f(X_t[j])-f(X_t[k])}{Temp[t]})$. The primary acceptance competition is held between the winner of the first competition $X_t[jk]$ and $X_t[i]$, where $X_t[i]$ is accepted with probability $1/(1 + \exp \frac{f(X_t[jk])-f(X_t[i])}{Temp[t]})$. The competitions are chosen to avoid the danger that copies of an individual are taking over the population. The anti-acceptance rule prefers poorer individuals, whereas the primary acceptance rule prefers the better individuals. This way, the population equivalent is created to generate a neighbour of the population at random.

Parallel recombinative simulated annealing (PRSA). Mahfoud and Goldberg [17] proposed a population-based simulated annealing algorithm which also made use of recombination. All individuals from the population have the same temperature which decreases every generation according to a cooling schedule. New individuals are generated using mutation and one point recombination between two individuals $X_t[i]$ and $X_t[j]$. PRSA uses the logistic acceptance rule to accept a candidate individual. Two possible competitions are considered: single acceptance/rejection holds two competitions between a parent vs. the child formed from its own right-end and the other parent left-end, or double acceptance/rejection holds one competition between both parents vs. both children using the coupled acceptance rule:

$$A_C((X_{NEW}[i], X_{NEW}[j]) \mid (X_t[i], X_t[j])) = \min(1, 1/(1 + \exp \frac{f(X_t[i])+f(X_t[j])-f(X_{NEW}[i])-f(X_{NEW}[j])}{Temp[t]}))$$

Massively parallel simulated annealing (mparSA). Rudolph [18] introduced the massively parallel simulated annealing algorithm. He experimentally achieved fast convergence to the optimum with a population of independent SA chains. Like SA, mparSA uses the Boltzmann function $P'_t(X_t[\cdot]) = \exp(-\frac{f(X_t[\cdot])}{Temp[t]})$, where f is a fitness function and $Temp[t]$ is the temperature for all individuals from the t-th generation. The algorithm associates the population with a connected graph. Each node has an individual which communicates with the individuals in the neighbouring nodes in the graph. For each chain, each step, the current state $X_t[i]$ is recombined with a neighbour and the result is mutated several times obtaining new neighbours. The best candidate individual $X_{NEW}[i]$ is then selected and is accepted with the MH acceptance rule $A(X_{NEW}[i] \mid X_t[i])$. The temperature is decreased each step according to a SA cooling scheduler. Rudolph pointed out that the conflicting goals of fast convergence and global convergence to the optimum can be satisfied with an adaptive proposal distribution, whereas it cannot with a fixed proposal distribution. As in

Evolutionary Strategies, mparSA uses a non-fixed proposal distribution $S_t(\cdot \mid \cdot)$ which is adapted each generation with a lognormally distributed random variable α_t: $S_t(\cdot \mid X_t[i]) = \alpha_t S_{t-1}(\cdot \mid X_t[i])$.

Related work. Cercueil and Francois [19] give an overview of literature where EAs are viewed as Monte Carlo methods which generates sample from a probability distribution defined on the trajectories of their population. This helps to unify the convergence theories for EAs. For doing that, Cerf [20] has a GA with the mutation rate related to a temperature, no crossover, and Boltzmann roulette selection. Each generation, an intermediate population is generated by mutation. The next population is generated from a Boltzmann distribution constructed from the intermediate population. Lozano and co-authors discuss a hybrid between the genetic algorithm and simulated annealing based on a probabilistic Boltzmann reduction operator [21]. In a very recent publication Strens proposes an EMCMC algorithm with an 'exclusive-or' proposal operator [22]. This operator takes one parent and two reference states, and generates an offspring state that disagrees from its parent in a similar way as the two reference states.

Sequential Monte Carlo (SMC) is a new branch of Monte Carlo simulation, not necessarily a Markov chain, which uses a population of samples to carry out on-line approximations of a target distribution. SMC has sampling procedures which are similar to proportional selection in EAs. Bienvenue et al. introduce niching in some SMC algorithms to maintain diversity in the population [23]. Del Moral [24] study the convergence of GAs with SMC. Higuchi [25] and Tito, Vellasco and Pacheco [26] proposes GA filter and Genetic Particle Filter, respectively. They integrate recombination into SMCs [25] to speed up convergence.

5 Elitist Coupled Acceptance Rule and Fitness Ordered Tempering

In the previous section we have surveyed a number of EMCMC algorithms, and discussed some techniques that showed how multiple MCMC chains could exchange information in order to speed up the convergence to some target distribution. In the following we introduce the elitist coupled acceptance rule (ECA) and fitness ordered tempering (FOT), whose main purpose is to converge efficiently to a target distribution that is biased towards the most likely states (or good solutions).

Fitness ordered tempering (FOT). FOT maintains a population of N chains each having its own temperature $Temp[i]$. As in parallel tempering, FOT has a - typically geometrical - increasing series of temperatures $Temp[1] = 1 < \ldots < Temp[N] = Temp^{max}$. The population of N solutions (or states) is sorted according to their fitness (or probability), and the solution at rank i gets the temperature $Temp[i]$, where the most fit solution gets $Temp[1] = 1$, and the worst solution $Temp[N] = Temp^{max}$. Therefore a solution has lower temperature than any solution worse than itself, unless they are copies of each other. In case

there are multiple copies of the same solution ties within the sorted population are broken randomly, so each copy gets an adjacent temperature. Copies receive a different temperature to avoid that multiple copies of good solutions will remain in the population almost indefinitely.

In case there are multiple solutions with the same fitness but who are not copies of each other, the temperature ladder has to be recomputed so that each unique solution with the same fitness gets the same temperature. The number of different temperature values $Temp[i]$ at each generation is therefore equal to the number of solutions with different fitness value, unless they are copies, in which case they also get a different temperature value. This scheme is necessary to avoid that some solution might get another temperature in an identically composed population, and ensures that FOT has a homogeneous Markov chain transition matrix at the population level. Contrary to parallel tempering, here the temperature assignment depends on the fitness of the solutions relative to the fitness of the other solutions in the current population. Since we want to remain in the vicinity of good solutions, we prefer to assign the lower temperatures to the better solutions.

Elitist coupled acceptance rule (ECA). The ECA algorithm applies a coupled Metropolis Hastings acceptance rule to two solutions and their two children. ECA accepts the best two solutions from the family of four if at least one of them is a child. However, when both children have a lower fitness than both their parents, the children can still replace the parents with a probability determined by the coupled acceptance rule. This increases the explorative character of the algorithm. Call $X'_t = (\ldots, X_t[i], \ldots, X_t[j], \ldots)$, $X'_{NEW} = (\ldots, X_{NEW}[i], \ldots, X_{NEW}[j], \ldots)$, $P'_{X_t}(X[\cdot]) = \exp\left(\frac{f(X_t[\cdot])}{Temp[\cdot]}\right)$, and $\max_2$ the function returning the two most fit solutions. The ECA acceptance rule is now given as:

$\texttt{ECA}(((X_{NEW}[i], X_{NEW}[j]) \mid (X_t[i], X_t[j])))$
1 **if** $\{X_t[i], X_t[j]\} = \{X_{NEW}[i], X_{NEW}[j]\}$
2 **then return** 1
3 $(X_{MAX}[i], X_{MAX}[j]) \leftarrow \max_2 (X_{NEW}[i], X_{NEW}[j], X_t[i], X_t[j])$
4 **if** $X_{MAX}[i] \in \{X_{NEW}[i], X_{NEW}[j]\} \vee X_{MAX}[j] \in \{X_{NEW}[i], X_{NEW}[j]\}$
5 **then return** 1
6 **else return** $\frac{P'_{X_t}(X_{NEW}[i]) \cdot P'_{X_t}(X_{NEW}[j])}{P'_{X_t}(X_t[i]) \cdot P'_{X_t}(X_t[j])} \cdot \frac{S(X'_t|X'_{NEW})}{S(X'_{NEW}|X'_t)}$

Note that the temperatures of the current and accepted solutions remain the same. ECA can be viewed as a combination between the family-competitive, elitist replacement rule from regular GAs [8] and the coupled acceptance rule A_C. To see the difference between ECA and A_C let us consider 2 parent states and their offspring such that $P'_{X_t}(X_{NEW}[i]) > P'_{X_t}(X_t[j]) > P'_{X_t}(X_t[i]) > P'_{X_t}(X_{NEW}[j])$. Call $F = \frac{P'_{X_t}(X_{NEW}[i])}{P'_{X_t}(X_t[i])} \cdot \frac{P'_{X_t}(X_{NEW}[j])}{P'_{X_t}(X_t[j])}$. If $F < 1$, A_C may loose $X_{NEW}[i]$ which is the best solution of this family, while if $F > 1$, A_C accepts $X_{NEW}[j]$ which is the worst solution.

Combining the FOT temperature assignment mechanism and the ECA acceptance rule leads to the ECA/FOT MCMC algorithm: each generation the temperatures of each individual in the population are recomputed with FOT, new individuals are proposed by mutation and recombination, and are accepted - or rejected - with the ECA acceptance rule. It is important to note that at the individual level the transition matrix of ECA/FOT is not stationary, since, in time, the same fitness values may be associated with different temperatures. At the population level however, the transition matrix is stationary because for the same population of individuals, the temperature is always assigned in the same way. The ECA/FOT MCMC is aperiodic, since, for each state, there is a non-zero probability to remain in the current state, and irreducible, because it has a non-zero probability to arrive from each state to each other state. If the state space is finite, ECA/FOT converges to a distribution $R(\cdot) = \prod_{N}^{i=1} R'(\cdot)$, where $R'(\cdot)$ is the unnormalized distribution for a single chain which depends on $Temp_{MAX}$ and $Temp_{MIN}$, the temperature assignment (e.g. geometrically) and the size of the population.

6 Experimental Results

To illustrate the use of the ECA/FOT algorithm we have applied 6 (E)MCMC algorithms to the well known deceptive trap function [27], and counted the number of different solutions visited as a function of their fitness. The 6 algorithms are multiple independent MCMC chains (MIC), population MCMC (popMCMC), parallel tempering (PT), parallel tempering with recombination, parallel recombinative simulated annealing (PRSA), and elitist coupled acceptance with fitness ordered tempering (ECA/FOT). The first three algorithms do not apply recombination, while the other three do. For comparison purposes, we sample for each algorithm from the Boltzmann distribution $\exp\left(\frac{f(\cdot)}{Temp[\cdot]}\right)$. In the MIC algorithm, each chain has been given a constant temperature calculated with the PT ladder. The PRSA algorithm uses the MH acceptance rule A (instead of the logistic acceptance rule), and two single acceptance/rejection competition between a parent and one of its children.

The trap function has 10 building blocks of length $k = 3$ and a linearly scaled fitness function: $f = \sum_{i=1}^{10} i \cdot f_i$. f_i is the fitness value of the i-th trap function and depends only on the number of one bits at each building block: $f_i(3) = 5, f_i(2) = 0, f_i(1) = 3, f_i(0) = 4$. The algorithmic parameters are chosen as follows. We set $Temp_{MAX} = 160$ for the MH acceptance rule and $Temp_{MAX} = 320$ for the ECA acceptance rule, resulting in an acceptance probability of 0.73 for two individuals with fitness difference equal to 50. $Temp_{MIN}$ is equal to 1. We choose a geometrically increasing temperature ladder that increases the temperature each step with $\beta = \exp\left(\frac{1}{s} \cdot \log \frac{Temp_{MIN}}{Temp_{MAX}}\right)$, where s is the total number of steps for the scheduler. At step j, $Temp_j = Temp_{MAX} \cdot \beta^j$. We choose the population size N equal with the number of generations to obtain the same β for all algorithms. If $\beta = 0.97$, we obtain $N = 250$. The mutation rate is $\frac{3}{l}$, which is the optimal rate for a deceptive function with building block length

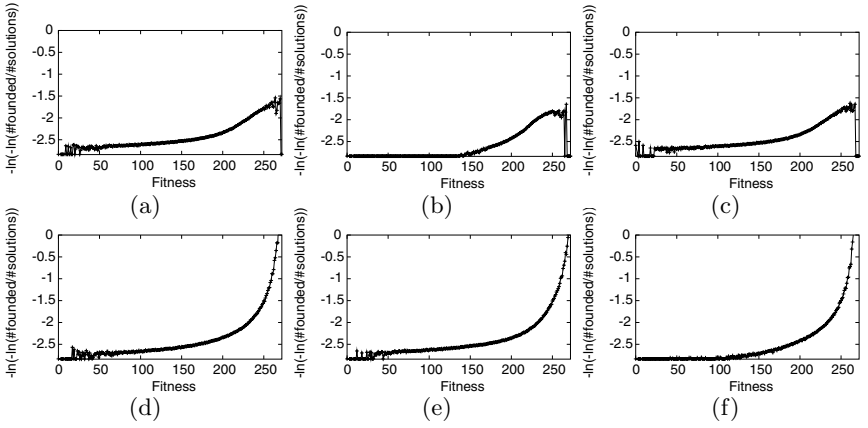


Fig. 1. The number of different solutions found over the total number of possible solutions for a fitness value on a logarithmic scale for (a) MIC, (b) popMCMC, (c) PT, (d) PT with recombination, (e) PRSA and (f) ECA/FOT

$k = 3$. The recombination operator is two point crossover. For each algorithm, we have made 10 independent runs and sampled the whole population every 10 generations, resulting in 25 samples and 6250 solutions in total.

Figure 1 shows – as expected – the advantage of using recombination as proposal operator for functions (or distributions) whose structure can be exploited by the crossover operator. One can also notice that ECA/FOT is more biased towards highly likely individuals than PT or PRSA.

7 Conclusion

Evolutionary Markov Chain Monte Carlo combine techniques form Evolutionary Computation and parallel Markov Chain Monte Carlo algorithms to design new algorithms for sampling or optimising complex distributions resp. functions. EMCMC algorithms offer new proposal operators and new acceptance rules. Individual states in the population can be single MCMC chains that interact with each other, though it is not necessary that they are indeed single MCMC chains. At the population level however - this is, states are entire populations of given size - EMCMC algorithms need to be MCMC chains.

We have surveyed a number of existing EMCMC algorithms in the literature, and categorised them in two classes: *family-competitive EMCMC* and *population-driven EMCMC* algorithms. We have also introduced an EMCMC algorithm, ECA/FOT, which applies a Fitness Ordered Temperature assignment mechanism and an Elitist Coupled Acceptance rule. EAC/FOT is, at population level, an MCMC which converges to a stationary distribution biased towards the most likely states.

Clearly, the merger of the EC and MCMC paradigms represents a rich source for future development of powerful sampling and optimisation algorithms.

References

1. Andrieu, C., de Freitas, N., Doucet, A., Jordan, M.: An introduction to MCMC for Machine Learning. Machine Learning (2003) 5–43.
2. Hastings, W.: Monte Carlo sampling methods using Markov chains and their applications. Biometrika **57** (1970) 97–109.
3. Metropolis, N., Rosenbluth, A., Rosenbluth, M., Teller, A., Teller, E.: Equation of state calculations by fast computing machines. Jornal of Chemical Phisics **21** (1953) 1087–1092.
4. Gilks, W., Richardson, S., Spiegelhalter, D., eds.: Markov Chain Monte Carlo in practice. Chapman & Hall (1996).
5. Geyer, C.J.: Markov Chain Monte Carlo maximum likelihood. In: Computing Sc. and Stat.: Proceedings of the 23rd Symposium on the Interface. (1991) 156–163.
6. Holloman, C.H., Lee, H.K.H., Higdon, D.M.: Multi-resolution Genetic Algorithms and Markov Chain Monte Carlo. Technical report, Duke University (2002).
7. Liu, J., Sabatti, C.: Simulated Sintering: Markov Chain Monte Carlo with spaces of varying dimensions. In: Bayesian Statistics 6. Oxford Univ. Press (1999) 389–413.
8. Thierens, D., Goldberg, D. E.: Elitist recombination: an integrated selection-recombination GA. In: Proceedings of the First IEEE World Congress on Computational Intelligence. (1994) 508 – 512.
9. 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.
10. Culberson, J.: Genetic invariance: A new paradigm for genetic algorithm design. Technical Report 92-02, Edmonton, Canada (1992)
11. Pelikan, M., Goldberg, D.E., Lobo, F.: A survey of optimization by building and using probabilistic models. Comp. Optimi. and Appl. **21** (2002) 5–20.
12. Liang, F., Wong, W.H.: Evolutionary Monte Carlo: Applications to C_p model sampling and change point problem. In: Statistica Sinica. (2000) 317–342.
13. Liang, F., Wong, W.: Evolutionary Monte Carlo for protein folding simulations. In: Journal of Chemical Physics. Number 115 (2001) 3374–3380.
14. Laskey, K.B., Myers, J.W.: Population Markov Chain Monte Carlo. Machine Learning (2003) 175–196.
15. Zhang, B.T., Cho, D.Y.: System identification using evolutionary Markov Chain Monte Carlo. System Architecture (2001) 287–599.
16. Goldberg, D.E.: A note on Boltzmann tournament selection for Genetic Algorithms and Population-oriented Simulated Annealing. Complex Systems **4** (1990) 445–460.
17. Mahfoud, S.W., Goldberg, D.E.: Parallel Recombinative Simulated Annealing: a Genetic Algorithm. Parallel Computing (1995) 1–28.
18. Rudolph, G.: Massively Parallel Simulated Annealing and its Relation to Evolutionary Algorithms. Evolutionary Computation (1994) 361–382.
19. Cercueil, A., François, O.: Monte Carlo simulation and population-based optimization. In: Congress on Evolutionary Computation 2001. (2001) 191–198.
20. Cerf, R.: A new genetic algorithm. Ann. of Appl. Probab. **6** (1996) 778–817.
21. Lozano, J.A., Larrañaga, P., Graña, M., Albizuri, F.X.: Genetic algorithms: bridging the convergence gap. Theoretical Computer Science **229** (1999) 11–22.
22. Strens, M.J.A.: Evolutionary MCMC sampling and optimization in discrete spaces. In: Proceedings of the Twentieth International Conference on Machine Learning ICML-2003 (2003).

23. Bienvenüe, A., Joannides, M., Bérard, J., Fontenas, E., François, O.: Niching in Monte Carlo Filtering Algorithms. In P. Collet, C. Fonlupt, J.-K. Hao, E. Lutton, M. Schoenauer, ed.: Artificial Evolution, 5th International Conference, EA 2001, Selected Papers, LNCS 2310, Springer (2002) 19–30.
24. Moral, P., Kallel, L., Rowe, J.: Modelling Genetic algorithms with Interacting Particle systems. In: Theoretical Aspects of Evolutionary Computing. L. Kallel, B. Naudts, A. Rogers (2001) 10–67.
25. Higuchi, T. In: Self-Organizing Time Series Model. Volume Sequential Monte Carlo Methods in Practice. Springer (2001) 429 – 444.
26. Tito, E., Vellasco, M., Pacheco, M.: Genetic Particle Filter: An evolutionary perspective of SMC methods. Technical report, Catolic Univ. of Rio de Janeiro (2002).
27. Goldberg, D.E., Deb, K., Horn, J.: Massive multimodality, deception, and Genetic Algorithms. In: Proceedings of Parallel Problem Solving from Nature (1992) 37–46.

A Hybrid Evolutionary Algorithm for CSP

Vincent Barichard, Hervé Deleau, Jin-Kao Hao, and Frédéric Saubion

LERIA, Université d'Angers
2 Bd Lavoisier, F-49045 Angers Cedex 01

Abstract. In this paper, we present a new framework for combining complete and incomplete methods in order to solve constraint satisfaction problems. This algorithm scheme uses constraint propagation techniques and local search heuristics embedded in an evolutionary computation context. The uniformity of the involved structures provides an harmonious interaction between the different implemented methods, and also allows to take advantage of the respective methods. Furthermore, the great flexibility of this model allows us to foresee various extensions. We emphasize the interest of our approach on some examples which are solved by means of an implementation.

1 Introduction

Constraint Satisfaction Problems (CSP) [14] provide a general framework for the modeling of many practical applications (planning, scheduling, time tabling...). CSP's are defined by a set of variables associated to domains of possible values and by a set of constraints. Constraints can be understood as relations over the variables and therefore, solving a CSP consists in finding an assignment of values to the variables that satisfies these constraints. Many resolution algorithms have been proposed to achieve this purpose and we may distinguish at least two classes of methods:

- *Complete methods* are able to explore the whole search space in order to find all the solutions or to detect that the CSP is not consistent. This approach requires an important computation effort and therefore encounters some difficulties with large scale problems. Complete methods are mainly based on local consistency mechanisms [9,11] which allow the algorithms to prune the search space.
- *Incomplete methods* mainly rely on the use of heuristics providing a more efficient exploration of interesting areas of the search space in order to find some solutions. But, these approaches do not ensure to collect all the solutions nor to detect inconsistency.

A common idea to get more efficient and robust algorithms consists in combining the previous approaches in order to take advantage of their respective advantages. The hybridization of the previous techniques has already been studied to solve CSP's [7,12]. But, most of the time, the combination is quite heterogeneous, due to the different structures and exploitation of the search space used

P. Liardet et al. (Eds.): EA 2003, LNCS 2936, pp. 79–90, 2004.

by each paradigm. The purpose of this paper is to propose an original uniform model in order to integrate these various methods.

Our model is based on the management of populations which have to adapt to two main criteria: local consistency which allows us to reduce the search space and global consistency which consists in reaching a solution. Therefore, individuals of these populations represent parts of the search space (this idea has been investigated in [2]). They cover the whole set of possible solutions at any time and are also used to intensify the search on particular areas.

2 An Evolutionary Algorithm for CSP

We present here a general formulation of Constraint Satisfaction Problems (CSP) [14]. A CSP is a tuple (X, D, C) where $X = \{x_1, \cdots, x_n\}$ is a set of variables taking their values in domains $D = \{D_1, \cdots, D_n\}$. Note that we consider in this paper only finite domains. A constraint is a relation $c \subseteq D_1 \times \cdots \times D_n$. An assignment is a function $v: X \rightarrow \bigcup_{1 \leq i \leq n} D_i$ such that $\forall\ 1 \leq i \leq n, v(x_i) \in D_i$. An assignment v is a solution of a CSP (X, D, C) if and only if $\forall c \in C, (v(x_1), \cdots, v(x_n)) \in c$. We denote Sol the set of all the solutions of the CSP.

We want to use an evolutionary framework allowing us to combine constraint propagation techniques and heuristics methods. Evolutionary algorithms are mainly based on the notion of adaptation of a population of individuals to a criterion thanks to evolution operators like crossover [6]. Our method acts on populations of individuals which represent parts of the search space of the initial CSP. These parts can be reduced to single points. We propose then two successive stages of adaptation. A first stage of local adaptation, which corresponds to a criterion of local consistency, in order to reduce the search space. The second stage corresponds to the global consistency criterion in order to extract solutions. Therefore two populations are introduced to achieve this purpose: a population to cover the search space (called the covering population) and a population to intensify the search (called the intensification population). Selection and evolution operators are introduced in order to control the various possible search strategies. The general principle of the approach is shown on Fig. 1.

2.1 Search Space Representation

A classical approach for representing the search space of a CSP (X, D, C) consists in considering the set of all possible assignments which corresponds to $S = D_1 \times \cdots \times D_n$. But, here, we use individuals which represent parts of this search space and which associate to each variable a subset of its domain.

Definition 1 (Individual).
Given a CSP (X, D, C), the set of individuals is defined as:

$$I = \{(d_1, \cdots, d_n) \mid \forall 1 \leq i \leq n, d_i \subseteq D_i\}\,.$$

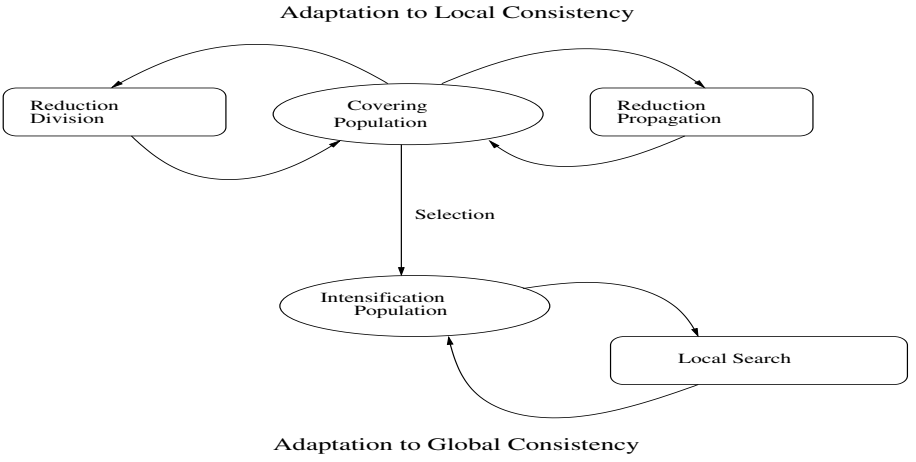


Fig. 1. General Principle

For any $\xi \in I$, ξ_i is the i^{th} component of ξ. We deduce that $\xi \in I$ is a solution iff $\forall 1 \leq i \leq n, |\xi_i| = 1$ and $\forall c \in C, \xi \in c$. ξ is inconsistent iff $\exists 1 \leq i \leq n, \xi_i = \emptyset$. We define an order $\sqsubseteq$ on individuals:

Definition 2 (Order on Individuals).

$$\forall \xi, \xi' \in I, \xi \sqsubseteq \xi' \text{ iff } \forall i, \xi_i \subseteq \xi'_i \,.$$

Remark that, according to this order, the smaller is an individual, the smaller is its corresponding part of the search space. We get a lattice $(I, \sqsubseteq)$ with a smallest element$(\emptyset, \cdots, \emptyset)$ and a biggest one $(D_1, \cdots, D_n)$. The notion of solution concerns only points. Let $point(\xi) = \{\xi' \in I \mid \forall 1 \leq i \leq n, |\xi'_i| = 1 \wedge \xi' \sqsubseteq \xi\}$ be the set of points contained in ξ. Then, $Sol(\xi) = point(\xi) \cap Sol$ is the set of solutions included in ξ. A population is a set of individuals and therefore, $P = 2^I$ is the set of all possible populations. The set of solutions included in a population $p \in P$ is $Sol(p) = \bigcup_{\xi \in p} Sol(\xi)$. We now extend the previous order to P.

Definition 3 (Order on Populations).

$$\forall\, p, p' \in P, p \sqsubseteq p' \text{ iff } \forall\, \xi \in p, \exists\, \xi' \in p' \xi \sqsubseteq \xi' \,.$$

We consider then a lattice of populations $(P, \sqsubseteq)$ with a smallest element $\emptyset$ and a biggest one $\top$. This lattice is explored according to two main principles. The first principle is the adaptation to the local consistency criterion while the second will be the adaptation to global consistency.

2.2 Adaptation to the Notion of Local Consistency

Local consistency methods [9] aim at eliminating points which are not solutions by a local examination of the constraints. This notion does not imply global

consistency but the reverse is true. The adaptation of the covering population to local consistency is performed in order to get a smaller population preserving the set of solutions. In order to reduce the individuals of the population, we use two basic operators: domain reduction and domain splitting.

Reduction by Propagation. Inspired by the work of [1,3], we present constraint propagation as a set of reduction operators allowing us to compute a fix point. We first define this notion on individuals.

Definition 4. *Given a CSP* (X, D, C)*, a constraint propagation operator for a constraint* $c \in C$ *is a function* $\pi : I \to I$ *which satisfies:*

- $\pi(\xi) \sqsubseteq \xi$ *(contraction)*
- $\xi \cap c \sqsubseteq \pi(\xi)$ *(correction)*
- $\xi \sqsubseteq \xi' \Rightarrow \pi(\xi) \sqsubseteq \pi(\xi')$ *(monotonicity)*

We use *red* to denote the set of reduction operators associated to the CSP and satisfying the previous definition. According to [3], we know that the successive application of operators from *red* allows us to reach a fixpoint which corresponds to the smallest individual (w.r.t.$\sqsubseteq$) which contains the same solutions.

In our context, constraint propagation acts on a population of individuals. In order to keep generality, we extend the previous definition as follows:

$$\begin{aligned} &\Pi : P \times red \to P \\ &\quad (p, \pi) \mapsto p_1 \cup \{\pi(\xi) \mid \xi \in p_2\} \text{ with } (p = p_1 \cup p_2) \wedge (p_1 \cap p_2 = \emptyset) \end{aligned}$$

This generalization allows us to apply simultaneously reductions on several individuals by giving a partition p_1, p_2 of p. We may, for instance, apply the same operator to all the individuals or to a particular individual. We use RED to denote the set of all previously defined generalized operators.

Let p_{RED} be the fixpoint obtained by the iterative application of reductions on the individuals of a population p. We delete from p_{RED} all the individuals which are not consistent (locally and therefore globally). These individuals $\xi \in p_{RED}$ are such that $\exists 1 \leq i \leq n, \xi_i = \emptyset$. We define then a general function:

Definition 5. Local Consistency

$$\begin{aligned} &CL : P \to P \\ &\quad p \mapsto \{\xi \in p_{RED} \mid \forall 1 \leq i \leq n, \xi_i \neq \emptyset\}. \end{aligned}$$

Remark that $CL(p) \sqsubseteq p$ and that solutions are preserved ($Sol(p) \subseteq Sol(CL(p))$ according to definition 4). This function allows us to move on the lattice of population from a population to a smaller one having the same set of solutions. We define now another operator to reduce again the search space.

Reduction by Splitting. We introduce here a particular operator which differs from usual crossover operators [6]. Actually, we use a cellular division in which an individual generates several new individuals. This division is controlled by a variable whose domain will be split. We define then a set of operators for the variables $x \in X$. For an individual $\xi \in I$ and a variable $x \in X$, we denote ξ_x the domain of x in ξ. Our operators are then indexed by a variable:

$$\begin{array}{l}\chi_x \colon I \to P \\ \quad \xi \mapsto \{\xi^1, \cdots, \xi^n\} \\ \quad \text{with } \bigcap_{i=1}^n \xi_x^i = \emptyset \wedge \bigcup_{i=1}^n \xi_x^i = \xi_x \\ \quad \forall\, y \neq x,\ \xi_y^1 = \cdots = \xi_y^n = \xi_y\,;\end{array}$$

χ_X denotes the set of these operators. Remark that $\forall\ 1 \leq i \leq n,\ \xi^i \sqsubseteq \xi$ and χ_x preserves solutions: $Sol(\xi) = \bigcup_{i=1}^n Sol(\xi^i)$. We generalize this definition in order to allow simultaneous divisions. A division strategy simply consists in choosing p_1 and p_2 and the variable x providing the operator χ_x. For this purpose, we introduce now two examples of selection operators which correspond to such strategies choices:

$$\begin{array}{l}\sigma_1 \colon P \to P \\ \quad p \mapsto \{min(p)\}\end{array}$$

where $min(p)$ provides as output an individual ξ whose domains are minimal (i.e., satisfying $\forall\, \xi' \in p, \xi' \neq \xi \forall\ 1 \leq i \leq n, |\xi_i| \leq |\xi'_i|$). This function σ_1 allows us to focus on the search on individuals whose domains will have decreasing sizes. We may also simulate another kind of exploration.

$$\begin{array}{l}\sigma_2 \colon P \to P \\ \quad p \mapsto \{\xi \in p \mid \nexists\, \xi' \in p, \xi' \sqsubseteq \xi\}\,.\end{array}$$

This operator correspond to an exploration in width of the search space. Let Σ be the set of selection operators. According to our general principle, this selection should be used to select the candidates for a division but also to select the individuals on which the search will be intensified in the second stage:

$$\begin{array}{l}\chi_x \colon P \times \Sigma \to P \\ \quad (p, \sigma) \mapsto \bigcup_{\xi \in \sigma(p)} \chi_x(\xi)\,.\end{array}$$

We define now our adaptation-evolution stage for the local consistency as a sequence of pairs $(\chi^i, \sigma^i)_{1 \leq i \leq k}$ with $\chi^i \in \chi_X$ and $\sigma^i \in \Sigma$. Given a population $p \in P$, we get its adaptation as the sequence:

$$\begin{array}{l}p_0 = CL(p) \\ p_{i+1} = CL((p_i \setminus \sigma^i(p_i)) \cup \chi^i(p_i, \sigma^i))).\end{array}$$

$LCA(p, (\chi_i, \sigma_i)_{1 \leq i \leq k}) = p_k$ is the population obtained after k iterations of the process *consistency-selection-division*. We have now to define the way we use the covering population to generate the intensification population.

2.3 Selection of the Sub-population for the Intensification Search

We are interested here in building a population whose purpose is to adapt itself to a global consistency criterion. This corresponds to generate individuals which are solutions or, at least, close to a solution. Our framework is general enough to apply methods working on any kind of individuals. But, in the rest of this paper, we will focus on individuals which are points issued from areas represented by individuals of the covering population. We define then an operator to generate points from the covering population. In order to guide the local search w.r.t. the area to explore (and therefore w.r.t. the individual of the covering population which can be considered as the father of this point), we index individuals of the intensification population (points) by individuals from the covering population (their fathers):

$$\begin{aligned}\gamma\colon & P \times \Sigma \to P \\ & (p,\sigma) \mapsto \gamma(p,\sigma) \text{ with } \gamma(p,\sigma) \subseteq \{\xi'_\xi \in I \mid \xi \in \sigma(p), \xi'_\xi \in point(\xi)\}\,.\end{aligned}$$

Individuals from the covering population restrict and focus the search on points of the intensification population $\gamma(p,\sigma)$ issued from the area they represent.

2.4 Global Consistency Adaptation

The purpose of this second stage is to adapt the intensification population to a global consistency criterion related to the notion of solution. This criterion can usually be stated as the number of constraints which are not satisfied by an individual. Therefore, an individual is better if it satisfies the maximum number of constraints. We use local search to achieve this adaptation. Our purpose is now to present the main components needed in a local search algorithm. We first define the evaluation function.

Definition 6 (Evaluation).

$$\begin{aligned}\psi\colon & I \to \mathbb{N} \\ & \xi \mapsto |\{c \in C \mid \xi \notin c\}|\,,\end{aligned}$$

where $|A|$ is the cardinality of a set A. We have obviously: $\forall\, \xi \in I, \psi(\xi) = \emptyset \Rightarrow \xi \in Sol$. According to the basic principle of local search algorithms, we need a way to move from an individual to another in order to find more and more suitable individuals w.r.t. the evaluation function. As mentioned before, an individual from the covering population is associated to each point of the intensification population. Therefore, our local search operators ω_ξ act on an individual ξ'_ξ and are restricted by ξ. Such an operator has the following template:

$$\begin{aligned}\omega_\xi\colon & I \to I \\ & \xi'_\xi \mapsto \omega_\xi(\xi'_\xi)\end{aligned}$$

with $\omega_\xi(\xi'_\xi) \subseteq point(\xi)$. The role of ξ appears now clearly as it restricts local search to the search area it represents. We first define a neighborhood function:

$$\begin{aligned}\mu_\xi\colon & I \to P \\ & \xi'_\xi \mapsto \mu(\xi'_\xi) \text{ with } \forall \xi''_\xi \in \mu(\xi'_\xi), \xi''_\xi \in point(\xi)\,.\end{aligned}$$

We have now to describe search strategies over these neighborhoods. We define two basic operators on individuals in order to model various local search heuristics. The first basic heuristics (usually called descent) consists in always moving to a better neighbor and stops if no such neighbor exists. Let

$$D_\xi \colon I \to P$$
$$\xi'_\xi \mapsto \begin{cases} \xi'_\xi \text{ if } \xi'_\xi \in Sol \\ D_\xi(\xi'_\xi) \subseteq \{\xi''_\xi \in I \mid \xi''_\xi \in \mu_\xi(\xi'_\xi), \psi(\xi''_\xi) < \psi(\xi'_\xi)\} \text{ otherwise.} \end{cases}$$

This operator is naturally extended to populations:

$$D_\xi(p) = \bigcup_{\xi'_\xi \in p} D_\xi(\xi'_\xi).$$

From an individual ξ'_ξ, such an operator always reaches a fixpoint which is a set of local optima. We have then: $\exists k \in \mathbb{N}, D_\xi(D^k_\xi(\xi'_\xi)) = D^k_\xi(\xi'_\xi)$. This property comes from the order induced on I by the function ψ. In order to diversify the search and to escape from this set if it does not contain any solution, we introduce another operator allowing us to choose alternative paths in the neighborhood.

$$CA_\xi \colon I \to P \times I$$
$$\xi'_\xi \mapsto \begin{cases} \xi'_\xi \text{ if } \xi'_\xi \in Sol \\ CA_\xi(\xi'_\xi) \subseteq \{\xi''_\xi \in I \mid \xi''_\xi \in \mu_\xi(\xi'_\xi)\} \text{ otherwise.} \end{cases}$$

A local search rl_ξ is then a finite sequence of symbols in $\{D_\xi, CA_\xi\}$. Given a local search rl_ξ and an individual ξ'_ξ, $rl_\xi(\xi'_\xi)$ is the population obtained by successive applications of operators in rl_ξ. Let RL be the set of all local searches.

From a practical point of view, implementation of such operators consists in choosing a single point for the subset generated at each iteration (the first point of the set, a random point from this set ...). Moreover, we may remark that this general definition corresponds to different classical local search algorithms and that a strategy is characterized by the way of alternating D_ξ and CA_ξ. This can be achieved by a temperature parameter as in simulated annealing [8], by the memorization of previous visited points as in tabu search [5] or by any other heuristics.

The global consistency adaptation function is defined as:

Definition 7.

$$GCA \colon P \times 2^{RL} \to P$$
$$(p, \rho) \mapsto \textstyle\bigcup_{\xi'_\xi \in p} rl_\xi(\xi') \ \textit{with}\ rl_\xi \in \rho\,.$$

2.5 General Algorithm

We present here the general algorithm in a simplified version where we apply successively a local and a global adaptation. We introduce a set allowing us to collect all the solutions found during the whole process. Remark that a solution may appear during the local consistency adaptation after a division whose result is a single solution point, or during the global consistency adaptation thanks to the local search.

$S = \emptyset$
$p \leftarrow \{(D_1, \cdots, D_n)\}$
Repeat
    `Choose` $\sigma \in \Sigma, x \in X$
    $p = LCA(p, (\chi_x, \sigma))$`Local consistency application`
    $S = S \cup (p \cap Sol)$ `Extraction of possibly obtained solutions`
    $p = p \setminus (p \cap Sol)$
    `Choose` $\sigma \in \Sigma, \rho \in 2^{RL}$ `Selection function for intensification` population
    $p' = GCA(\gamma(p, \sigma), \rho)$ `Local search application`
    $S = S \cup (p' \cap Sol)$ `Extraction of possibly obtained solutions`
`Until` *stop*.

The stop criterion *stop* mentioned here can be considered from different points of view. One may choose to stop either because the covering population p is empty (in this case, the entire search space has been explored and all the solutions have been found) or because a first solution (or a certain number of solutions) has been obtained. But, one may also restrict the number of allowed iterations. These different possibilities allow us to choose between complete and incomplete search processes. The output of the algorithm is double since we have the set of found solutions and the current cover of the set of all solutions.

2.6 Extensions and Specificities

We discuss here the possible extensions and specificities of our model.
Covering population size: The maximal size of the population allows us to control and to limit the development of the search areas. Selection operators allow then to control reductions by splitting. If a maximal size is fixed, splitting will stop in favour of local search. We remove individuals from the covering population in two cases: when they correspond to solution points or when the individuals are not locally consistent. We may also loose completeness and remove individuals when a solution has been found in the corresponding area for instance (in order to find several diversified solutions). It shows that the covering population allows us to guide the local search during the intensification stage.
Max-CSP problems: Our framework is also suitable for the Max-CSP problem which consists in satisfying the maximum number of constraints in a CSP. We only have to keep the best solution found during the global consistency adaptation w.r.t. the evaluation function.
Constraint Optimization Problems: This general framework can be extended to constraint optimization problems where an objective function has to be optimized under a set of constraints. In that case, we may use the objective function to estimate the value of an area and then remove areas which are not promising enough. This corresponds to an integration of so-called *branch and bound* techniques.
Introduction of recombination operators and extension of local search: Taking benefits from our intensification population, we could extend local search

mechanisms by adding recombination operators. We have to check then that such recombinations are not escaping from the search area or control such an escape. It is also possible to imagine a local search which escapes from the restricted area. In this case we could introduce a notion of distance to control this evolution.
Distributed algorithms: The computation model proposed here can be adapted to distributed computation. Actually, populations and individuals can be distributed on different processes and treated separately.

2.7 Related Works

Many existing works propose hybridization between local search based methods and constraint propagation techniques but they often deal with a specific algorithm dedicated to a particular problem.

We should mention [4] which presents a general overview of the use of local search in constraint programming. [7] aims at homogenizing the combination of these methods and their main aspects. We also mention [12] as an application of such hybridization for the vehicle routing problems. [13] proposes a combination technique which improves the efficiency of the evolutionary methods for CSP.

3 Application

In this section, we present an implementation of the concepts described above. This prototype allows us to validate our approach on different CSP examples. The local consistency stage is implemented by constraint propagation operators which are based on bound consistency (see [10]). The split function consists in cutting in two random parts the smallest domain of the minimal individual.

The purpose of this section is not to test a high performance algorithm on large scale benchmarks but to compare the benefit of the combination of methods w.r.t. a single use of them.

For the global consistency stage, we choose a simple hill climbing method with a random walk. For each individual of the covering population, a single point is inserted in the intensification population. The covering and intensification stages are first used in an independent manner. Then, their results are compared to those obtained by the combination of these two stages. The covering stage can be viewed as a complete resolution method with backtracking and local consistency at each node of the tree. The intensification stage simulates a local search method.

3.1 Selected Problems

We propose here a sample of various families of problems [10] (w.r.t. the number of variables, constraints and domains).
S+M=M: This is a well-known crypto-arithmetic problem. It consists in solving the equation $SEND+MORE = MONEY$ by assigning a different digit to each

letter. It contains 8 variables, one equality constraint and many disequations which ensure that all variables are all different.
Race Puzzle: This problem consists in finding the order of arrival in a race knowing particular information about the competitors. It has 6 variables and 29 constraints involving disequations.
Knapsack: This is a knapsack problem with 30 variables whose domain size is 20 and 3 knapsack constraints. There is no objective function to optimize, but a minimal cost to satisfy.
Affectation : This problem consists in affecting tasks to workforces in a bijective manner according to a particular cost. The instance 1 comprises 16 variables that belong to $\{0, 1\}$ and 9 constraints. The instance 2 comprises 25 variables and 11 constraints.
Scheduling: This is a linear scheduling problem. Instance 1 has 6 variables and 29 inequality constraints and instance 2 has 15 variables and 29 constraints. Each variable has a domain size of 1000.
Magic Square: This is the classical magic square benchmark which consists in positioning integer numbers from 1 to n^2 in a square board of size n such that the sums of all columns, lines and diagonals are equal.

3.2 Experimental Results

The values given in the following arrays correspond to the truncated average of 50 independent runs. We count the total number of generated individuals (ngi) in the covering population. We also count the total number of moves (nm) performed by the local search on the intensification population. Concerning local search alone, we also mention the success rate (sr). The standard deviation for the number of moves (stdnm) is mentioned only when results are available for GCA to compare it with LCA+GCA. The mark "-" indicates that we did not get any answer (memory > 4GB or computation time > 1h). For LCA+GCA, we limit the number of moves for each local search use to 50. GCA alone is used with a maximum of $500,000$ moves.

We compare first the number of individuals generated in the covering population with LCA and LCA+GCA to get the first solution. For LCA alone, it corresponds to the number of nodes of a classical backtracking tree. Table 1 shows that LCA+GCA finds a solution with a smaller number of individuals compared to LCA alone. In the combination, the relative efficiency of the GCA part depends on the problem.

For problems with one solution such as *S+M=M* or *Race Puzzle*, the benefit is less significant than for the *Scheduling* or *Affectation* problems.

Table 1 also shows the efficiency of the local consistency which guides the local search (see comparisons LCA+GCA vs. GCA), in particular on the *S+M=M*, *Race Puzzle* or *Magic-Square* problems, where GCA alone fails to find a solution. Actually, GCA has good results when problems have many solutions with a good spread (especially on the *Knapsack* problem which is indeed an easy satisfaction problem but with large domains for our LCA). We highlight that for the *Affectation* problems, the robustness of GCA alone is weak. Indeed, the standard

Table 1. First solution (LCA vs. LCA+GCA vs. GCA)

Problem	LCA ngi	LCA + GCA (ngi,nm) [stdnm]	GCA (sr,nm) [stdnm]
S+M=M	3111	(2532,1191)	(0%,-)
Race Puzzle	129	(96,44)	(0%,-)
Knapsack	-	(8,202) [19]	(100%,36) [17]
Affectation-1	14	(4,11) [2]	(100%,214) [165]
Affectation-2	34	(8,44) [3]	(100%,1480) [556]
Scheduling-1	109	(1,1) [1]	(100%,7) [3]
Scheduling-2	218	(2,9) [2]	(100%,14) [5]
Magic-Square-3	41	(30,57)	(0%,-)
Magic-Square-4	43643	(10121,5117)	(0%,-)
Magic-Square-5	-	(119909,354258)	(0%,-)

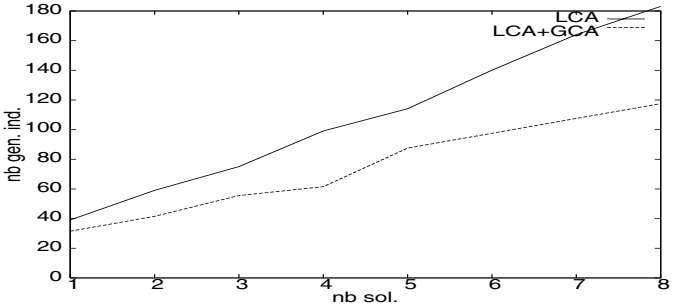


Fig. 2. Computation of different solutions

deviation on the different runs according to the number of moves, is very high compared to the combination LCA+GCA which has a more reliable behavior.

Computation time is not addressed here because, as mentioned before, our prototype shows the interest of our framework but does not provide an optimal use of the methods and of their hybridization. We should mention that the generation of one individual during the LCA stage is more costly than one move in the GCA stage. Note that this prototype does not include improvements such as global constraints or improved constraint propagation procedures and cannot be compared to commercial solvers. Moreover, the examples are encoded using a limited available language.

Finally, we calculate the computation cost needed to get several solutions with LCA+GCA on the *Magic-Square-3* problem. This problem has got 8 solutions and LCA finds all of them after exploring 182 nodes. Figure 2 shows the efficiency of the combination of the mechanisms which progress together to get quickly a set of distinct solutions. To get several distinct solutions is a really good asset compared to GCA alone and could be very interesting for number of problems.

4 Conclusion

We have presented a general evolutionary framework for solving CSP. This model is based on the collaboration between constraint propagation techniques and local search methods. The novelty of this approach relies in the uniform representation of the search space by populations which must adapt to different consistency criterions. It provides at the same time an exhaustive covering of the set of solutions and the use of some powerful methods to intensify the solution search. We showed on some examples that the combination between complete and incomplete methods, made easier by our framework, gets better results compared to an independent use of these techniques. Furthermore, the unified evolutionary concept allows us to consider various extensions and research ways as presented in Section 2.6.

References

1. K.R. Apt. The rough guide to constraint propagation. In *Proc. of CP'99*, volume 1713 of *LNCS*, pages 1–23, 1999. (Invited lecture).
2. Nicolas Barnier, Pascal Brisset. Combine and Conquer: Genetic Algorithm and CP for Optimization. In *Proc. of CP98*, LNCS 1520, p 463, 1998.
3. Frédéric Benhamou. Heterogeneous constraint solving. *Proceedings of ALP'96*, number 1139 in LNCS. Springer-Verlag, 1996.
4. Filippo Focacci, Francois Laburthe, and Andera Lodi. Local search and constraint programming. In Fred Glover and Gary Kochenberger, editors, *Handbook of Metaheuristics*, chap. 13. Kluwer, 2002.
5. F. Glover and M. Laguna. *Tabu Search*. Kluwer Academic Publisher, 1997.
6. D.E. Goldberg. *Genetic Algorithms for Search, Optimization, and Machine Learning*. Reading, MA:Addison-Wesley, 1989.
7. Narendra Jussien and Olivier Lhomme. Local search with constraint propagation and conflict-based heuristics. *Artificial Intelligence*, 139(1):21–45, 2002.
8. S. Kirkpatrick, C. Gelatt, and M. Vecchi. Optimization by Simulated Annealing : an experimental evaluation. *Science*, (220):671–680, 1983.
9. A. Mackworth. *Encyclopedia on Artificial Intelligence*, chapter Constraint Satisfaction. J. Wiley, 1987.
10. Kim Mariott and Peter J. Stuckey. *Programming with Constraints, An introduction*. MIT Press, 1998.
11. R. Mohr and T.C. Henderson. Arc and path consistency revisited. *Artificial Intelligence*, 28:225–233, 1986.
12. P. Shaw. *Using Constraint Programming and Local Search Methods to Solve Vehicle Routing Problems*. In Proc. of CP'98, pp 417-431, 1998.
13. V. Tam and P.J. Stuckey. Improving evolutionary algorithms for efficient constraint satisfaction. *The International Journal on Artificial Intelligence Tools*, 2(8), 1999.
14. E. Tsang. *Foundations of Constraint Satisfaction*. Academic Press, London, 1993.

Optimising Graph Partitions Using Parallel Evolution

R. Baños[1], C. Gil[1], J. Ortega[2], and F.G. Montoya[3]

[1] Dept. Arquitectura de Computadores y Electronica, Universidad de Almeria, La Cañada de San Urbano s/n, 04120 Almeria (Spain), {rbanos, cgil}@ace.ual.es,
[2] Dept. Arquitectura y Tecnologia de Computadores, Universidad de Granada Campus de Fuentenueva, Granada (Spain), julio@atc.ugr.es,
[3] Dept. Ingenieria Civil, Universidad de Granada Campus de Fuentenueva, Granada (Spain), pgil@ugr.es

Abstract. The graph partitioning problem consists of dividing the vertices of a graph into a set of balanced parts, such that the number of edges connecting vertices in different parts is minimised. Although different algorithms to solve this problem have been proposed in complex graphs, it is unknown how good the partitions are since the problem is, in general, NP-complete. In this paper we present a new parallel evolutionary algorithm for graph partitioning where different heuristics, such Simulated Annealing, Tabu Search, and some Selection Mechanisms are mixed. The efficiency of the new algorithm is compared with other previously proposed algorithms with promising results.

1 Introduction

The Graph Partitioning Problem (GPP) occurs in many areas, such as VLSI design [1,2], test patterns generation [3], data-mining [4], efficient storage of great data bases [5], etc. The problem is to find a partition of the vertices of a graph into K roughly equal parts, verifying that the number of edges connecting vertices in different subgraphs is minimised. As the problem is NP-complete [6], efficient procedures that provide solutions of high quality in a reasonable amount of time are very useful. Different strategies have been proposed to solve the graph partitioning problem, classified into combinatorial approaches [7,8], based on geometric representations [9,10], multilevel and clustering algorithms [11,12], and genetic algorithms [13]. There are also hybrid schemes [14,15,16] that combine different strategies.

In [15] a hybrid heuristic for circuit partitioning including an extension for standard graphs was proposed. In [16] this heuristic was extended to build a multilevel algorithm. In both papers, the mixed heuristic demonstrated a good performance. Nevertheless, still remain important aspects to study. One of them is to analyse the effect of modifying the parameters of Simulated Annealing, and

P. Liardet et al. (Eds.): EA 2003, LNCS 2936, pp. 91–102, 2004.

the other is to determine the importance of the initial partition in the optimisation process. In this work we analyse these aspects by taking advantage of parallel processing. Thus, we propose a new parallel algorithm that uses this hybrid heuristic with elitist selection mechanisms that improve the results obtained by the serial version.

Section 2 gives a more precise definition of the graph partitioning problem and describes the cost function used in the optimisation process. Section 3 describes the proposed algorithm, while Sect. 4 provides and analyses the experimental results with several test graphs. Finally, Sect. 5 gives the conclusions of the paper and suggests future work topics.

2 Graph Partitioning Problem

Given a graph $G=(V, E)$, where V is the set of vertices, with $|V|=n$, and E the set of edges that determines the connectivity among the vertices, the GPP consists of dividing V into K balanced parts, $V_1, V_2, ..., V_K$, so that $V_i \cap V_j = \phi$ for all $i \neq j$; and $\sum_{k=1}^{K} |V_k| = |V|$. The balance condition is defined as the maximum subdomain weight, S=max($|V_k|$), for k=1, ..., K, divided by the perfect balance, n/K. If a certain imbalance, x%, is allowed, then the GPP searches a partition such that the number of cuts is minimised subject to the constraint that S$\leq n/K*((100+x)/100)$. Whenever the vertices and edges have weights, $|v|$ denotes the weight of vertex v, and $|e|$ denotes the weight of edge e. All the test graphs used to evaluate the quality of our algorithm have vertices and edges with weight equal to one ($|v|$=1 and $|e|$=1). However, our procedure is able to process graphs with any weight values.

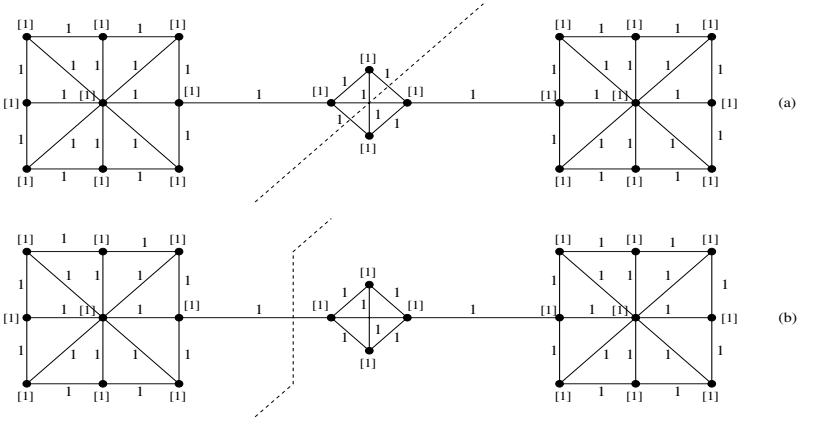


Fig. 1. Two ways to divide the graph

Figure 1 shows two possible partitions of a given graph. In Fig. 1(a) the graph is divided into two equally balanced parts, although the number of cuts is not minimised. On the other hand, Fig. 1(b) shows a partition with the optimal number of cuts. Nevertheless, this partition does not fulfill the requirement of load balancing. This example clearly shows the opposition of both objectives. In order to choose one or other solution, the following cost function could be considered:

$$c(s) = \alpha \cdot ncuts(s) + \beta \cdot \sum_{k=1}^{K} 2^{imbalance(k)}(s) \quad (1)$$

In this function, both objectives are taken into account. The first term is associated with the edges that are cut when the partition is made, while the second term corresponds to the penalty associated to the imbalance.

3 Parallel Evolutionary Algorithm

3.1 The Heuristic: Refined Mixed Simulated Annealing and Tabu Search (rMSATS)

In this section, a description of the hybrid heuristic proposed in [15] is given. Next, we detail the two phases of rMSATS.

i)Initial partitioning phase The first step to solve the GPP is to make an initial partitioning of the target graph. In this step rMSATS uses the procedure denominated Graph Growing Algorithm (GGA) [17]. This algorithm starts from a randomly selected vertex, which is assigned to the first subgraph, as their adjacent vertices. This recursive process is repeated until this subgraph reaches n/K vertices. From this point, the following visited vertices are assigned to a new subgraph, and the process is repeated until all the vertices are assigned to a subgraph. As the position of the initial vertex determines the structure of the initial partition, its random selection offers a very useful diversity in the search process. In our experiments we have considered several partitions starting from different vertices in order to explore the search space from different initial positions. The strategy used to choose the initial vertex in GGA is based on random interval selection. If p different processes are executed, the process P_i make the first partition starting from a randomly chosen vertex in the interval $\left[\frac{i}{p} \cdot n, \frac{i+1}{p} \cdot n\right[$. Although the test graphs are irregulars, this strategy assures diversity in the initial partitions. Experimental results indicate that the application of rMSATS with different initial partitions obtains best results than when only one initial vertex is used.

ii) Refinement phase. The application of an algorithm such as GGA is not enough to obtain a partition with sufficient quality. Therefore, it is necessary to perform a refinement phase that explores efficiently the search space. The problem with local search and hill climbing techniques is that the search may stop at local optima. In order to overcome this drawback, rMSATS mixes Simulated Annealing (SA) [18], and Tabu Search (TS) [19]. The use of both heuristics

results in a hybrid strategy that allows the search process to escape from local minima, while simultaneously prevents the occurrence of cycles. The idea of SA consists of taking a variable called temperature, t, whose value diminishes in the successive iterations based on a factor termed *Tfactor*. The variable t is included within the Metropolis function and acts simultaneously as a control variable for the number of iterations of the algorithm, and as a probability factor for a certain solution to be accepted. The decrease of t implies a reduction in the probability of accepting movements that worsen the cost function. On the other hand, the use of a limited neighbourhood when the search space is explored can cause the appearance of cycles. In order to avoid this problem, TS complements SA. Thus, when a worsening movement is accepted by the Metropolis rule, the vertex is moved and included in the tabu list. The set of vertices included in the tabu list cannot be moved in the current iteration, although they are removed from this list at the end of the next one. Experimental results [15] indicate that the use of both techniques improves the results obtained when SA or TS are applied separately.

In our experiments the effect of modifying the initial temperature, *Ti*, and the temperature decrement factor, *Tfactor*, is analysed (see Fig. 2). If *Ti* and *Tfactor* are randomly chosen an imbalance problem occurs because different number of iterations are executed in each process. In order to solve this problem we propose to calculate *Tfactor* in function of *Ti* and the number of iterations set for each process. Thus, all the processes start with different initial temperatures, and the algorithm calculates the value of *Tfactor* to assure that all processes execute the same number of iterations. In Fig. 2 several configurations of *Ti* and *Tfactor* are shown. In this example, a fixed number of 1000 iterations has been established. If *Ti* is high (line A) *Tfactor* is low, so the temperature decreases faster, while with lower values of *Ti* (line J) *Tfactor* is higher, and the temperature decreases more slowly.

3.2 The Heuristic within a Parallel Scheme: PrMSATS

In the previous section, a description of the parameters of rMSATS has been given. In order to explore the search space using several initial partitions and annealing parameters, parallel processing has been considered. Furthermore, parallel processing is very useful because graph partitioning is a NP-complete problem, and the graphs appearing in real applications are usually large.

In [20] parallel evolutionary algorithms are classified as: (1) global single-population master-slave, where a single population is used, and the evaluation of the fitness is distributed among several processors; (2) single-population fine-grained, which consists of a spatially-structured population, where selection and mating are restricted to a small neighbourhood, but neighbourhoods overlapping allows some interaction among all the individuals; and (3) multiple-population coarse-grained, where several subpopulations which exchange (migrate) individuals occasionally.

The evolution of the population in rMSATS is not based on the application of crossover and mutation operators to each individual, but each execution of

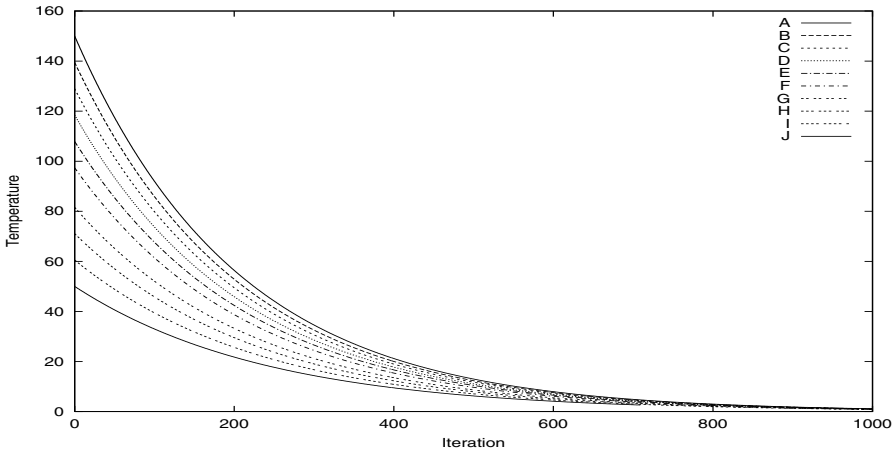


Fig. 2. Parallel variation of the temperature using different annealing parameters

rMSATS independently uses its own parameters over a single solution. For this reason, and considering the high cost of the migrations due to the large size of the test graphs, a reduced number of communications are considered.

In what follows, we detail the characteristics of PrMSATS: i) PrMSATS uses one processor for each individual of the population; ii) each processor applies GGA to the target graph starting from a random vertex chosen as we have indicated previously; iii) each processor uses a different initial temperature in according to its identifier, *id*, as we explain in what follows. An interval of initial temperatures $[Ti\_Min, Ti\_Max]$ is set. Then, the process 1 starts in Ti_Min, the process P starts in Ti_Max, and the others are evenly distributed along this interval; iv) Several migration policies among processes have been considered. The first policy, Fig. 3(a), is based on communicating only the processes in the neighbourhood, it is, the process P_i only can interchange information alternatively with P_{i-1} and P_{i+1}. This policy corresponds to a fine-grained implementation. The second policy, Fig. 3(b), is based on an incremental migration, it is, the number of processors that interchange information increases according to the number of previous communications. It is based on that at the beginning the value of t is high, reason why the migrations must be reduced to a certain number of individuals, while in the next iterations the value of t decreases, and therefore the probability of accepting bad movements. Thus, the best individuals of the population must be migrated to a higher number of processors. The third migration policy, Fig. 3(c), is based on broadcasting the best individual of the population to the other processors.

An elitist tournament selection criterion has been used. If p processes are evaluated in a given iteration, the winner of the tournament sends its current solution to the others, that must continue applying rMSATS with this solution, and their own values of t and $Tfactor$. Using this elitist strategy, the algorithm

searches for the best solutions using different SA parameters, instead of wasting the computational resources exploring solutions with worse fitness. An important question to analyse is the migration frequency. Taking into account the characteristics of rMSATS, and in order to avoid the premature convergence of the algorithm, the communications have been established as we detail in what follows. Let C be the number of communications, and R the number of refinement iterations. Then, the communications are performed only at the iterations $\{\frac{T}{C}, 2 \cdot \frac{T}{C}, \ldots, C \cdot \frac{T}{C}\}$. In our experiments a value of C=5 has been considered. Next, PrMSATS procedure is detailed.

```
begin_PrMSATS
  generate an initial solution s0;
  s=s0; s*=s0; s'=s0;
  select an initial temperature t0>0; t=t0;
  select the temperature reduction factor, tfactor;
  n_failures=0;  iteration=0;
  while ((iteration<max_iteration)and (t>0)and(n_failures<max_failures))
  begin
    if {(iteration == 1/2 · max_iteration)|| ... ||(iteration == (2^C − 1)/2^C · max_iteration)} then
    begin
      Elitist_tournament_selection(migration_policy);
    end_if
    for each (boundary_vertex) do
    begin
      obtain a neighbour solution s⁻ from the current solution s;
      movement_cost=c(s⁻) − c(s);
      if (Metropolis(movement_cost,t)) then
      begin
        s=s⁻;
        if (movement_cost > 0) then
          include the movement from s⁻ to s as tabu;
        if (c(s⁻)<c(s')) then
          s'=s⁻;
      end_if
    end_for
    if (c(s')<(c(s*))) then
    begin
      n_failures=0;
      s*=s';
    end_if
    else n_failures=n_failures+1;
    s=s';
    update tabu list;
    t=tfactor·t; iteration=iteration+1;
  end_while
  s* is the solution;
end_PrMSATS
 where :
   s0 : initial solution;
   s : current solution;
   s⁻ : neighbour solution obtained from s solution;
   s' : temporal best solution;
   s* : final best solution;
```

4 Experimental Results

The executions of our algorithm were performed by using test graphs with different sizes and topologies. These graphs belong to a public domain set that is frequently used to compare and evaluate graph partitioning algorithms. Table

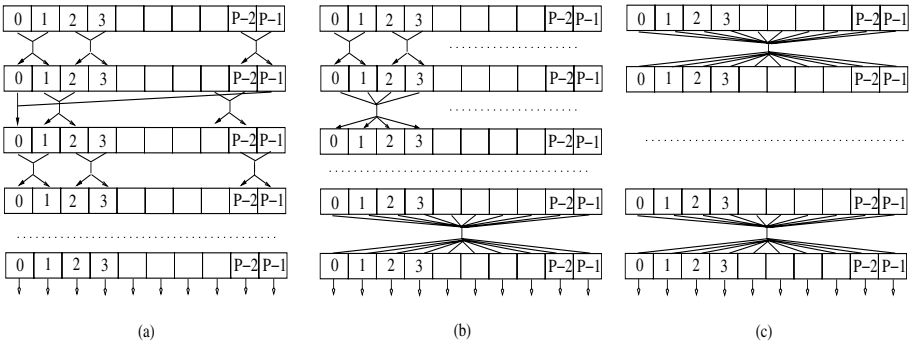


Fig. 3. Migration policies used: (a) Fine-grained, (b) Incremental, (c) Broadcast

1 gives a brief description of them: number of vertices, number of edges, maximum connectivity *(max)* (number of neighbours to the vertex with the highest neighbourhood), minimum connectivity *(min)*, average connectivity *(avg)*, and file size.

Table 1. Set of test graphs used to evaluate our procedure

Graph	$\|V\|$	$\|E\|$	*min*	*max*	*avg*	*description*	*File Size (KB)*
add20	2395	7462	1	123	6.23	20-bit adder	63
add32	4960	9462	1	31	3.82	32-bit adder	90
wing_nodal	10937	75488	5	28	13.80	3D mesh	768
fe_4elt2	11143	32818	3	12	5.89	2D mesh	341
vibrobox	12328	165250	8	120	26.81	Vibroacoustic matrix	1679
memplus	17758	54196	1	573	6.10	Vibroacoustic matrix	536
cs4	22499	43858	2	4	3.90	3D mesh	506
bcsstk32	44609	985046	1	215	44.16	3D mesh	11368
brack2	62631	366559	3	32	11.71	3D mesh	4358
fe_tooth	78136	452591	3	39	11.58	3D mesh	5413
fe_rotor	99617	662431	5	125	13.3	3D mesh	7894
fe_ocean	143437	409593	1	6	5.71	3D mesh	5242
wave	156317	1059331	3	44	13.55	3D mesh	13479

These test graphs, together with the best solutions known for them, can be found at [21]. These solutions indicate the number of cuts classified by levels of imbalance (0%, 1%, 3% and 5%). Thus, the reduction in the number of cuts is considered as an objective, while the imbalance degree (less than 5% in our experiments) is considered as a restriction. Under these conditions, the cost function described in (1), has parameters α=1, and β=0; with imbalance(k)$\leq$5 for all k in the interval [1,K].

Table 2 shows the results obtained by rMSATS using 1, 5 and 20 different initial partitions, obtained by GGA from different initial vertices. The same temperature values, Ti=100 and $Tfactor$=0.995, are used. With these parameters, the number of iterations of the algorithm is 1500. As we can see the use of dif-

ferent initial partitions provides a diversity that allows the improvement of the solution in most cases compared with the use of less initial partitions.

In Fig. 4 the results obtained with different ranges of Ti are shown. In these executions 20 different initial partitions have been considered. Results indicate that the use of different values of Ti, and $Tfactor$ (calculated for 1500 iterations) obtain better results than when the same parameters, Ti=100 and $Tfactor$=0.995, are used. Nevertheless, none of the three selected intervals obtain a clear improvement compared with the others.

Table 3 summarises the results obtained by the three migration schemes implemented in PrMSATS, with 20 individuals. Results show that the first migration scheme improves the results in 11% of the cases, the second one in the 31%, and the third in the 35%, while in the rest of cases (23%) the results are equal. If we compare these results with those previously showed in Table 2, we see that in most cases PrMSATS improves rMSATS.

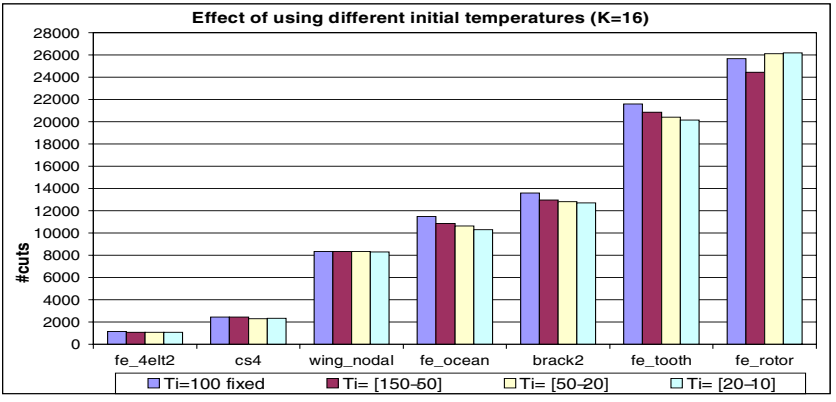


Fig. 4. Performance of PrMSATS with different temperature intervals.

Finally, Table 4 shows the best results of PrMSATS (Table 3) compared with the best known solutions obtained by other algorithms [21], with imbalance less than 5%. In 9 cases, at least one of the three migration schemes implemented in PrMSATS improves the previously best known solutions, while in other 4 cases the results are similar. The runtime of PrMSATS can oscillate between few minutes and several hours, depending of the test graph considered. Nevertheless, the runtimes are comparable with those of the algorithms included in [21].

5 Conclusions

In this paper a new parallel evolutionary algorithm for graph partitioning has been presented. This parallel algorithm mixes SA, TS and an elitist selection

Table 2. Effect of using different number of initial partitions

Graph	algorithm	K=2		K =4		K =8		K=16		K =32	
		cuts	imb.	cuts	imb.	cuts	imb.	cuts	imb.	cuts	imb.
add20	rMSATS[1]	843	5	1349	5	1791	5	2159	5	2637	4
	rMSATS[5]	715	5	**1223**	5	1769	5	2128	5	2499	4
	rMSATS[20]	**701**	5	1232	5	**1749**	5	**2122**	5	**2481**	4
add32	rMSATS[1]	51	2	84	5	154	5	165	5	279	5
	rMSATS[5]	11	5	82	3	123	5	171	3	285	5
	rMSATS[20]	**10**	0	**55**	2	**88**	5	**158**	5	**243**	5
wing nodal	rMSATS[1]	1674	5	3626	3	5485	5	8370	5	11905	5
	rMSATS[5]	**1673**	5	3631	5	5442	5	8394	5	11882	5
	rMSATS[20]	**1673**	5	**3545**	5	**5385**	5	**8341**	5	**11831**	5
fe 4elt2	rMSATS[1]	206	0	385	3	752	5	1223	5	1805	5
	rMSATS[5]	**130**	0	**372**	4	715	5	1209	5	1767	5
	rMSATS[20]	**130**	0	394	3	**704**	5	**1170**	5	**1704**	5
vibro box	rMSATS[1]	12229	5	**20777**	5	31092	5	34460	5	42118	5
	rMSATS[5]	11767	5	20923	5	28688	5	34115	5	41418	5
	rMSATS[20]	**10714**	3	20868	5	**26201**	5	**33476**	5	**41270**	5
mem plus	rMSATS[1]	7600	5	10795	5	12644	5	14158	5	15492	5
	rMSATS[5]	7402	4	10554	5	12687	5	14014	5	**15129**	5
	rMSATS[20]	**7119**	5	**10401**	5	**12460**	5	**13858**	5	15306	5
cs4	rMSATS[1]	861	4	1554	4	2261	5	2807	5	3355	5
	rMSATS[5]	446	4	1094	5	1914	5	2651	5	3405	5
	rMSATS[20]	**421**	4	**1067**	5	**1802**	5	**2472**	5	**3287**	5
bcsstk32	rMSATS[1]	10422	1	29685	5	52697	5	68424	5	86580	5
	rMSATS[5]	**10248**	2	**21181**	5	45154	5	70488	5	85455	5
	rMSATS[20]	10807	4	21406	5	**42607**	5	**61386**	5	**82726**	5
brack2	rMSATS[1]	711	1	3874	5	**8028**	5	14816	5	20676	5
	rMSATS[5]	711	2	3743	5	8490	5	14295	5	20497	5
	rMSATS[20]	**683**	4	**3458**	5	8304	5	**13614**	5	**20208**	5
fe tooth	rMSATS[1]	3966	5	10486	5	15920	5	23375	5	32624	5
	rMSATS[5]	**3896**	5	10332	5	16707	5	22602	5	28514	5
	rMSATS[20]	4054	5	**10117**	5	**15695**	4	**21596**	5	**27642**	5
fe rotor	rMSATS[1]	2058	2	9445	5	17735	5	28694	5	39102	5
	rMSATS[5]	1968	3	7632	3	**15875**	5	26222	5	38192	5
	rMSATS[20]	**1956**	4	**7535**	4	16896	5	**25672**	5	**37770**	5
fe ocean	rMSATS[1]	406	2	**2670**	5	**6583**	5	12232	5	18482	5
	rMSATS[5]	356	3	2856	5	7238	5	12029	5	17696	5
	rMSATS[20]	**317**	3	2954	5	6962	5	**11501**	5	**16030**	5
wave	rMSATS[1]	15496	5	30899	5	47786	5	58013	5	74920	5
	rMSATS[5]	10132	5	**28235**	5	44929	5	57533	5	76448	5
	rMSATS[20]	**8670**	5	28474	5	**39365**	5	**53437**	5	**72380**	5

Table 3. Results obtained by the three migration policies

Graph	algorithm	K=2		K=4		K=8		K=16		K=32	
		cuts	imb.	cuts	imb.	cuts	imb.	cuts	imb.	cuts	imb.
add20	PrMSATS^A	**684**	5	1249	5	**1772**	5	**2125**	5	2462	4
	PrMSATS^B	**684**	5	**1245**	5	1773	5	**2125**	5	**2458**	4
	PrMSATS^C	700	5	**1245**	5	1784	5	2132	5	2498	4
add32	PrMSATS^A	**11**	4	**54**	3	88	5	139	5	**238**	3
	PrMSATS^B	**11**	4	**54**	3	88	5	140	5	**238**	5
	PrMSATS^C	**11**	1	57	4	**85**	5	**136**	5	**238**	3
wing nodal	PrMSATS^A	**1670**	5	**3616**	4	5391	5	**8323**	5	11826	5
	PrMSATS^B	**1670**	5	**3616**	5	**5389**	5	8326	5	11821	5
	PrMSATS^C	1672	5	3937	5	5467	5	8372	5	**11753**	5
fe 4elt2	PrMSATS^A	**130**	0	350	2	**707**	5	1132	5	**1689**	5
	PrMSATS^B	**130**	0	350	1	**707**	5	**1111**	5	1725	5
	PrMSATS^C	**130**	0	**349**	0	781	3	1169	5	1715	5
vibro box	PrMSATS^A	**10310**	0	20545	5	27091	5	33243	5	41518	5
	PrMSATS^B	**10310**	3	20532	5	27088	5	33242	5	**41513**	5
	PrMSATS^C	10916	0	**20531**	5	**26937**	5	**33240**	5	41603	5
mem plus	PrMSATS^A	7254	5	10488	5	12291	5	13702	5	15307	5
	PrMSATS^B	7252	5	10482	5	**12286**	5	**13701**	5	**15304**	5
	PrMSATS^C	**7217**	5	**10475**	5	12360	5	14043	5	15402	5
cs4	PrMSATS^A	365	5	1042	4	1737	5	2457	5	3289	5
	PrMSATS^B	**364**	5	**1036**	4	**1727**	5	**2456**	5	3328	5
	PrMSATS^C	377	5	1041	5	1737	5	2571	5	**3318**	5
bcsstk32	PrMSATS^A	**10545**	5	22268	5	40076	5	63267	5	82048	5
	PrMSATS^B	**10545**	5	22269	5	40076	5	63267	5	82048	5
	PrMSATS^C	**10545**	5	**21778**	5	**39902**	5	**63110**	5	**80861**	5
brack2	PrMSATS^A	672	4	**2793**	5	7808	5	13856	5	18891	5
	PrMSATS^B	672	4	**2793**	5	7803	5	**13854**	5	**18889**	5
	PrMSATS^C	**665**	4	3720	3	**7291**	5	14409	5	18954	5
fe tooth	PrMSATS^A	3878	5	9036	5	15520	5	21675	5	28414	5
	PrMSATS^B	3898	1	**8972**	5	15575	5	21670	5	**28401**	5
	PrMSATS^C	**3853**	3	10351	5	**15053**	5	**21175**	5	28842	5
fe rotor	PrMSATS^A	**1966**	3	7477	3	**15873**	5	**25585**	5	37579	5
	PrMSATS^B	1968	2	**7475**	3	15874	5	25586	5	**37547**	5
	PrMSATS^C	1967	2	8006	5	17780	5	26146	5	38874	5
fe ocean	PrMSATS^A	**317**	3	2864	5	6718	5	10697	5	17248	5
	PrMSATS^B	**317**	3	2863	5	6731	5	**10679**	5	17329	5
	PrMSATS^C	325	3	**2515**	5	**6272**	5	10834	5	**16593**	5
wave	PrMSATS^A	8614	5	27060	5	**38660**	5	**53520**	5	67053	5
	PrMSATS^B	**8613**	5	27056	5	38662	5	**53520**	5	**66999**	5
	PrMSATS^C	9769	1	**25355**	2	43018	5	55558	5	70019	5

mechanism in order to improve the quality of the solutions. The first conclusion obtained is that the diversity of the initial partitions is essential in the search process. As the selection of adequate parameters of rMSATS is very difficult due to the characteristics of the problem, different values of *Ti* and *Tfactor* have been considered. The implemented elitist selection strategies allow the computational resources to be concentrated in the exploration of the best solutions. In general, the quality of the partitions is better in PrMSATS than in rMSATS.

Our future work in this area is focused to parallelise the multilevel version of rMSATS, in order to improve the solutions. On the other hand, the treatment of the problem using multiobjective optimisation techniques will be considered.

Table 4. Comparison of PrMSATS with best known solutions

Graph	algorithm	*K*=2		*K* =4		*K* =8		*K*=16		*K* =32	
		Cuts	imb.	cuts	imb.	cuts	imb.	cuts	imb.	cuts	imb.
add20	PrMSATS	684	5	1245	5	1772	5	**2125**	5	**2458**	4
	Graph_Archive	**618**	1	**1184**	5	**1705**	5	2186	5	2758	3
add32	PrMSATS	11	4	54	3	85	5	136	5	238	3
	Graph_Archive	**10**	0	**33**	5	**69**	5	**117**	5	**212**	5
wing nodal	PrMSATS	**1670**	5	3616	4	5389	5	8323	5	**11753**	5
	Graph_Archive	**1670**	5	**3566**	5	**5387**	5	**8316**	5	12024	5
fe 4elt2	PrMSATS	**130**	0	**349**	4	707	5	1111	5	1689	5
	Graph_Archive	**130**	0	**349**	0	**597**	5	**1007**	3	**1651**	3
vibro box	PrMSATS	**10310**	0	20531	5	26937	5	33240	5	41513	5
	Graph_Archive	**10310**	0	**19245**	0	**24158**	5	**31695**	5	**41176**	5
mem plus	PrMSATS	7217	5	10475	5	12360	5	13701	5	15304	5
	Graph_Archive	**5353**	5	**9427**	5	**11939**	1	**13279**	5	**14384**	5
cs4	PrMSATS	364	5	1036	4	1727	5	2456	5	3318	5
	Graph_Archive	**363**	3	**936**	3	**1472**	3	**2126**	3	**3080**	5
bcsstk 32	PrMSATS	10545	5	21778	5	39902	5	63110	5	80861	5
	Graph_Archive	**4667**	0	**9728**	3	**21307**	3	**38320**	5	**62691**	5
brack2	PrMSATS	**665**	4	**2793**	5	7291	5	13854	5	18889	5
	Graph_Archive	668	5	2808	5	**7080**	3	**11958**	3	**17952**	3
fe tooth	PrMSATS	**3853**	3	8972	5	15053	5	21175	5	28401	5
	Graph_Archive	3982	0	**7152**	5	**12646**	5	**18435**	1	**26016**	5
fe rotor	PrMSATS	**1966**	3	**7475**	3	15873	5	25585	5	37547	5
	Graph_Archive	1974	5	8097	0	**13184**	5	**20773**	1	**33686**	1
fe ocean	PrMSATS	317	3	2515	5	6272	5	10679	5	16593	5
	Graph_Archive	**311**	3	**1704**	3	**4019**	3	**7838**	3	**12746**	3
wave	PrMSATS	**8613**	5	25355	2	38660	5	53520	5	66999	5
	Graph_Archive	8868	5	**18058**	1	**30583**	5	**44625**	5	**63725**	5

References

1. Alpert, C.J., and Kahng, A., Recent Developments in Netlist Partitioning: A Survey. Integration: the VLSI Journal, 19/1-2 (1995) 1-81.
2. Banerjee, P., Parallel Algorithms for VLSI Computer Aided Design. Prentice Hall, Englewoods Cliffs, New Jersey, 1994.
3. Gil, C.; Ortega, J., and Montoya, M.G., Parallel VLSI Test in a Shared Memory Multiprocessors. Concurrency: Practice and Experience, 12/5 (2000) 311-326.
4. Mobasher, B., Jain, N., Han, E.H., Srivastava J. Web mining : Pattern discovery from world wide web transactions. Technical Report TR-96-050, Department of computer science, University of Minnesota, Minneapolis, (1996).
5. Shekhar S. and DLiu D.R.. Partitioning similarity graphs: A framework for declustering problems. Information Systems Journal, 21/6 (1996) 475-496.
6. Garey, M.R., and Johnson, D.S, Computers and Intractability: A Guide to the Theory of NP-Completeness, W.H. Freeman & Company. San Francisco, 1979.
7. Kernighan, B.W., and Lin, S., An Efficient Heuristic Procedure for Partitioning Graphics, The Bell Sys. Tech. Journal, (1970) 291-307.
8. Fiduccia, C., and Mattheyses, R., A Linear Time Heuristic for Improving Network Partitions, In Proc. 19th IEEE Design Automation Conference, (1982) 175-181.
9. Simon, H.D. and Teng, S., How Good is Recursive Bisection?, SIAM J. Scientific Computing, 18/5 (1997) 1436-1445.
10. Gilbert, J., Miller, G., and Teng, S., Geometric Mesh Partitioning: Implementation and Experiments, In Proc. 9th Int. Parallel Processing Symposium,(1995) 418-427.
11. Karypis, G. and Kumar V., Multilevel K-way Partitioning Scheme for Irregular Graphs. Journal of Parallel and Distributed Computing, 48/1 (1998) 96-129.
12. Cong, J., and Smith, M., A Parallel Bottom-up Clustering Algorithm with Applications to Circuit Partitioning in VLSI Design, In Proc. ACM/IEEE Design Automation Conference, (1993) 755-760.
13. Bui, T. N. and Moon, B., Genetic Algorithm and Graph Partitioning. IEEE Transactions on Computers, 45/7 (1996) 841-855.
14. Soper, A.J., Walshaw, C., and Cross, M., A Combined Evolutionary Search and Multilevel Optimisation Approach to Graph Partitioning, Mathematics Research Report 00/IM/58, University of Greenwich, (2000).
15. Gil, C.,Ortega, J.,Montoya, M.G.,and Baños R.,A Mixed Heuristic for Circuit Partitioning.Computational Optimization and Applications Journal.23/3 (2002) 321-340.
16. Baños R., Gil, C., Ortega, J., Montoya, F.G., Multilevel Heuristic Algorithm for Graph Partitioning. In Proc. 3rd European Workshop on Evolutionary Computation in Combinatorial Optimization. LNCS, 2611 (2003) 143-153.
17. Karypis, G. and Kumar V., A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs. SIAM Journal on Scientific Comput. 20/1 (1998) 359-392.
18. Kirkpatrick, S., Gelatt, C.D., and Vecchi, M.P., Optimization by simulated annealing. Science, 220, (1993) 671-680.
19. Glover, F. and Laguna M., Tabu Search. In Modern Heuristic Techniques for Combinatorial Problems. C.R. Reeves (eds.), Blackwell, London, 70-150, 1993.
20. Cantu-Paz, E., A Survey of Parallel Genetic Algorithms. Technical Report IlliGAL 97003, University of Illinois at Urbana-Champaign, 1997.
21. Graph Partitioning Archive. "http://www.gre.ac.uk/ c.walshaw/partition/". URL time: April 25th, 2003, 20^{45}.

Recombination Operators for Satisfiability Problems

Frédéric Lardeux, Frédéric Saubion, and Jin-Kao Hao

LERIA, University of Angers
2 Bd Lavoisier, 49045 Angers Cedex 01, France
{Frederic.Lardeux,Frederic.Saubion,Jin-Kao.Hao}@univ-angers.fr

Abstract. In this paper, we present several recombination operators that are specially designed for SAT problems. These operators take into account the semantic information induced by the structure of the given problem instance under consideration. Studies are carried out to assess the relative performance of these recombination operators on the one hand, and to show the high effectiveness of one of them when it is embedded into a hybrid genetic local search algorithm on the other hand.

1 Introduction

The satisfiability problem (SAT) [6] consists in finding a truth assignment that satisfies a well-formed Boolean expression. An instance of the SAT problem is defined by a set of boolean variables (also called atoms) $\mathcal{X} = \{x_1, ..., x_n\}$ and a boolean formula $\phi: \{0,1\}^n \rightarrow \{0,1\}$. A literal is a variable or its negation. A (truth) assignment is a function $v: \mathcal{X} \rightarrow \{0,1\}$. The formula is said to be satisfiable if there exists an assignment satisfying ϕ and unsatisfiable otherwise. The formula ϕ is in conjunctive normal form (CNF) if it is a conjunction of clauses where a clause is a disjunction of literals. In this paper, ϕ is supposed to be in CNF.

SAT is originally stated as a decision problem but we are also interesting here in the MAX-SAT problem which consists in finding an assignment which satisfies the maximum number of clauses in ϕ.

SAT and its variants have many practical applications (VLSI test and verification, consistency maintenance, fault diagnosis, planning...). During the last decade, several improved solution algorithms have been developed and important progress has been achieved. These algorithms can be divided into two main classes:

Complete algorithms: Designed to solve the initial decision problem, the most powerful complete algorithms are based on the Davis-Putnam-Loveland procedure [2]. They differ essentially by the underlying heuristic used for the branching rule [13,19]. Specific techniques such as symmetry-breaking, backbone detecting or equivalence elimination are also used to reinforce these algorithms [1,4,12].

P. Liardet et al. (Eds.): EA 2003, LNCS 2936, pp. 103–114, 2004.

Incomplete algorithms: They are mainly based on local search [8,10,11,15] and evolutionary algorithms [3,5,14,7,9]. Walksat [15] and UnitWalk [10] are famous examples of incomplete algorithms. Incomplete algorithms represent one of the most powerful approaches for finding models of very large SAT instances and for solving large scale MAX-SAT instances.

In this paper, we are interested in the hybrid genetic local search approach for SAT and in particular the design of recombination (crossover) operators. We introduce two new recombination operators taking into account the constraint structure of the given problem instance under consideration.

To assess the performance of these recombination operators, we compare them with two other crossover operators. We also study the combined effect of our crossover operators when they are embedded into a genetic local search algorithm. We show experimentally that such a hybrid algorithm is able to produce highly competitive results compared with other state of the art evolutionary algorithms as well as with WalkSat and UnitWalk, two leading SAT solvers.

In the next section, we present the hybrid evolutionary framework for SAT. We propose then an analysis of the recombination operators and of the interaction between LS and recombination operators. At last, we compare our hybrid algorithm to other well-known algorithms in order to evaluate its performance.

2 Hybrid Evolutionary Algorithm Framework

In this section, we define the main lines of a hybrid evolutionary algorithm obtained by mixing a recombination stage with a local search (LS) process. Given an initial population, the first step consists in selecting some best elements (i.e. assignments) according to a fitness function *eval*. Then, recombinations are performed on this selected population. Each child is individually improved using a local search process and inserted under certain conditions in the population. The general algorithm is described in figure 1. It is clear that this algorithm can be adjusted by changing the selection function, the recombination operator or the local search method but also by modifying the insertion conditions.

Representation. The most obvious way to represent an individual for a SAT instance with n variables is a string of n bits where each variable is associated to one bit. Other representation schemes are discussed in [7]. Therefore the search space is the set $\mathcal{S} = \{0,1\}^n$ (i.e. all the possible strings of n bits) and an individual X obviously corresponds to an assignment. $X|i$ denotes the truth value of the i^{th} atom. Given an individual X and a clause c, we use the boolean function $sat(X,c)$ to denote the fact that the assignment associated to X satisfies the clause c.

Fitness Function. Given a formula ϕ and an individual X, the fitness of X is defined to be the number of clauses which are not satisfied by X:

$$\begin{aligned} eval\colon \mathcal{S} &\to \mathbb{N} \\ X &\mapsto card(\{c | \neg sat(X,c) \wedge c \in \phi\}) \end{aligned}$$

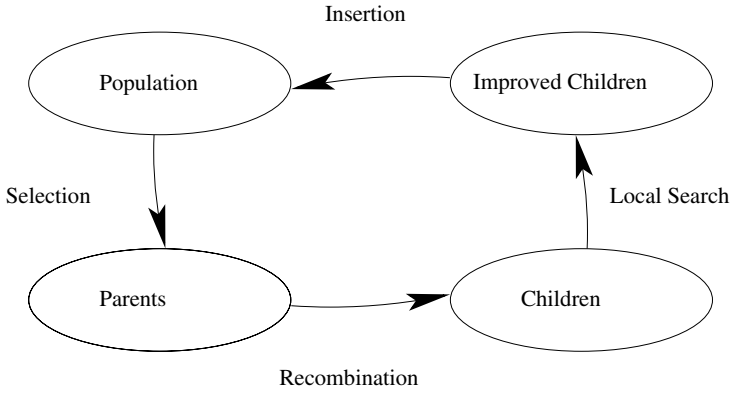


Fig. 1. Algorithm Scheme

where *card* denotes as usual the cardinality of a set. The smallest value of this function equals 0 if ϕ is satisfiable and an individual having this fitness value corresponds to a satisfying solution.

Selection Process. The selection operator is a function which takes as input a given population and extracts some individuals which will serve as parents for the recombination stage. To insure an efficient search, it is necessary to keep some diversity in the population. Indeed, if the selected parents are too similar, some region of the search space $\mathcal{S}$ will not be explored.

Recombination. This operator creates new assignments from two selected parent assignments. Classical random crossover may be used. However, as we show later in the paper, specific crossovers that are meaningful to the SAT problem may be much more powerful.

Local Search. The local search process improves a configuration by a sequence of small changes in order to reach a local optimum according to the evaluation function, a good neighborhood function [18] and some mechanisms like tabu list, random walk, and so on.

Insertion. An improved child can be inserted according to different conditions (e.g. if it is better than any individual of the population). These conditions are important with respect to the average equality and the diversity of the population. A child is inserted in the population if it is better than the worst of the sub-population of possible parents.

3 Recombination Operators

A recombination operator (also called crossover) has to take into account as much as possible the semantics of the individuals in order to control the general search process and to obtain an efficient algorithm. In SAT and MAX-SAT, the variables are the atoms and a constraint structure is induced by the clauses of

the given formula. Therefore, while the representation focuses on the atoms, an efficient crossover should take into account this whole constraint structure.

We first define a function allowing to change the value of the variables:

$$\begin{aligned} flip\colon \{0,1\} &\to \{0,1\} \\ x &\mapsto 1-x \end{aligned}$$

Then, we define an improvement function:

$$\begin{aligned} imp\colon \mathcal{S} \times \mathbb{N} &\to \mathbb{N} \\ (X,i) &\mapsto card(\{c \mid sat(X[i \leftarrow flip(X|i)],c) \wedge \neg sat(X,c)\}) \\ &\quad -card(\{c \mid \neg sat(X[i \leftarrow flip(X|i)],c) \wedge sat(X,c)\})\,. \end{aligned}$$

This function computes the improvement obtained by flipping the i^{th} component of X and was previously introduced in GSAT and Walksat [16,15]. It corresponds to the gain of the solution according to the function *eval* (i.e. the number of false clauses which become true by flipping the atom i minus the number of satisfied clauses which become false). Remark that if this number is negative then the number of false clauses increases if the flip is performed.

In order to take into account the specific structures of a problem, we will use the relations over the variables induced by the clauses. There are three possibilities as to the satisfaction of a clause under the two parents:

- the clause is satisfied in the two parents,
- the clause is unsatisfied in the two parents,
- the clause is satisfied in only one parent.

The aim of our recombination is then to obtain a child which benefits from the parents but with the maximum number of true clauses. Therefore, one should try to correct false clauses and to maintain true ones.

When a clause c is false for the two parents, a possible solution to turn c to true is to flip a variable of c [1]. Therefore, the child receives an opposite value for this variable with respect to its parents. Unfortunately, this action may produce false clauses. To limit this number of appearing false clauses, the choice of the variable is guided by the *imp* evaluation function.

When a clause c is true for the two parents, we may copy in the child all the values assigned to the variables of c in one of the parent in the child. Unfortunately, this action would only take into account the structure of the chosen parent. To be fair, we should copy values of variables coming from both parents. But, these values can be different. A solution is to select the variable whose flip has the smallest impact and to put its value such that the corresponding literal is true in c. Since only one variable is necessary to maintain this clause true, we may again guide this operation with the *imp* function.

[1] If a clause is false for both parents, then all the variables appearing in this clause have necessarily the same value in both parents. This comes from the fact that a clause can be false only if all of its literals are false.

Finally, when a clause c is true for one parent and false for the other, the solution proposed by Fleurent and Ferland [5] is: "The corresponding variables [to this clause] are assigned values according to the parent satisfying the identified clause."

It is clear that the order in which the clauses are traversed is important. Here, they are traversed in the same order that they appear in the studied benchmark.

We now present the definitions of four crossovers. Three of them are structured operators using the previous remarks. We present also the classical uniform crossover, which is used as a reference for comparisons. Each operator is a function $cross: \mathcal{S} \times \mathcal{S} \to \mathcal{S}$ (i.e. two parents produce one child).

Let X and Y be two parents and Z be the resulting child.

Uniform Crossover
Each variable of Z takes the value of X or Y with equal probability.

Corrective Clause Crossover (CC)
For each clause c such that $\neg sat(X,c) \wedge \neg sat(Y,c) \wedge \neg sat(Z,c)$ and for all positions i such that the variable x_i appears in c, we compute $\sigma = imp(X,i) + imp(Y,i)$ and we set $Z|k = flip(X|k)$ where k is the position such that σ is maximum. All the variables of Z with no value take the value of X or Y with equal probability.

Corrective Clause and Truth Maintenance Crossover (CCTM)
For each clause c such that $\neg sat(X,c) \wedge \neg sat(Y,c) \wedge \neg sat(Z,c)$ and for all positions i such that the variable x_i appears in c, we compute $\sigma = imp(X,i) + imp(Y,i)$ and we set $Z|k = flip(X|k)$ where k is the position such that σ is maximum. For all the clauses c such that $sat(X,c) \wedge sat(Y,c)$ and for all positions i such that the variable x_i appears in c and its associated literal is true at least in one parent, we compute $\sigma = imp(X,i) + imp(Y,i)$ and we value $Z|k$ such that $sat(Z,c)$ where k is the position such that σ is minimum. All the variables of Z with no value take the value of X or Y with equal probability.

Fleurent and Ferland Crossover [5] (F&F)
For each clause c such that $sat(X,c) \wedge \neg sat(Y,c)$ (resp. $\neg sat(X,c) \wedge sat(Y,c)$) and for all the positions i such that the variable $x_i \in c$, $Z|i = X|i$ (resp. $Z|i = Y|i$). All the variables of Z with no value take the value of X or Y with equal probability.

Notice that CC, CCTM and F&F crossovers differ on the use of the truth values of the clauses induced by the parents. As mentioned above, the key idea is to correct false clauses, to preserve true clauses and to maintain the structure of the assignments. In order to study the characteristics of the different operators, we reduce the algorithm to a sequence of recombination stages on a population with or without the selection process between two consecutive stages. The tests are realized with a random 3-sat instance (`f500.cnf`) with 500 variables and a ratio of clauses-to-variables of 4.3. Two measures are used: the profile of *number of crossovers* vs. *fitness* and *number of crossovers* vs. *population entropy.*

Entropy expresses the diversity of the population. The smallest value is 0, indicating that all the individual of the population are the same. The largest value is 1, indicating that all the individuals are different.

Definition 1. *Entropy [5] Let n_{ij} = number of times where the variable i is set to j in the population P.*

$$entropy(P) = \frac{\sum_{i=1}^{n} \sum_{j=0}^{1} \frac{n_{ij}}{card(P)} log \frac{n_{ij}}{card(P)}}{nlog2}$$

The population size is 100 and the sub-population of possible parents has a size of 15. Two parents are randomly chosen in this sub-population. A child is inserted in the population if its fitness is better than the fitness of the worst individual of the sub-population.

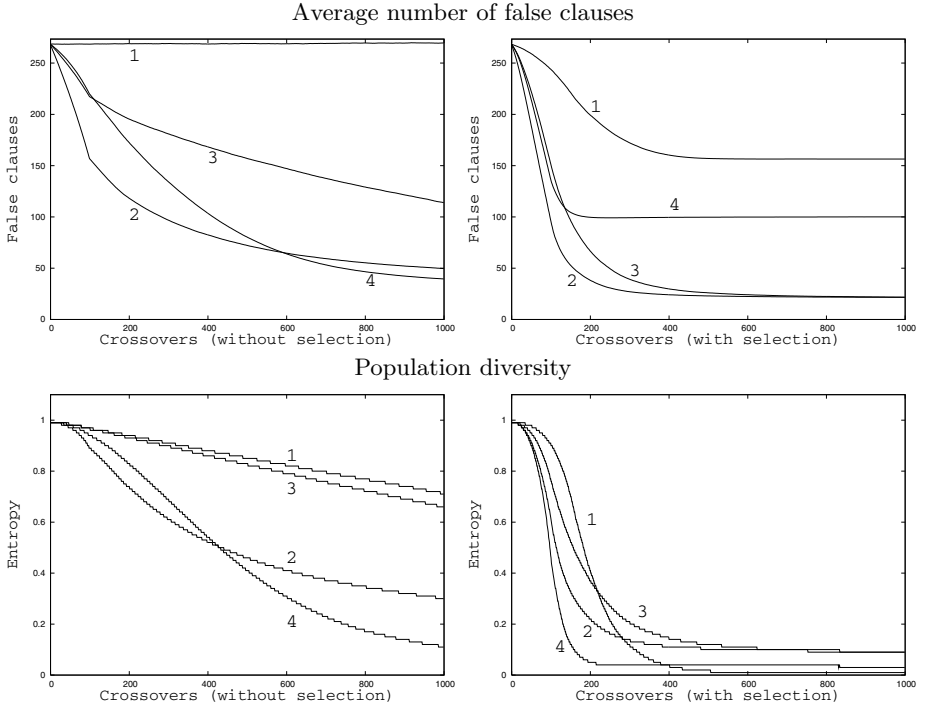


Fig. 2. The left column shows the crossovers without the selection stage and the right column shows the crossovers with the selection stage (1-Uniform crossover, 2-CCTM crossover, 3-CC crossover and 4-F&F crossover)

Figure 2 shows the performances of the four crossover operators combined with or without selection according to the two measures. One observes that:

- the selection damages the F&F crossover behavior,
- the selection improves the results of the uniform crossover, the CCTM crossover and the CC crossover,
- the uniform crossover does not provide good results with or without selection,
- the CCTM crossover and the CC crossover have a better entropy than the F&F crossover,
- the CC crossover and the CCTM crossover with the selection process provide very good results with a diversity higher than all the other crossovers.

Therefore, an efficient crossover is not necessary a crossover which quickly improves the whole population.Rather, the crossover should be able to ensure a good trade-off between quality and diversification of the population. The diversification allows a better exploration of the search space and prevents the population from stagnating in poor local optima.

4 Crossovers and Local Search

A good crossover insures a meaningful diversification leading the search to promising areas. Now it is necessary to add LS to perform an intensified search of solutions in the identified particular areas. Based on this idea, the GASAT algorithm is a hybridization of LS and a specific recombination operator [9].

To assess further the performance of each crossover operator within a whole algorithm, we insert each of this four cross operators in GASAT and employ a tabu search as local searcher. Now we run with the conditions detailed in Sect. 5.2 each of the four hybrid algorithms on four instances. A `3-blocks` is a blocks world instance, `par16-4-c` is a parity function instance and `f1000` and `color10-3` are presented in details in Sect. 5.2. Experimental conditions are the same for the four algorithms. Notice that the F&F crossover is used without the selection process since this gives better results (Fig. 2).

Two comparison criteria are used in order to evaluate the different crossovers improved by a LS. The first criterion is the success rate (%) which corresponds to the number of successful runs divided by the number of tries. The second criterion is the average number of crossovers (cr.) for the successful runs.

Table 1. Comparison of different crossovers included in the GASAT algorithm (if no assignment is found then the average number of false clauses is given)

	CC		CCTM		F&F		Uniform	
instances	%	cr.	%	cr.	%	cr.	%	cr.
3-blocks	10.00	439.00	*(1 cl.)*		*(1 cl.)*		*(1 cl.)*	
color-10-3	100.00	287.00	93.33	224.06	96.66	255.20	93.33	218.00
f1000	90.35	235.31	66.66	177.38	83.33	189.90	80.00	146.08
par16-4-c	5.00	788.00	10.00	206.00	*(2.80 cl.)*		*(2.75 cl.)*	

Table 1 shows a slight dominance of the CC crossover. These results show the interaction between CC and LS is more powerful than the interactions between the other crossovers and LS operator.

5 Experimental Results with the CC Crossover

In this section, we evaluate the GASAT algorithm with the CC crossover on different classes of benchmarks. We compare first GASAT with five evolutionary algorithms presented in [7]. Then we show comparisons with the well known Walksat [15] and UnitWalk [10], a winner of the SAT2003 competition. All our tests are realized on a cluster with Linux and Alinka (5 nodes with each of them 2 CPU Pentium IV 2, 2 Ghz and 1 GB of RAM) used in sequential run.

5.1 Comparisons with Evolutionary Algorithms from [7]

The purpose of this section is to compare GASAT with several state-of-the-art evolutionary algorithms presented in [7]. To make the comparison as fair as possible, we run GASAT under the same condition as that used in [7]. More precisely, each problem instance is solved between 5 and 50 times, each run being limited to 3×10^5. The benchmarks used in this experimentation are:

- suite A, 4 groups of 3 instances with 30, 50, 75 and 100 variables,
- suite B, 3 groups of 50 instances with 50, 75 and 100 variables,
- suite C, 5 groups of 100 instances with 20, 40, 60, 80 and 100 variables.

All these instances are random 3-sat instances with a ratio clauses-to-variables of 4.3.

Table 2. Comparison between evolutionary algorithms (%-success rate, fl.-average number of flips of successful runs)

Instances			GASAT		SAWEA		RFEA2		RFEA2+		FlipGA		ASAP	
Suite	nb.	var.	%	fl.	%	fl.	%	fl.	%	fl.	%	fl.	%	fl.
A	3	30	99	1123	100	34015	100	3535	100	2481	100	25490	100	9550
A	3	40	100	1135	93	53289	100	3231	100	3081	100	17693	100	8760
A	3	50	91	1850	85	60743	100	8506	100	7822	100	127900	100	68483
A	3	100	95	7550	72	86631	99	26501	97	34780	87	116653	100	52276
B	50	50	96	2732	-	-	100	12053	100	11350	100	103800	100	61186
B	50	75	83	6703	-	-	95	41478	96	39396	82	29818	87	39659
B	50	100	69	28433	-	-	77	71907	81	80282	57	20675	59	43601
C	100	20	100	109	100	12634	100	365	100	365	100	1073	100	648
C	100	40	100	903	89	35988	100	3015	100	2951	100	14320	100	16644
C	100	60	97	9597	73	47131	99	18857	99	19957	100	127520	100	184419
C	100	80	66	7153	52	62859	92	50199	95	49312	73	29957	72	45942
C	100	100	74	1533	51	69657	72	68053	79	74459	62	20319	61	34548

Results of GASAT are shown in Table 2, together with the results of the five evolutionary algorithms reported in [7]. One observes that on these random instances GASAT with CC crossover does not have the best success rate but requires in general much fewer flips to obtain a solution. Next section will show more results of GASAT on much larger random instances as well as structured instances.

5.2 Comparison with Walksat and UnitWalk

Due to the incomplete and non deterministic character of GASAT, Walksat and UnitWalk, each algorithm has been run 20 times on each benchmark with an execution time limited to 1 hour. These algorithms are used with their standard parameters. The number of flips is limited to 101×10^5 for the three algorithms. We provide now the precise parameters for each algorithm.

GASAT: GASAT works with a population of 10^2 individuals. During the generation of this population, a LS of 10^3 flips is applied to each individual. The selection process for the parents and the insertion condition for the child are activated. The number of possible parents for the crossover is limited to 15 different individuals. The number of allowed crossovers is 10^3 and a LS (tabu search) of at most 10^4 flips is applied on each child. The size of the tabu list is set to 10% of the number of variables in the problem.

Walksat[2]: We use the version v41. The number of tries is 10 with at most 101×10^4 flips for each try. Walksat uses the "novelty" heuristic with a noise set to 0.5.

UnitWalk[3]: We use UnitWalk version 0.981. The maximum number of flips is 101×10^5 with 1 run.

Benchmarks. Two classes of instances have been used: structured instances and random instances. Some instances are satisfiable and others are unsatisfiable. These instances[4] were used by the SAT2002 [17] or SAT2003 competition.

- structured instances:
  - `color-10-3`, `color-18-4`, `color-22-5` (chessboard coloring problem),
  - `difp_19_0_arr_rcr`, `difp_19_99_arr_rcr` (integer factorization),
  - `g125.17, g125.18` (graph coloring instances),
  - `mat25.shuffled`, `mat26.shuffled` (matrix multiplication),
  - `par32-5`, `par32-5-c` (problems of learning the parity function).
- random instances:
  - `f1000`, `f2000` (DIMACS instances),
  - 2 instances of 500 variables generated by `hgen2` with 2 different seeds,
  - 2 instances generated by `glassy` one with 399 and 450 variables.

[2] Walksat is available : `http://www.cs.washington.edu/homes/kautz/walksat/`

[3] UnitWalk is available : `http://logic.pdmi.ras.ru/ arist/UnitWalk/`

[4] Available at `http://www.info.univ-angers.fr/pub/lardeux/SAT/benchmarks-EN.html`

For each instance, the number of clauses is given in the column *cl* and the number of variables is given in the column *var*. The random seed is also given when it is known.

Evaluation. Three comparison criteria are used to evaluate GASAT and to compare it with UnitWalk and Walksat. The first criterion is the success rate (%) which corresponds to the number of success divided by the number of runs. This criterion is important since it highlights the search power of the algorithm. The two other criteria are the average number of flips (fl.) and the average time in seconds (sec.) for the successful runs. The number of flips for GASAT include the flips performed during the crossover operations.

Table 3. Structured instances (if no assignment is found then the average number of false clauses is given between parentheses, ? indicates the satisfiability of an instance is unknown)

Benchmarks				GASAT			Walksat			UnitWalk [5]		
instances	var	cls	sat	%	fl.[6]	sec.	%	fl.[6]	sec.	%	fl.[6]	sec.
color-10-3	300	6475	Y	100	1046	93.33	100	404	24.41	(*13.00 clauses*)		
color-18-4	1296	95905	?	(*25.95 clauses*)			(*38.00 clauses*)			(*65.00 clauses*)		
color-22-5	2420	272129	?	(*9.10 clauses*)			(*13.60 clauses*)			(*57.00 clauses*)		
difp_19_0_arr_rcr	1201	6563	Y	(*6.75 clauses*)			(*21.90 clauses*)			100	10100	319.53
difp_19_99_arr_rcr	1201	6563	Y	(*9.25 clauses*)			(*21.25 clauses*)			(*88.00 clauses*)		
g125.17	2125	66272	Y	(*3.00 clauses*)			35	458	251.46	(*28.00 clauses*)		
g125.18	2250	70163	Y	70	5501	860.81	100	9	2.40	(*23.00 clauses*)		
mat25.shuffled	588	1968	N	(*3.35 clauses*)			(*3.00 clauses*)			(*28.00 clauses*)		
mat26.shuffled	744	2464	N	(*3.20 clauses*)			(*2.55 clauses*)			(*25.00 clauses*)		
par32-5-c	1339	5350	Y	(*5.90 clauses*)			(*12.70 clauses*)			(*60.00 clauses*)		
par32-5	3176	10325	Y	(*4.15 clauses*)			(*12.10 clauses*)			10	10100	191.65

Table 4. Random instances (if no assignment is found then the average number of false clauses is given between parentheses, ? indicates the satisfiability of an instance is unknown)

Benchmarks					GASAT			Walksat			UnitWalk [5]		
gen	seed	var	cls	sat	%	fl.[6]	sec.	%	fl.[6]	sec.	%	fl.[6]	sec.
glassy	1069116088	399	1862	Y	(*5.00 clauses*)			(*5.60 clauses*)			(*43.00 clauses*)		
glassy	325799114	450	2100	Y	(*8.10 clauses*)			(*10.70 clauses*)			(*38.00 clauses*)		
f1000	-	1000	4250	Y	90	2485	35.75	100	367	9.17	100	1386	6.45
f2000	-	2000	8500	Y	5	3417	207.00	70	586	26.65	100	1232	10.41
hgen2	1205525430	500	1750	Y	(*1.00 clause*)			(*1.00 clause*)			(*8.00 clauses*)		
hgen2	512100147	500	1750	Y	(*1.00 clause*)			(*1.00 clause*)			(*21.00 clauses*)		

[5] When no assignment is found UnitWalk does not give the best number of false clauses but only the last found.

[6] You must multiply by 10^3 to obtain the real number of flips.

Results Analysis. The results given in Table 3 and Table 4 do not show a clear supremacy of one algorithm over other ones. However, GASAT is generally more efficient on structured instances than on random instances. These results are not surprising since the CC crossover is based on the idea of exploring clause structure. Taking the two tables together, one may conclude GASAT gives very competitive results on the tested instances.

6 Conclusion

We have presented two new recombination operators for SAT and compared them with two previous known operators. The most powerful crossover is the Corrective Clause crossover. Inserted in the GASAT algorithm, it gives very interesting results. By their global and local nature, the crossover and the LS operators act interactively to ensure a good compromise between exploration and exploitation of the search space. Moreover, a selection mechanism is also employed to preserve the diversity of the population.

GASAT with the CC crossover is evaluated on both structured and random instances. Its performances are compared with two well-known algorithms (Walksat and UnitWalk) and several state-of-the-art evolutionary algorithms. The experimentations show that GASAT gives globally very competitive results, in particular, on structured instances. Meanwhile, it seems that GASAT behaves less well on some random instances.

The new crossovers reported in this paper are a first step. Studies are on the way to have a better interactivity between crossovers and LS. We are also working on other hybridizations such as CC and complete algorithms.

Acknowledgments. We would like to thank the anonymous referees for their helpful comments and remarks.

References

1. Belaid Benhamou and Lakhdar Sais. Theoretical study of symmetries in propositional calculus and applications. In *CADE'92*, pages 281–294, 1992.
2. Martin Davis, George Logemann, and Donald Loveland. A machine program for theorem-proving. *Communications of the ACM*, 5(7):394–397, Jul 1962.
3. Kenneth A. De Jong and William M. Spears. Using genetic algorithm to solve NP-complete problems. In *Proc. of the Third Int. Conf. on Genetic Aalgorithms*, pages 124–132, San Mateo, CA, 1989.
4. Olivier Dubois and Gilles Dequen. A backbone-search heuristic for efficient solving of hard 3-SAT formulae. In Bernhard Nebel, editor, *Proc. of the IJCAI'01*, pages 248–253, San Francisco, CA, 2001.
5. Charles Fleurent and Jacques A. Ferland. Object-oriented implementation of heuristic search methods for graph coloring, maximum clique, and satisfiability. In *Cliques, Coloring, and Satisfiability: Second DIMACS Implementation Challenge*, volume 26 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 619–652, 1994.

6. Michael R. Garey and David S. Johnson. *Computers and Intractability , A Guide to the Theory of NP-Completeness*. W.H. Freeman & Company, San Francisco, 1978.
7. Jens Gottlieb, Elena Marchiori, and Claudio Rossi. Evolutionary algorithms for the satisfiability problem. *Evolutionary Computation*, 10(1):35–50, 2002.
8. Pierre Hansen and Brigitte Jaumard. Algorithms for the maximum satisfiability problem. *Computing*, 44(4):279–303, 1990.
9. Jin-Kao Hao, Frédéric Lardeux, and Frédéric Saubion. Evolutionary computing for the satisfiability problem. In *Applications of Evolutionary Computing*, volume 2611 of *LNCS*, pages 259–268, 2003.
10. Edward A. Hirsch and Arist Kojevnikov. UnitWalk: A new SAT solver that uses local search guided by unit clause elimination. PDMI preprint 9/2001, Steklov Institute of Mathematics at St.Petersburg, 2001.
11. Brigitte Jaumard, Mihnea Stan, and Jacques Desrosiers. Tabu search and a quadratic relaxation for the satisfiability problem. In *Cliques, Coloring, and Satisfiability: Second DIMACS Implementation Challenge*, volume 26 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 457–478, 1994.
12. Chu Min Li. Integrating equivalency reasoning into davis-putnam procedure. In *Proc. of the AAAI'00*, pages 291–296, 2000.
13. Chu Min Li and Anbulagan. Heuristics based on unit propagation for satisfiability problems. In *Proc. of the IJCAI'97*, pages 366–371, 1997.
14. Elena Marchiori and Claudio Rossi. A flipping genetic algorithm for hard 3-SAT problems. In *Proceedings of the Genetic and Evolutionary Computation Conference*, volume 1, pages 393–400, 1999.
15. Bart Selman, Henry A. Kautz, and Bram Cohen. Noise strategies for improving local search. In *Proc. of the AAAI, Vol. 1*, pages 337–343, 1994.
16. Bart Selman, Hector J. Levesque, and David G. Mitchell. A new method for solving hard satisfiability problems. In *Proc. of the AAAI'92*, pages 440–446, San Jose, CA, 1992.
17. Laurent Simon, Daniel Le Berre, and Edward A. Hirsch. The SAT2002 competition. *Technical report, Fifth International Symposium on the Theory and Applications of Satisfiability Testing, May 2002.*
18. Mutsunori Yagiura and Toshihide Ibarak. Efficient 2 and 3-flip neighborhood search algorithms for the MAX SAT: Experimental evaluation. *Journal of Heuristics*, 7(5):423–442, 2001.
19. Hantao Zhang. SATO: An efficient propositional prover. In *Proc. of the 14th International Conference on Automated deduction*, volume 1249 of *LNAI*, pages 272–275, Berlin, 1997.

Recombination and Self-Adaptation in Multi-objective Genetic Algorithms

Bruno Sareni, Jérémi Regnier, and Xavier Roboam

Laboratoire d'Electrotechnique et d'Electronique Industrielle
Unité mixte de Recherche INPT-ENSEEIHT / CNRS
BP 7122 – 2 rue Camichel – 31071 Toulouse Cedex 7 – France
{sareni, regnier, roboam}@leei.enseeiht.fr,
http://www.leei.enseeiht.fr

Abstract. This paper investigates the influence of recombination and self-adaptation in real-encoded Multi-Objective Genetic Algorithms (MOGAs). NSGA-II and SPEA2 are used as example to characterize the efficiency of MOGAs in relation to various recombination operators. The blend crossover, the simulated binary crossover and the breeder genetic crossover are compared for both MOGAs on multi-objective problems of the literature. Finally, a self-adaptive recombination scheme is proposed to improve the robustness of MOGAs.

1 Introduction

In recent years, extensive research has been done in the field of Multi-Objective Genetic Algorithms (MOGAs) [1], [2], [3], [4]. However, most of works have been focused on selection, elitism and niching operators. Although the need of studying the influence of recombination and self-adaptation in MOGAs has been underlined [5], only a few contributions on these issues have been carried out [2], [6]. In this paper, we investigate the efficiency of three crossover operators for real-encoded MOGAs and propose a self-adaptive recombination scheme which improves their robustness. The second versions of the Non-dominated Sorting Genetic Algorithm (NSGA-II) [3] and of the Strength Pareto Genetic Algorithm (SPEA2) [4] are used as example to characterize the efficiency of the studied recombination operators on test problems of the literature.

2 Elitist Multi-objective Genetic Algorithms

Since the mid-1990s, there has been a growing interest in solving multi-objective problems by Genetic Algorithms. In particular, elitist MOGAs based on Pareto approaches have become more and more popular because of their capabilities to approximate the set of optimal trade-offs in a single run [1], [3], [4]. Elitist MOGAs use

P. Liardet et al. (Eds.): EA 2003, LNCS 2936, pp. 115–126, 2004.

an external population, namely *archive*, which preserves non-dominated individuals in the population. At each generation, individuals (*parents*) selected from the archive (and/or from the population) following Pareto domination rules are crossed and mutated to create new individuals (*children*). The population of children and the archive are merged to assess the non-dominated set of the next generation. If the number of non-dominated individuals is higher than the size of the archive, a *clustering* method is used to preserve most representative solutions and eliminate others in order to keep a constant archive size. Note that niching is used in the selection scheme when individuals involved in a tournament have the same Pareto domination rank.

The second version of the Non-dominated Sorting Genetic Algorithm (NSGA-II) is based on the principles previously exposed. NSGA-II determines all successive fronts in the population (the best front corresponding to the non-dominated set). Moreover, a *crowding distance* is used to estimate the density of solutions surrounding each individual on a given front. In a tournament, if individuals belong to the same front, the selected one is that with the greater crowding distance. This niching index is also used in the clustering operator to uniformly distribute the individuals on the Pareto front. All details of the algorithm can be found in [3].

The new version of the Strength Pareto Genetic Algorithm (SPEA2) is rather similar to the NSGA-II. It essentially differs in the clustering method used to merge non-dominated individuals in the archive and on the selection method based on a fitness assignment (called strength) related to Pareto ranking of individuals in the population [4].

In this, study NSGA-II and SPEA2 are taken as reference to investigate the influence of recombination and self-adaptation in real-encoded MOGAs.

3 Recombination and Self-Adaptive Procedures

We examine various recombination and self-adaptive procedures for real-encoded MOGAs :

3.1 The Blend Crossover

From two parent solutions $p_1(i)$ and $p_2(i)$, the blend crossover (BLX-α) creates one child $c(i)$ as follows [7] :

$$c(i) = p_1(i) + \beta(p_2(i) - p_1(i)) \tag{1}$$

where β is a random variable in the interval $[-\alpha, 1+\alpha]$, i denoting the index related to the object variable of the child and parents solutions. If α is set to zero, this crossover creates a random solution inside the range defined by the parents similarly to the arithmetical crossover [8]. Eshelman and Schaffer have reported that BLX-0.5 (with $\alpha = 0.5$) performs better than BLX with any other α value in a number of test problems.

3.2 The Simulated Binary Crossover

The simulated binary crossover (SBX) simulates the working principle of the single point crossover operator on binary strings [9]. From two parent solutions $p_1(i)$ and $p_2(i)$, it creates two children $c_1(i)$ and $c_2(i)$ as follows :

$$\begin{cases} c_1(i) = 0.5[(1+\beta)p_1(i) + (1-\beta)p_2(i)] \\ c_2(i) = 0.5[(1-\beta)p_1(i) + (1+\beta)p_2(i)] \end{cases} \tag{2}$$

with a spread factor β defined by (3),

$$\beta = \begin{cases} (2u)^{\frac{1}{\eta+1}} & \text{if } u < 0.5 \\ \left(\frac{1}{2(1-u)}\right)^{\frac{1}{\eta+1}} & \text{otherwise} \end{cases} \tag{3}$$

where u is a random variable in the interval [0,1] and η is a nonnegative real number that characterizes the distribution of the children in relation to their parents. A large value of η gives a higher probability for creating children near parents. Acting alone and without any mutation operator, SBX presents interesting properties of self-adaptation similarly to Evolution Strategies [9].

3.3 The Breeder Genetic Crossover

From two parent solutions $p_1(i)$ and $p_2(i)$, the BGX crossover creates one child $c(i)$ as follows [10]:

$$c(i) = p_1(i) \pm \frac{(p_2(i) - p_1(i))}{\|p_2(i) - p_1(i)\|} \Delta_i \delta \tag{4}$$

where Δ_i is normally set to 0.5 times the domain of definition of the object variable i and the metric denotes the Euclidean distance in the object variable space. δ is computed from a distribution that favors small values:

$$\delta = 2^{-ku} \tag{5}$$

where u is a random variable in the interval [0,1], the precision constant k being typically set to 16. Note that in [10], the child was placed more often in the direction to the best parent ($p_1(i)$ being the parent with the better fitness) and the minus sign in (4) was chosen with probability 0.9. In this work, the choice of $p_1(i)$ and the sign in (4) are made with a probability 0.5.

In Fig. 1, the probability density function per child for the previous investigated crossover operators is depicted. The corresponding parents p_1 and p_2 are marked with a full circle. Note these three crossover operators are rather complementary since BGX with k=16 essentially reinforces the accuracy in the neighborhood of the parents whereas SBX-0.5 favors global exploration.

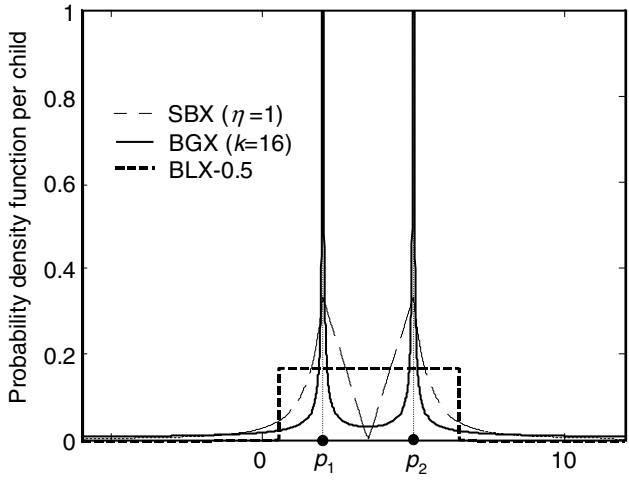


Fig. 1. Probability distribution of children solutions related to each crossover operators

3.4 Self-Adaptive Mutation

We also investigate a self-adaptive mutation operator used in standard Evolution Strategies [11], [9]. Each individual is characterized by its object variables $x(i)$ and associated standard deviation $\sigma(i)$. Children $(c(i),\sigma_c(i))$ are created from their parents $(p(i),\sigma_p(i))$ using the following rule :

$$\sigma_c(i) = \sigma_p(i)\exp(\tau' N(0,1) + \tau N_i(0,1)) \tag{6}$$

$$c(i) = p(i) + \sigma_c(i)\, N_i(0,1) \tag{7}$$

where $N(0,1)$ denotes a normally distributed random number with mean 0 and standard deviation 1. $N_i(0,1)$ indicates that the random number is generated anew for each value of i. The factors τ and $\tau'$are commonly set to $(2m^{1/2})^{-1/2}$ and $(2m)^{-1/2}$ where m denotes the number of object variables [11].

3.5 Self-Adaptive Recombination

As it is not possible to *a priori* know which crossover operator will be the most efficient on a specific problem, we propose a self-adaptive scheme similar to that of Spears for binary encoded GAs [12]. It consists in associating in the chromosome of individuals an additional gene (X-*gene*) that codes the type of crossover to apply during the recombination. When recombining two parents, the operating crossover is randomly chosen from the X-*gene* of the parents. Using this procedure, the MOGA will favor the crossover that produces the best children through the selection operator. To avoid premature convergence to a particular type of crossover, the X-*gene* also undergoes mutation.

4 Experimental Tests

4.1 Test Problems

We consider three multi-objective problems of the literature [13], [14] displayed in Table 1. ZDT4 is a multimodal continuous problem, which contains 21^9 local Pareto fronts. The global Pareto front is obtained with g=1 and is convex. ZTD6 has a non-uniformly distributed search space with solutions non-uniformly distributed along the Pareto front (the front is biased for solutions for which $f_1(x_1)$ is close to one). The Pareto front is obtained with g=1 and is non-convex. SCH is a generalization of the Schaffer's problem. A large variable space domain and a convex Pareto front characterize it.

Table 1. Test problems used in this study (minimization of both objectives)

Problem	Characteristics
ZDT4	$f_1(x) = x_1 \qquad 0 \le x_1 \le 1$ $f_2(x) = g\left(1 - \sqrt{\frac{x_1}{g}}\right) \quad -5 \le x_i \le 5 \quad i = 2,...,10$ where $g = 91 + \sum_{i=2}^{10}(x_i^2 - 10\cos(4\pi x_i))$
ZDT6	$f_1(x) = 1 - \exp(-4x_1)\sin^6(6\pi x_1)$ $f_2(x) = g(1-(f_1/g)^2) \qquad 0 \le x_i \le 1 \qquad i = 1,...,10$ where $g = 1 + 9\left(\sum_{i=2}^{10} x_i / 9\right)^{0.25}$
SCH	$f_1(x) = \frac{1}{40}\sum_{i=1}^{10} x_i^2 \quad -1000 \le x_i \le 1000 \quad i = 1,...,10$ $f_2(x) = \frac{1}{40}\sum_{i=1}^{10}(x_i - 2)^2$

4.2 Performance Criteria

Three performance criteria are used to assess the efficiency of MOGAs :

Average deviation to the theoretical Pareto-optimal front. The average distance $\bar{\varepsilon}$ of the non-dominated set to the Pareto-optimal front is computed as follows [13],

$$\bar{\varepsilon} = \frac{1}{|F|}\sum_{a \in F} \min\left\{\| a - a^* \| \quad a^* \in F^*\right\} \tag{8}$$

where F (respectively F^*) denotes the non-dominated set in the final population (respectively the theoretical Pareto-optimal front), a and a^* belonging to each subset. The metric in (8) is the Euclidean distance computed in the objective space.

Spread. We define the spread $\bar{\varepsilon}_{\min}$ as the average minimum distance of the non-dominated set to the Pareto-optimal solutions that minimizes each objective independently.

$$\bar{\varepsilon}_{\min} = \frac{1}{n}\sum_{i=1}^{n} \min\left\{ \left\| a - a^*_{i\min} \right\| \;\; a \in F \right\} \tag{9}$$

where n is the number of objectives and $a^*_{i\min}$ represents the theoretical solution of the Pareto-optimal front that minimizes the i-th objective. Spread characterizes the ability of MOGAs to detect boundary solutions of the Pareto-optimal front.

Spacing. Spacing Δ is a measure based on consecutive distances among the solutions of the non-dominated set [3]. It assesses the ability of the MOGAs to distribute its population uniformly along the Pareto-optimal front.

$$\Delta = \frac{1}{|F|-1}\sum_{i=1}^{|F|-1} \left| d_i - \bar{d} \right| \tag{10}$$

where d_i is the Euclidean distance between two consecutive solutions of the non-dominated set, $\bar{d}$ being the average of these distances. A value of zero for this metric indicates all the non-dominated solutions found are equidistantly spaced. Unlike the definition of Δ in [3], we do not include in the non-dominated set the boundary solutions of the theoretical Pareto-optimal front to take into account the spread (spread is independently evaluated by (9)).

Note that these criteria are complementary to assess the efficiency of MOGAs as indicated by Fig. 2 which illustrates various situations of the non-dominated set in relation to the theoretical Pareto-optimal front.

5 Tests Results

We successively compare the efficiency of NSGA-II and SPEA2 on the previous test problems using each crossover operators and self-adaptive procedures of sect. 3. All tests are made with the same number of objective function evaluations. NSGA-II and SPEA2 are run for 200 generations with a population size of 100. The archive size is also set to 100 and the crossover probability is 1. Both MOGAs use the BGA mutation operator [10] with a mutation rate of 1/m (where m is the number of variables). The self-adaptive recombination scheme employs the BGX with k=16, the SBX with η =1 and the BLX-0.5. The X-*gene* undergoes mutation with a probability of 5%. NSGA-II and SPEA2 with self-adaptive mutations operate with initial standard deviations set to 1/10 times the domain of definition of the object variables and without

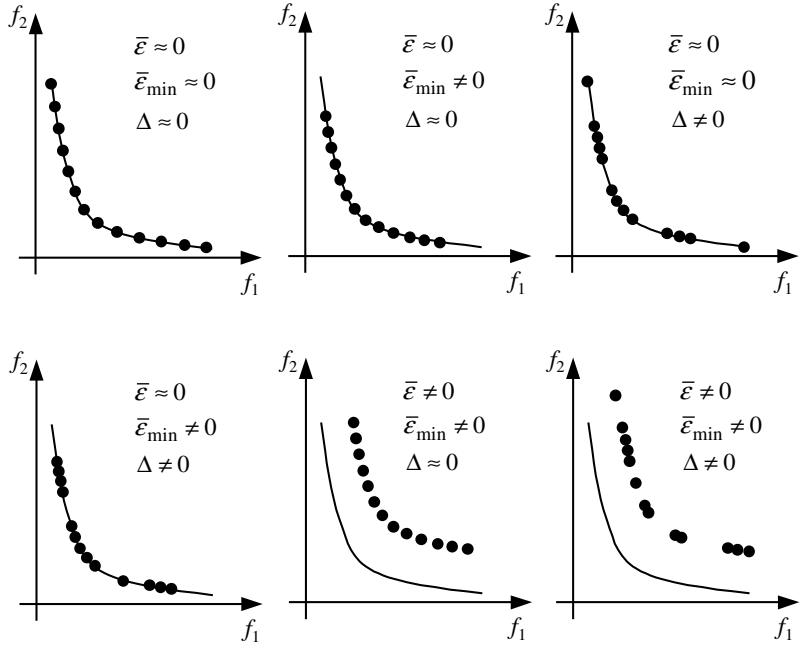


Fig. 2. Illustration of the performance criteria used in this study on various situations. The non-dominated set is symbolized with full circles and the theoretical Pareto optimal front is represented by a continuous line. $\bar{\varepsilon} \approx 0$ (resp. $\bar{\varepsilon} \neq 0$) indicates a good (resp. a bad) average deviation, $\bar{\varepsilon}_{\min} \approx 0$ (resp. $\bar{\varepsilon}_{\min} \neq 0$) a good (resp. a bad) spread and $\Delta \approx 0$ (resp. $\Delta \neq 0$) a good (resp. a bad) spacing

any crossover operator. For all investigated tests, 100 runs are made with random populations to take into account the stochastic nature of the MOGA. An average statistic is taken from the final population for the performance criteria.

5.1 Influence of the Recombination Operators

We present in Tables 2–4 the values of the performance criteria on the investigated problems for the NSGA-II and the SPEA2 in relation to each recombination and self-adaptive procedures. From these results, we propose a sort of the recombination and self-adaptive operators on these test problems for each MOGA. Note that this sort was established by giving priority to the average deviation $\bar{\varepsilon}$.

As also underlined in a earlier study [2], it can been seen that MOGAs without crossover operator perform poorly even if they use self-adaptive mutations. Table 2 shows that NSGA-II and SPEA2 with BLX-0.5 give the best results on ZDT4. BLX-0.5 associated with the SPEA2 works well on ZDT6 but performs poorly when it is coupled with the NSGA-II. This behavior will be also noted for the self-adaptive recombination scheme in the following section. NSGA-II with the BGX crossover

works well on ZDT6 but performs extremely poorly on SCH (convergence was not achieved after 200 generations; only one non-dominated individual was found in the final population in all runs). Except with the SBX crossover, NSGA-II and SPEA2 fail to spread correctly their population on SCH. Therefore, SBX is ranked at the top of the sort for this problem despite a slightly lowest quality for $\bar{\varepsilon}$ and Δ.

Finally, it can be seen that *best results are always obtained by simple crossover acting alone*. However, the efficiency of each crossover operator clearly depends on the characteristics of the test problem. Therefore, using simultaneously multiple crossover operators through a self-adaptive recombination scheme tends to improve the robustness of the MOGAs. We verify this property since the self-adaptive scheme performs extremely well in all cases (the ranking efficiency always equals 2).

Table 2. Performance criteria on problem ZDT4. Results are averaged on 100 runs with random initial populations. Best results are indicated in bold types and margin errors with 95% confidence are given in brackets

MOGA	Operator	Av. dev. $\bar{\varepsilon}$	Spread $\bar{\varepsilon}_{\min}$	Spacing Δ	Ranking
NSGA-II	BGX (k=16)	2.231 [0.185]	1.769 [0.148]	0.112 [0.011]	4 (poor)
	SBX (η=1)	1.961 [0.170]	1.656 [0.138]	0.033 [0.009]	3 (good)
	BLX-0.5	**1.483** [0.151]	**1.275** [0.122]	0.019 [0.006]	**1** (excellent)
	Self-Ad. Mut.	10.41 [1.000]	9.439 [0.939]	**0.017** [0.001]	5 (very poor)
	Self-Ad. Rec.	1.688 [0.164]	1.437 [0.136]	0.021 [0.008]	2 (very good)
SPEA2	BGX (k=16)	5.343 [0.436]	4.367 [0.380]	0.301 [0.041]	4 (poor)
	SBX (η=1)	2.233 [0.221]	1.894 [0.163]	0.028 [0.008]	3 (good)
	BLX-0.5	**1.527** [0.148]	**1.373** [0.122]	0.021 [0.005]	**1** (excellent)
	Self-Ad. Mut.	10.55 [1.090]	9.525 [1.015]	**0.012** [0.002]	5 (very poor)
	Self-Ad. Rec.	1.616 [0.161]	1.427 [0.133]	0.024 [0.006]	2 (very good)

Table 3. Performance criteria on problem ZDT6. Results are averaged on 100 runs with random initial populations. Best results are indicated in bold types and margin errors with 95% confidence are given in brackets

MOGA	Operator	Av. dev. $\bar{\varepsilon}$	Spread $\bar{\varepsilon}_{\min}$	Spacing Δ	Ranking
NSGA-II	**BGX (k=16)**	**0.000** [0.000]	**0.000** [0.000]	**0.006** [0.000]	**1** (excellent)
	SBX (η=1)	0.185 [0.018]	0.001 [0.000]	0.202 [0.022]	4 (poor)
	BLX-0.5	0.081 [0.044]	**0.000** [0000]	0.096 [0.054]	3 (good)
	Self-Ad. Mut.	1.866 [0.114]	0.442 [0.077]	0.543 [0.063]	5 (very poor)
	Self-Ad. Rec.	0.014 [0.006]	**0.000** [0.000]	0.024 [0.010]	2 (very good)
SPEA2	BGX (k=16)	0.068 [0.008]	**0.000** [0.000]	0.071 [0.006]	3 (good)
	SBX (η=1)	0.186 [0.016]	**0.000** [0.000]	0.141 [0.015]	4 (poor)
	BLX-0.5	**0.059** [0.005]	**0.000** [0.000]	**0.055** [0.005]	**1** (very good)
	Self-Ad. Mut.	1.662 [0.126]	0.554 [0.090]	0.484 [0.049]	5 (very poor)
	Self-Ad. Rec.	0.061 [0.006]	**0.000** [0.000]	0.056 [0.007]	2 (very good)

Table 4. Performance criteria on problem SCH. Results are averaged on 100 runs with random initial populations. Best results are indicated in bold types and margin errors with 95% confidence are given in brackets

MOGA	Operator	Av. dev. $\overline{\varepsilon}$	Spread $\overline{\varepsilon}_{\min}$	Spacing Δ	Ranking
NSGA-II	BGX (k=16)	no convergence achieved in 200 generations			5 (very poor)
	SBX (η=1)	0.007 [0.000]	**0.086** [0.006]	0.006 [0.000]	**1** (excellent)
	BLX-0.5	**0.004** [0.000]	0.209 [0.008]	**0.004** [0.000]	3 (good)
	Self-Ad. Mut.	0.091 [0.006]	0.195 [0.059]	0.008 [0.001]	4 (poor)
	Self-Ad. Rec.	**0.004** [0.000]	0.118 [0.009]	0.005 [0.000]	2 (very good)
SPEA2	BGX (k=16)	no convergence achieved in 200 generations			5 (very poor)
	SBX (η=1)	**0.006** [0.000]	**0.100** [0.007]	0.005 [0.000]	**1** (very good)
	BLX-0.5	0.009 [0.001]	0.289 [0.010]	**0.002** [0.000]	3 (good)
	Self-Ad. Mut.	0.302 [0.306]	0.445 [0.299]	0.008 [0.002]	4 (poor)
	Self-Ad. Rec.	**0.006** [0.000]	0.129 [0.009]	0.004 [0.000]	2 (very good)

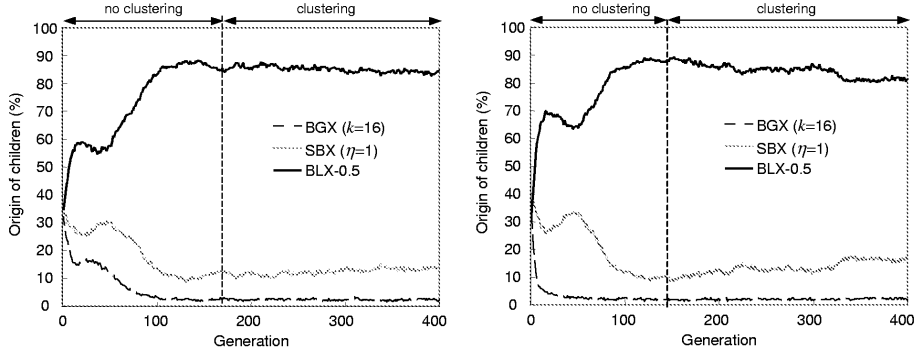
5.2 Analysis of the Self-Adaptive Recombination Scheme

To understand the mechanism of the self-adaptive recombination scheme, we plot in Fig. 3-5 the origin of children for each investigated test problem as a function of the generation number. Results are extended to 400 generations and averaged on 100 runs. We also indicate the threshold from which the clustering is operating.

It can be seen that the self-adaptive recombination scheme is able to direct the MOGA towards the crossover operator which performs the best on a given test problem. Therefore, the BLX-0.5 which has the best global exploration properties, is rapidly favored on ZDT4 to avoid misleading local Pareto-optimal fronts (see Fig. 3). Because of the large search space, this crossover is also preferred on SCH at the beginning of the search but both MOGAs finally switch towards the SBX crossover to better distribute individuals on boundary points (see Fig. 5 and Table 3). Curiously, the behavior of the self-adaptive recombination scheme on ZDT6 is radically different for the SPEA2 and the NSGA-II. These results are in accordance with those of sect. 5.1 (see Table 3) which show that SPEA2 performs better with SBX and NSGA-II better with BGX. However, both MOGA need to use the BGX crossover to avoid misleading attractors of high f_2 values in the neighbourhood of the boundary point of the Pareto front defined by $f_1^* \approx 0.28$.

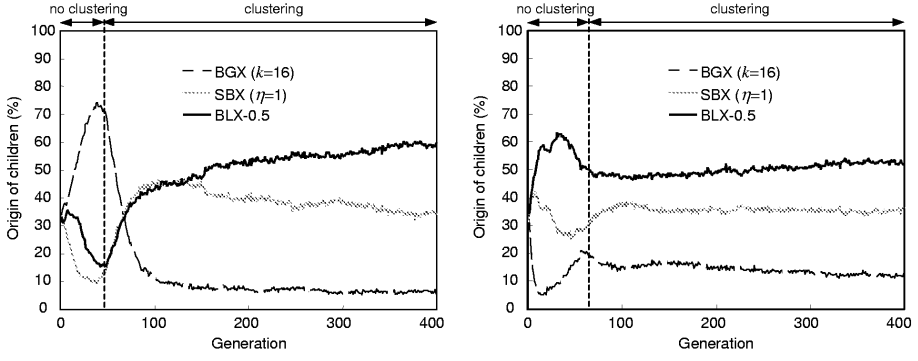
Note that the steady state behavior of the archive characterized by clustering operations does not necessary correspond to the steady state operations of the self-adaptive recombination scheme. Recombination rates of each crossover operator can evolve continuously during the search as shown in Fig 3-5.

Finally, we examine in Table 5-7 the influence of the X-*gene* mutation rate on the efficiency of MOGAs with the studied self-adaptive recombination scheme. Results show that performance criteria are not very sensitive to this factor. However, low mutation rates have to be preferred to exploit benefit of self-adaptation through the selection procedure.



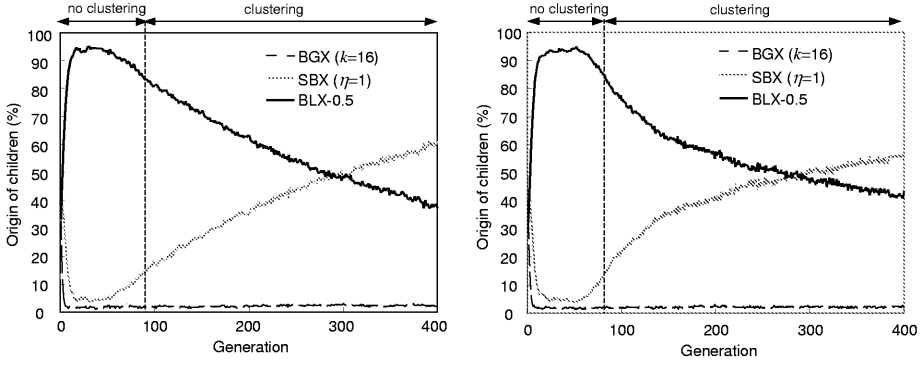
(a) NSGA-II (b) SPEA2

Fig. 3. Origin of children in the self-adaptive recombination scheme. Results on ZDT4 (average on 100 runs)



(a) NSGA-II (b) SPEA2

Fig. 4. Origin of children in the self-adaptive recombination scheme. Results on ZDT6 (average on 100 runs)



(a) NSGA-II (b) SPEA2

Fig. 5. Origin of children in the self-adaptive recombination scheme. Results on SCH (average on 100 runs)

Table 5. Influence of the X-*gene* mutation rate on ZDT4 (average on 100 runs). Best results are indicated in bold types and margin errors with 95% confidence are given in brackets

MOGA	X-*gene* mutation rate	Av. dev. $\overline{\varepsilon}$	Spread $\overline{\varepsilon}_{\min}$	Spacing Δ
NSGA-II	0	1.806 [0.198]	1.516 [0.158]	0.027 [0.012]
	5%	**1.688** [0.164]	**1.437** [0.136]	0.021 [0.008]
	20%	1.699 [0.182]	1.460 [0.148]	**0.019** [0.005]
	100% (random case)	2.084 [0.187]	1.783 [0.151]	0.035 [0.010]
SPEA2	0	1.624 [0.138]	1.446 [0.118]	**0.018** [0.004]
	5%	**1.616** [0.161]	**1.427** [0.133]	0.024 [0.006]
	20%	1.733 [0.182]	1.521 [0.135]	0.020 [0.006]
	100% (random case)	2.192 [0.201]	1.838 [0.147]	0.037 [0.009]

Table 6. Influence of the X-*gene* mutation rate on ZDT6 (average on 100 runs). Best results are indicated in bold types and margin errors with 95% confidence are given in brackets

MOGA	X-*gene* mutation rate	Av. dev. $\overline{\varepsilon}$	Spread $\overline{\varepsilon}_{\min}$	Spacing Δ
NSGA-II	0	**0.005** [0.005]	**0.000** [0.000]	**0.011** [0.005]
	5%	0.014 [0.006]	**0.000** [0.000]	0.024 [0.010]
	20%	0.015 [0.006]	**0.000** [0.000]	0.026 [0.010]
	100% (random case)	0.012 [0.004]	**0.000** [0.000]	0.023 [0.008]
SPEA2	0	0.076 [0.008]	**0.000** [0.000]	0.064 [0.007]
	5%	**0.061** [0.006]	**0.000** [0.000]	**0.056** [0.007]
	20%	0.071 [0.008]	**0.000** [0.000]	0.064 [0.008]
	100% (random case)	0.080 [0.007]	**0.000** [0.000]	0.069 [0.007]

Table 7. Influence of the X-*gene* mutation rate on SCH (average on 100 runs). Best results are indicated in bold types and margin errors with 95% confidence are given in brackets

MOGA	X-*gene* mutation rate	Av. dev. $\overline{\varepsilon}$	Spread $\overline{\varepsilon}_{\min}$	Spacing Δ
NSGA-II	0	0.004 [0.000]	0.210 [0.009]	**0.004** [0.000]
	5%	**0.004** [0.000]	0.118 [0.009]	0.005 [0.000]
	20%	0.004 [0.000]	**0.103** [0.009]	0.005 [0.000]
	100% (random case)	0.006 [0.000]	0.124 [0.009]	0.005 [0.000]
SPEA2	0	**0.003** [0.000]	0.289 [0.009]	**0.002** [0.000]
	5%	0.006 [0.000]	**0.129** [0.009]	0.004 [0.000]
	20%	**0.003** [0.000]	0.132 [0.009]	0.004 [0.000]
	100% (random case)	0.006 [0.000]	0.169 [0.009]	0.004 [0.000]

6 Conclusion

In this paper, we have shown that the efficiency of real-encoded MOGAs strongly depends on the crossover operators used to explore new solutions of the search space. Therefore, we have proposed a self-adaptive recombination scheme based on three complementary crossover operators to reduce the sensitivity to the crossover proce-

dure and improve MOGA robustness. First results are promising and further investigations on difficult constrained problems should confirm the advantage of using this technique in MOGAs.

References

1. Corne D.W, Knowles J.D., Oates M.J. : The Pareto Envelope-based Selection Algorithm for Multiobjective Optimization. In M. Schoenauer, K. Deb, G. Rudolph, X. Yao, E. Lutton, J.J. Merelo and H.P. Schwefel eds., Proceedings of the Parallel Problem Solving from Nature VI Conference, Springer (2000) 839–848
2. Costa L., Oliveira P. : An Evolution Strategy for Multiobjective Optimization. Congress on Evolutionary Computation (CEC'2002) IEEE Service Center, Piscataway, New Jersey, Vol. 1 (2002) pp. 97–102
3. Deb K, Agrawal S., Pratab A., Meyarivan T.: A fast-elitist non-dominated sorting genetic algorithm for multiobjective optimization: NSGA-II. Proceeding of the Parallel Problem Solving from Nature VI Conference (2000) 849–858
4. Zitzler E., Laumanns M., Thiele L. : SPEA2: Improving the Strength Pareto Evolutionary Algorithm. In K. Giannakoglou, D. Tsahalis, J. Periaux, P. Papailou and T. Fogarty (eds.) EUROGEN 2001, Evolutionary Methods for Design, Optimization and Control with Applications to Industrial Problems, Athens, Greece, September (2001) 12–21
5. Laumanns M., Rudolph G., Schwefel H.P. : Approximating the Pareto Set: Concepts, Diversity Issues, and Performance Assessment, Technical Report CI-72/99, Dortmund: Department of Computer Science/LS11, University of Dortmund, Germany, ISSN 1433-3325 (1999)
6. Laumanns M., Rudolph G., Schwefel H.P. : Mutation Control and Convergence in Evolutionary Multi-Objective Optimization. In R. Matousek and P. Osmera (eds.), Proceedings of the 7th International Mendel Conference on Soft Computing (MENDEL 2001), Brno University of Technology, Brno, Czech Republic, (2001) 97–106
7. Eshelman L.J., Schaffer J.D. : Real-coded genetic algorithms and interval schemata. In D. Whitley (Ed). Foundations of Genetic Algorithms II (1993) 187–202
8. Michalewicz, Z.: Genetic Algorithms + Data Structures = Evolution Programs. 3rd edn. Springer-Verlag, Berlin Heidelberg New York (1996)
9. Deb K, Beyer H.G : Self-Adaptation in Real-Parameter Genetic Algorithms with Simulated Binary Crossover. Genetic and Evolutionary Computation Conference (GECCO-99), Orlando, FL. (1999)
10. Schlierkamp-Voosen D., Mühlenbein H. : Strategy Adaptation by Competing Subpopulations. Parallel Problem Solving from Nature 3 – PPSN III, Jerusalem, Springer (1994) 199–208
11. Bäck Th.: Evolutionary algorithms in Theory and Practice. Oxford University Press, New York (1996)
12. Spears W.M. : Adapting crossover in evolutionary algorithms. Proceeding of the 5th Annual Conference on Evolutionary Programming, San Diego, CA, Morgan Kaufmann Publishers (1995)
13. Zitzler E., Deb K., Thiele L. : Comparison of multiobjective evolutionary algorithms : Empirical results. Evolutionary Computation 8 (2) (2000) 173–195
14. Schaffer J.D. : Multiple objective optimization with vector evaluated genetic algorithms. In J.J. Grefenstette ed. Proccedings of the First International Conference on Genetic Algorithms and Their Applications, Pittsburgh, PA (1985) 93–100

Automatic Optical Fiber Alignment System Using Genetic Algorithms

Masahiro Murakawa, Hirokazu Nosato, and Tetsuya Higuchi

National Institute of Advanced Industrial Science and Technology (AIST),
Tsukuba Central 2, 1–1–1 Umezono, Tsukuba, Ibaraki, Japan
{m.murakawa,h.nosato,t-higuchi}@aist.go.jp

Abstract. We propose and demonstrate an automatic optical fiber alignment system using genetic algorithms. Connecting optical fibers is difficult because the connecting edges should be aligned with sub-micron-meter resolution. It, therefore, takes long time even for a human expert. Although automatic fiber alignment systems are being developed, they cannot be used practically if the degrees of freedom of fiber edges are large. To overcome this difficulty, we have developed an automatic fiber alignment system using genetic algorithms, which incorporate a special local learning method. In experiments, fiber alignment of five degrees of freedom can be completed within a few minutes, whereas it would take a human expert about half an hour.

1 Introduction

With the growth in optical fiber communications, fiber alignment has become the focus of much industrial attention[1]. This is a key production process because its efficiency greatly influences the overall production rates for the opto-electric products used in optical fiber communications. Fiber alignment is necessary when two optical fibers are connected, when an optical fiber is connected to a photo diode (PD) or a light emission diode (LED), and when an optical fiber array is connected to an optical wave guide.

Metallic wire connection is relatively easy because an electric current will flow as long as the two wires are in contact. The connection between two optical fibers, however, requires much greater precision, in the order of sub-micron-meters. Therefore, experienced technicians are needed for fiber alignment, but as such technicians are in limited supply, this causes a bottleneck in the mass production of opto-electric components for optical fiber communication. To overcome this, various algorithms for automatic alignment have been devised. However, existing automatic-alignment algorithms are only capable of aligning fibers according to three degrees of freedom (DOF). Thus, the present authors have devised a new algorithm based on GAs that is capable of aligning fibers according to five or more DOF. The advantages of the algorithm are summarized below:

1. Improvement in transmission efficiency
   Unless two optical fibers are connected in an optimal way, there will be a

P. Liardet et al. (Eds.): EA 2003, LNCS 2936, pp. 129–140, 2004.

reduction in transmission efficiency. However, as alignment is possible according to a greater DOF, there are no reductions in transmission efficiency.
2. Improvement in reliability
   Automatic alignment reduces the numbers of tasks to be conducted by technicians, which leads to the improved reliability of products.
3. Improvement in production efficiency
   The fiber alignment process is often a critical part in the entire production process, and so by reducing fiber-alignment times, production efficiency can be improved considerably.
4. Cost reductions
   Less expensive opto-electric components are often inferior in terms of precision. However, variations from the desired specifications for each component can be compensated by using our alignment algorithm for greater DOF connections. This makes it possible to reduce the overall costs of a system.

Conventional alignment systems cannot achieve connections with 5 DOF because such precise fiber alignment is impossible given practical time constraints. However, our algorithm successfully carried out alignments for 4 different initial conditions in a few minutes.

This paper is organized as follows: In section 2, the difficulty with the conventional fiber alignment system is briefly described. Section 3 introduces our GA-based alignment algorithm, and section 4 describes the results of alignment experiments. Finally, section 5 discusses the results and draws some conclusions.

2 Optical Fiber Alignment

2.1 Transmission Loss Due to Non-optimal Connections

Although metallic wire connections are very easy, optical fiber connections require extreme precision. This is because optical signals can only pass along the core of a fiber, which is only a few micron-meters in diameter. Therefore, when connecting optical fibers, the cores of the two fibers must be aligned to face each other exactly with sub-micron-meter precision. In making such alignments, there are a number of factors that must be considered which can reduce transmission efficiency, such as shifts in the light axis (Fig. 1(a)(b)), bending (Fig. 1(c)), and when core surfaces are not flush (Fig. 1(d)). To deal with these factors, alignment algorithms capable of greater DOF connections are required.

2.2 Automatic Alignment System

Automatic alignment systems can shift slightly the light axes of the two optical fibers to minimize transmission loss[2][3]. Once alignment is complete, the light axis is fixed by laser processing or a setting resin. Fig. 2 shows the organization of the typical automatic alignment system. The system consists of a light source, alignment stages, a stage controller, a power meter to measure the light intensity, and a controlling PC. An alignment stage moves the tip of one optical fiber with

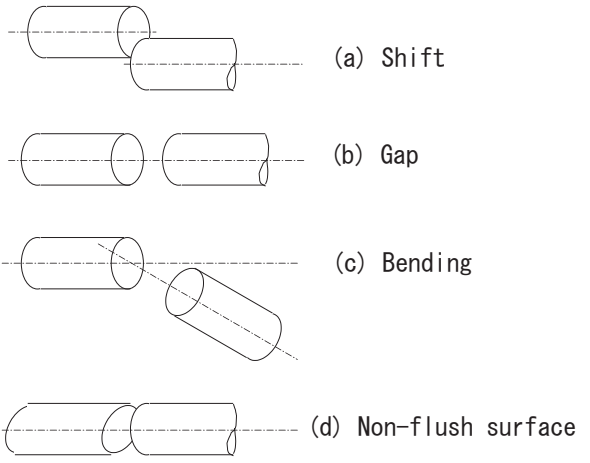


Fig. 1. Factors for connection loss

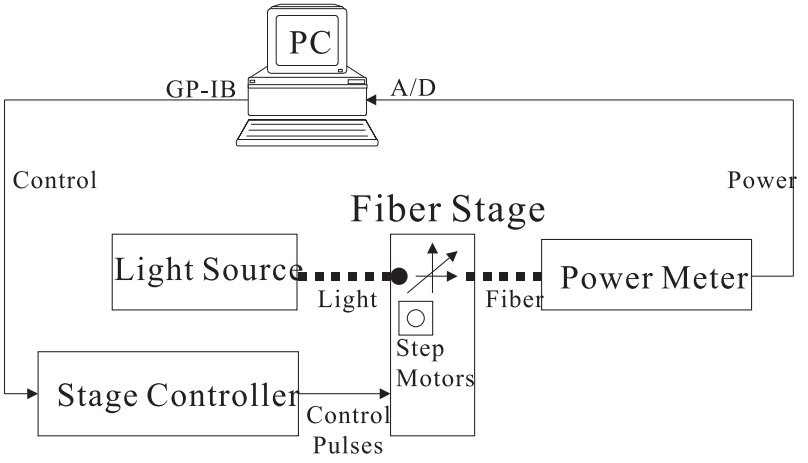


Fig. 2. Automatic fiber alignment system

micron-meter precision using step motors. The PC collects information from the power meter and feeds this back to the stage controller through a GP-IB interface in order to control the alignment stage. The control signals are generated by the PC where the automatic-alignment algorithm is executed.

2.3 Conventional Automatic Alignment Algorithms

Alignment time depends completely on the performance of automatic alignment algoritms. Various algorithms are available on the market. For example, the alignment software sold by Suruga Seiki Co.Ltd. [2] provides two search pro-

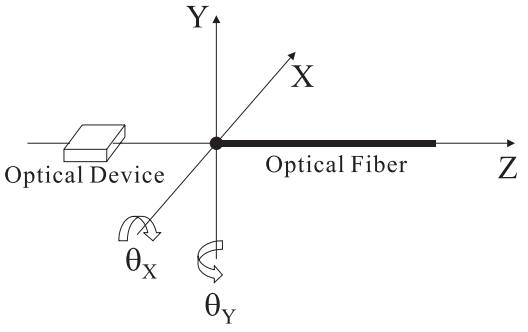


Fig. 3. Optical axis

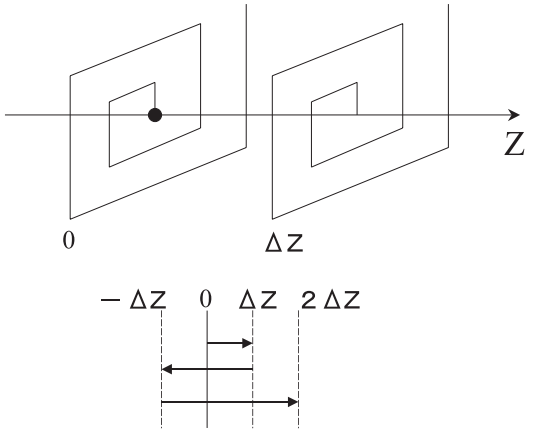


Fig. 4. Field search routine

grams; a field search and a peak search program. The field search program is used when the two light axes are shifted greatly. The program searches the light by moving the X,Y,Z axes in Fig. 3 in spiral way as shown in Fig. 4. Once the light, of which intensity level is higher than the noise level, has been detected, the peak search program is initiated. The program iterates sequential optimization of the X,Y,Z axes.

However, these programs have the following weakness; (1) local optimum in the intensity distribution due to the light interference, (2) flatness in the intensity distribution for angle adjustments, and (3) noise-sensitivity. Due to these weaknesses, these software programs are not capable of alignments according to 5 DOF, have long alignment times, and result in large transmission losses.

3 Automatic Alignment with GA

Our alignment algorithm uses a GA called the Minimal Generation Gap (MGG) model[4], a kind of non-generational GAs. The MGG model has two advantages; (1) it can maintain the diversity within a population by simple operations, and (2) the convergence speed is relatively fast. We also employ a speacial local learning method in order to improve search efficiency.

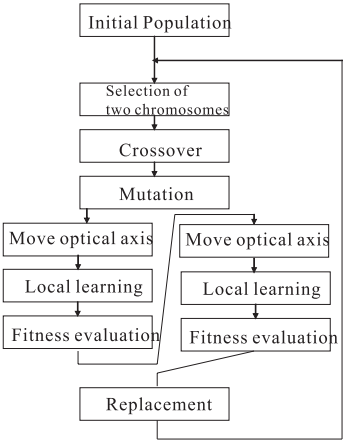


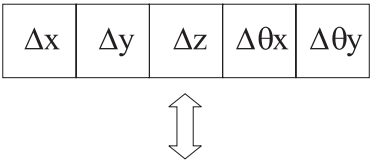
Fig. 5. Flowchart of fiber alignment based on GAs

Fig. 5 gives a flowchart of our alignment algorithm. The positions of the light axes are coded as chromosomes, and then an initial population with the size of N_p is generated. Fig. 6 describes the coding. The chromosome consists of 5 genes, which correspond to 5 DOF (i.e. X,Y,Z,θ_X, θ_Y in Fig. 3). Each gene represents a difference (32-bits real value) from the initial position before alignment. The initial gene values are set to uniform random values between $-L_i$ and $+L_i (i = 1, \cdots, 5)$.

The crossover is the single-point cross-over with a probability of P_c, the mutation is the Gaussian mutation[5] with a probability of P_m. The fitness value is the light intensity obtained from the power meter. For example, if the output of the power meter is –10dB, then the fitness value is –10.0.

3.1 Local Learning

To accelerate the alignment, we have adopted a special local-learning method, which utilizes measurements of the output power taken while the step motors are moving according to a newly set gene value. Because it takes more than one second for the stepmotors to reconfigure the positioning of the fibers, fiber output can be repeatedly measured more than 10 times to provide evaluations



$x=x_0+\Delta x$, $y=y_0+\Delta y$, $z=z_0+\Delta z$, $\theta x=\theta x_0+\Delta\theta x$, $\theta y=\theta y_0+\Delta\theta y$

Fig. 6. Coding to chromosomes

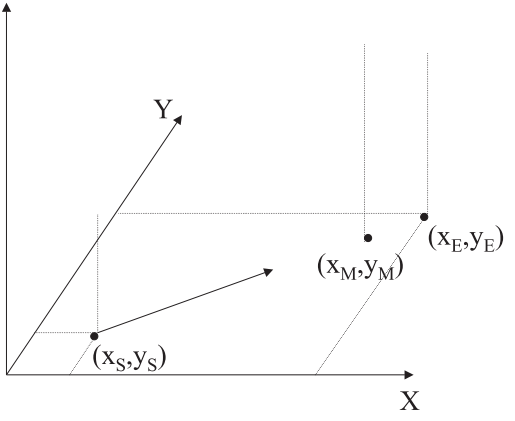


Fig. 7. Local learning

of intermediate configuration positions. The local-learning method compares the fitness function values for these intermediate configurations, and replaces current chromosome with the chromosome representing the stage positions that gave the local maximum output power value, which leads to an improvement in search efficiency.

Using Fig. 7 as an example, the local-learning method for X-Y axes alignment are described. First, the chromosome to be evaluated is taken to be (X_E, Y_E), and the present position of the stage is taken to be (X_S, Y_S). Then, the X,Y stages are gradually moved by the step motors from (X_S, Y_S) to (X_E, Y_E). On this route, there is a case when the power value F_E at (X_E, Y_E) is less than a intermediate power value F_M at (X_M, Y_M) as shown in Fig. 7. Therefore, to improve the search efficiency, the intermediate power value F_k are measured by the power meter, and the corresponding stage positions (x_k, y_k) are stored in pairs on a memory. Then, after the stages reached at (X_E, Y_E), the highest power value F_M is selected as a local maximum among $F_E, F_k (k = 1, \cdots, n)$, and the evaluated chromosome is rewritten to corresponding positions (x_M, y_M).

4 Alignment Experiments

4.1 Experimental System

Fig. 8 shows the experimental alignment system for development. It consists of a light source, a power meter, a control PC, and the stage controller. As shown in Fig. 9, the stage is integrated with five step motors. The resolution of X,Y,Z stages is 0.05μm, the resolution of θ_X, θ_Y is 0.003 deg.

The objective of the experiments was to align the tips of two optical fibers with lens according to the five DOF.

4.2 Experimental Conditions

Four experiments were conducted in which we prepared 4 different initial conditions by altering the light axes from the positions aligned by an experienced engineer. At the initial positions, the power level is slightly higher than the noise level. The parameters used at the experiments were $L_1 = 15.0\mu m$, $L_2 = 15.0\mu m$, $L_3 = 100.0\mu m$, $L_4 = 0.5$deg, $L_5 = 0.5$deg, $N_P = 20$, $P_c = 0.5$, $P_m = 1.0$, $n = 4$. The motor speed was 4000PPS (Pulse Per Second).

In each experiment, alignment was successfully completed in three minutes. The details of the experiments are described below.

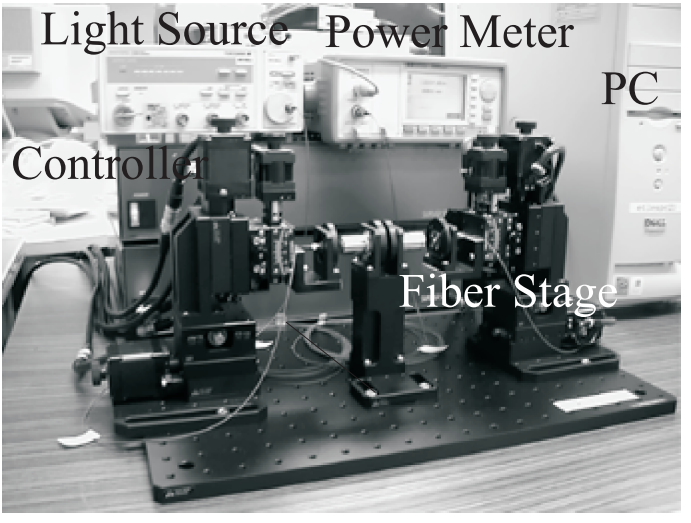


Fig. 8. Fiber alignment system for development

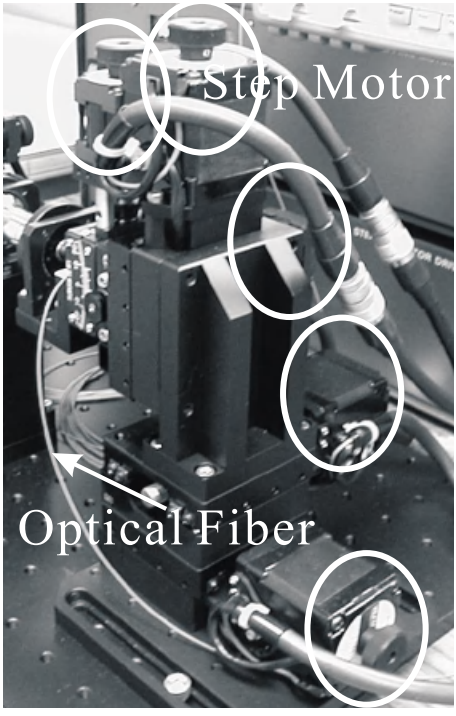


Fig. 9. Stage for fiber alignment

Table 1. Alignment Results for the Experiment 1

Power(dBm)	Avg.	Max.	Min.	Std.
Conventional (Peak Search)	2.58	3.94	1.23	0.73
GA without LL (Local Learning)	3.26	4.31	2.44	0.66
GA	4.41	4.73	3.85	0.27
HC (Hill Climbing)	3.83	5.00	1.52	1.08

4.3 Experiment 1

The initial position was $(X, Y, Z, \theta_X, \theta_Y) = (10, 10, -80, 0, 0)$. This position was set by altering only X,Y, and Z axes. We show the result of 10 trials in Table 1. The best average performances was attained by the proposed automatic alignment method. From the comparison of results by GA and GA without the local learning (LL), we can see the local learning worked effectively. Moreover, from the comparison of results by GA and the hill climbing method (HC), we can see the GA avoided the local minimums. The HC employed the Gaussian mutation in our GA as the state transition operator.

Figure 10 shows how the optical fiber tips converged in the experiment. The conventional peak search algorithm converged with about three minutes.

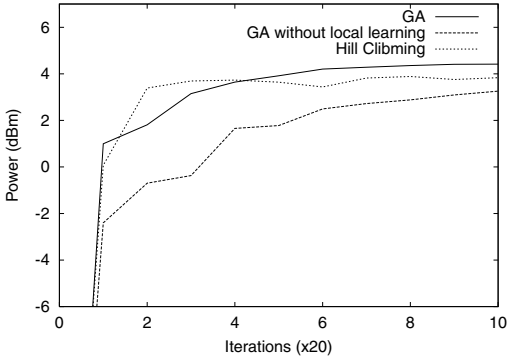


Fig. 10. Power versus Iterations for Experiment 1

The other methods took about seven minutes for 10 generations. However, after fourth generation (about three minutes), the GA outperformed the conventional algorithm and attained practical intensity level.

Table 2. Alignment Results for the Experiment 2

Power(dBm)	Avg.	Max.	Min.	Std.
Conventional	0.96	1.55	-0.34	0.60
GA without LL	2.22	3.30	1.21	0.56
GA	3.12	3.52	2.37	0.36
HC	2.30	3.67	-0.39	1.37

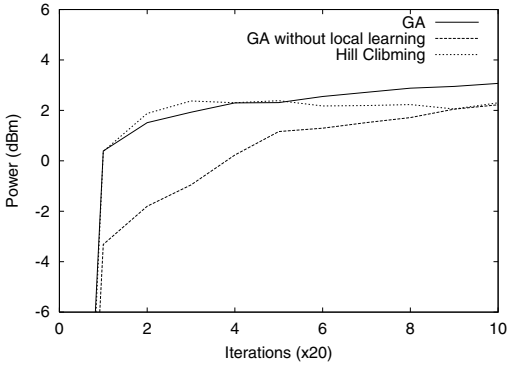


Fig. 11. Power versus Iterations for Experiment 2

4.4 Experiment 2

The initial position was $(X, Y, Z, \theta_X, \theta_Y) = (2, -2, 80, -0.4, -0.4)$. This position was set by altering Z,θ_X, θ_Y axes largely. We show the result of 10 trials in Table 2 and Fig. 11. The best average performances was attained by our GA. The performance of the conventional algorithm was naturally worst, because it could not align the θ_X, θ_Y axes.

Table 3. Alignment Results for the Experiment 3

Power(dBm)	Avg.	Max.	Min.	Std.
Conventional	1.22	1.76	0.60	0.36
GA without LL	2.09	3.13	1.19	0.61
GA	3.28	3.57	2.97	0.20
HC	2.91	4.40	1.66	0.79

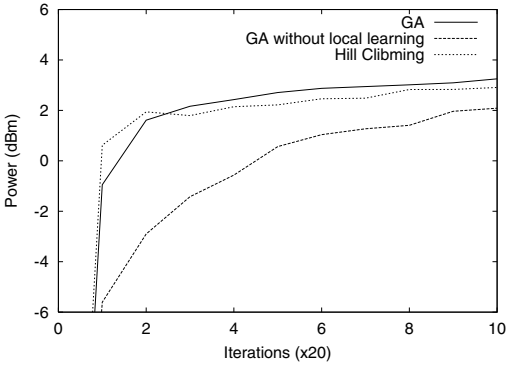


Fig. 12. Power versus Iterations for Experiment 3

4.5 Experiment 3

The initial position was $(X, Y, Z, \theta_X, \theta_Y) = (-10, -2, 0, -0.4, 0.4)$. This position was set by altering X,θ_X, θ_Y axes largely. We show the result of 10 trials in Table 3 and Fig. 12. The best average performance was attained by our GA the same as in the experiments 1 and 2.

4.6 Experiment 4

The initial position was $(X, Y, Z, \theta_X, \theta_Y) = (0, 0, 0, -0.8, 0.8)$. This position was set by altering θ_X, θ_Y axes greatly. For this experiment, the L_4 and L_5 were extended to 1.0deg. We show the result of 10 trials in Table 4 and Fig. 13. The

Table 4. Alignment Results for the Experiment 4

Power(dBm)	Avg.	Max.	Min.	Std.
Conventional	-1.14	-0.74	-1.81	0.34
GA without LL	2.42	3.39	0.89	0.84
GA	3.20	3.55	2.84	0.26
HC	-0.66	3.12	-14.6	5.11

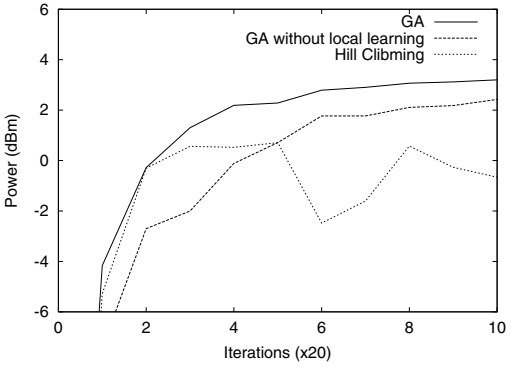


Fig. 13. Power versus Iterations for Experiment 4

best average performance was attained by our GA. Large standard deviation of the results by HC suggests that this experimental condition had many local minimums in the fitness function.

In every experiment from 1 to 4, the standard deviation by our GA is smallest in the four algorithms. This indicates that our GA is robust to the noise. However, the best result for 10 trials is highest with HC (except in experiment 4). So, our GA combined with HC might improve the search efficiency.

5 Conclusion

This paper has described an automatic fiber alignment system using genetic algorithms. The experimental results show that our system can align optical axis according to the five DOF with three minutes from four different initial conditions. The results also show that our local learning algorithm can speed up the alignment.

Our method can be applied not only to the fiber alignment, but also to alignment of optical components. An femtosecond laser system has been developed at AIST, in which the positioning of laser components such as mirrors and prisms can be aligned automatically[6]. The GA-based adjustment can align the laser components automatically within 10 minutes, whereas it often takes a week for experienced technicians in order to achieve optimal performance. Thus, GAs

are effective in alignment of complicated optical systems with multi-DOF. Our method opens up the possibility of developing new optical instruments in a wide variety of industrial applications in the future.

Acknowledgments. This work was partly supported by Industrial Technology Research Grant Program in 2002 from New Energy and Industrial Technology Development Organization (NEDO) of Japan.

References

1. Hayes, J.: Fiber Optics Technician's Manual 2nd edition. Delmar Publishers (2001).
2. http://www.suruga g.co.jp/jp/suruga/ost/wb/wb_home.html: (Auto alignment system).
3. http://www.moritex.co.jp/english/e_products/frame_main_opt.html: (Automated optical alignment)
4. Satoh, H., Yamamura, M., Kobayashi, S.: Minimal generation gap model for gas considering both exploration and exploitation. In: Proceedings of the Fourth International Conference on Soft Computing (IIZUKA 96). (1996) 494–497.
5. Schwefel, H.P., ed.: Evolution and Optimum Seeking. John Wiley & Sons (1995).
6. Murakawa, M., Itatani, T., Kasai, Y., Kikkawa, H., Higuchi, T.: An evolvable laser system for generating femtosecond pulses. In: Proceedings of the Second Genetic and Evolutionary Computation Conference (GECCO2000), Morgan Kaufmann (2000) 636–642.

Large-Scale Scheduling of Casting Sequences Using a Customized Genetic Algorithm

Kalyanmoy Deb and Abbadi Raji Reddy

Kanpur Genetic Algorithms Laboratory (KanGAL)
Indian Institute of Technology Kanpur
Kanpur, PIN 208 016, INDIA
deb@iitk.ac.in
http://www.iitk.ac.in/kangal/deb.htm

Abstract. Scheduling a sequence for molding a number of castings each having different weights is an important large-scale optimization problem often encountered in foundries. In this paper, we attempt to solve this complex, multi-variable, and multi-constraint optimization problem using different implementations of genetic algorithms (GAs). In comparison to a mixed-integer linear programming solver, GAs with problem-specific operators are found to provide faster (with a sub-quadratic computational time complexity) and more reliable solutions to very large-sized (over one million integer variables) casting sequence optimization problems. In addition to solving the particular problem, the study demonstrates how problem-specific information can be introduced in a GA for solving large-sized real-world problems efficiently.

1 Introduction

Optimal scheduling problems often arise in different real-world problem solving activities and are routinely solved using classical search and optimization algorithms including linear programming methods. The difficulties often faced in solving such problems are (i) dimensionality of the search space and (ii) integer restriction of the decision variables. If the resulting problem is linear (that is, the objective function and constraints are all linear functions of the decision variables), the linear programming (LP) approaches are ideal candidates to solve such problems (Taha, 1989). Although the first difficulty is usually not a matter for solving such problems using an LP, the second difficulty requires an LP approach to be used with an integer programming approach, such as the branch-and-bound method. Since the branch-and-bound approach requires branching every non-integer variable into two different LPs, the presence of a large number of integer decision variables demands an exponentially large number of function evaluations to solve the problem to optimality.

For the past few decades, optimal scheduling problems have also been solved using various non-traditional methods, such as simulated annealing, genetic algorithms, tabu search etc. (Kirkpatrick, Gelatt and Vecchi, 1983, Goldberg, 1989;

P. Liardet et al. (Eds.): EA 2003, LNCS 2936, pp. 141–152, 2004.

Glover, 1997). In some of these methods, although the second difficulty of handling integer variables is not a matter, the first difficulty of handling a large number of decision variables is not well researched.

In this paper, we tackle one such scheduling problem in which the purpose is to find an optimal casting sequence often encountered in an automated foundry. Metal is melted in a large crucible and the molten metal is used to cast a number of molds. One such melting operation is called a *heat*. With a restriction of a finite sized crucible used in a heat, it is desired to find a schedule for casting different molds from a number of heats. It turns out that the resulting problem is a mixed-integer linear program (MILP).

In the remainder of the paper, we have designed new GA recombination and mutation operators using problem-specific information. Since the overall objective function is an average utilization of molten metal in individual heats, the best partial sequences from two existing solutions can be combined together to form a new solution. The mutation operator attempts to make the new solution obtained in the recombination operator feasible. An innovative method of generating the initial population is also suggested. Using this modified GA, the MILP problem having the number of variables as large as 1,006,750 (more than a million!) is solved resulting in 99.84% metal utilization. This study remains as one of the largest (if not, the largest) problems which has been reported to solve by any evolutionary algorithm till to date.

2 Scheduling of Casting Sequences

In a typical foundry, casting of various sizes are made from a heat by melting metal in a large crucible of size W. We assume that the total number of castings to be made is so large that a multi-day schedule requiring a total of H heats is necessary. Let us also assume that the casting k having a weight w_k has a demand of r_k copies. Thus, the total demand is $R = \sum_{k=1}^{K} r_k$ (where K is the total number of different castings) and the overall amount of metal required to make all castings is $\sum_{k=1}^{K} r_k w_k$.

With the above parameters, one can schedule a casting sequence for multiple days by introducing decision variables x_{ki} denoting the number of copies of casting k made from the i-th heat. Using this variable, we write the underlying optimization problem as follows:

$$\left.\begin{array}{ll} \text{Maximize } \frac{1}{H}\sum_{i=1}^{H} \dfrac{100\sum_{k=1}^{K} w_k x_{ki}}{W_i}, & \\ \text{Subject to } \sum_{i=1}^{H} x_{ki} = r_k, & \text{for } k = 1, 2, \ldots, K, \\ \sum_{k=1}^{K} w_k x_{ki} \leq W_i, & \text{for } i = 1, 2, \ldots, H, \\ x_{ki} \geq 0, & \\ x_{ki} \text{ is an integer.} & \end{array}\right\} \quad (1)$$

It is obvious from the above formulation that the problem is a linear programming (LP) problem. There can be two difficulties that an LP solver can face to solve the above problem:

1. The number of decision variables and constraints are large, and
2. All decision variables are integer-valued.

The total number of decision variables are $n = HK$ and total number of constraints are $(H + K)$. However, since an LP solver will first convert an inequality constraint into an equality constraint using a slack variable, the total number of decision variables for an LP solver become $n = (K + 1)H$. Although these numbers may not be daunting for a reasonable number of heats and castings, the integer restriction of the variables is a serious matter. The commonly-used technique to handle such decision variables is the branch-and-bound (BB) method (Deb, 1995; Reklaitis, Ravindran and Ragsdell, 1983). It is known that the BB method is exponential to the number of variables.

To investigate the efficacy of such a mixed-integer LP technique, we have applied LINGO (a commonly-used integer programming software) to the above problem. We have tried to solve a problem with $K = 10$ castings (having a demand of $r = \{7, 7, 6, 9, 9, 5, 2, 2, 7, 6\}$ copies). The weight of each casting varies as follows: $w_k = \{175, 150, 85, 75, 95, 76, 94, 200, 85, 50\}$ kg. We have used a crucible of size $W = 650$ kg for each heat and allowed 10 heats in total to make all castings. The LINGO requires 592 function evaluations to arrive at a solution having an overall efficiency of 95.05%. Table 1 shows the obtained solution. The optimum efficiency achievable with 10 heats is $100 \sum_{k=1}^{10} r_k w_k/(10 \times W)$ or 95.05%, which is what the LINGO has found. However, with the standard input-output procedure of using the LINGO software, we were *unable* to find a feasible solution (integer values on all variables) for a problem beyond 200 decision variables in any reasonable number of function evaluations and time (up to seven hours on 1.7 GHz Pentium IV processor). The main reason is the use of the exponential branch-and-bound algorithm used in LINGO.

Table 1. The solution obtained by LINGO.

Heat	Casting Number										Utilization/	Efficiency
No.	1	2	3	4	5	6	7	8	9	10	Cruc. Size	(%)
1	0	1	1	0	0	0	2	1	0	0	623/650	95.85
2	2	0	0	0	1	0	0	0	2	0	615/650	94.62
3	1	0	0	1	3	1	0	0	0	0	611/650	94.00
4	2	0	0	0	1	0	0	1	0	0	645/650	99.23
5	0	0	0	1	0	2	0	0	1	6	612/650	94.15
6	1	1	0	0	2	1	0	0	0	0	591/650	90.92
7	0	0	2	2	1	0	0	0	2	0	585/650	90.00
8	0	3	0	0	0	1	0	0	1	0	611/650	94.00
9	0	2	3	0	1	0	0	0	0	0	650/650	100.00
10	1	0	0	5	0	0	0	0	1	0	635/650	97.69
	7	7	6	9	9	5	2	2	7	6	Average	95.05

3 Genetic Algorithm Approaches

In order to investigate how a standard GA would perform to solve the mixed-integer linear program given in (1), we first use a binary-coded GA, where each decision variable is coded in a 4-bit binary substring. The decoded value of the 4-bit substring is used as the corresponding value of x_{ki}. Thus, each x_{ki} takes an integer in the range $[0, 15]$. For n variables, we require a total of $\ell = 4n$ bits. Each equality constraint is used to eliminate one decision variable. Only inequality constraints are handled by using the penalty function approach (Deb, 1995). Each constraint is normalized and total normalized constraint violation is multiplied by a factor 1,000 and subtracted from the objective function value. (Note that the objective function is maximized here.) Standard GA operators such as the tournament selection operator, single-point recombination (with a probability 0.8) and bit-wise mutation (with a probability of $1/\ell$, where ℓ is the string length) are used. We always keep a fixed number of castings ($K = 10$), but vary the total number of heats H (thereby allowing to vary the number of copies desired for each casting), so that the total number of decision variables (n) vary between 100 to 300. Binary-coded GAs with an increasing population size (from 100 to 1,000) for larger sized problems are used and GAs are run till a feasible solution is found. Table 2 shows the population size and the number of function evaluations (average of 5 runs is tabulated) needed in each case. Although a GA with a random initial population and standard operators can solve 200 or 300 variable problems, but the required number of function evaluations increases exponentially. A real-coded GA (with SBX ($\eta_c = 0.3$) and polynomial mutation ($\eta_m = 0.1$) operators (Deb, 2001)) also shows similar performance. These results are not surprising. The generic binary and real-coded GAs are not particularly designed to solve linear problems. To make GAs useful in solving these problems and make GAs competitive to a classical optimization approach, their operators must be redesigned with problem-specific information.

Table 2. Average function evaluations needed to find feasible solution using standard binary-coded and real-coded GAs.

	Binary-coded GAs			Real-coded GAs		
Number of Variables	Population Size	Efficiency	Function Eval.	Population Size	Efficiency	Function Eval.
100	100	96.15	13,600	100	95.94	23,740
200	300	95.01	1,42,200	200	92.81	1,21,760
300	1,000	90.11	14,12,400	700	95.14	5,84,220

3.1 Using Problem-Specific GA Operators

In this subsection, we propose a new customized GA which starts with a biased initial population and knowledge-specific recombination and mutation operators.

Initial Population Generation. The optimization problem described in (1) involves many constraints, all of which must be satisfied by a feasible solution. However, since the constraints are linear, a simple methodology can be adopted to ensure that the first set of constraints are satisfied in all initial solutions. First, every variable x_{ki} is initialized within $[0, a]$ (a being an upper limit to the number of copies of a casting k that can be made from one heat). Thereafter, we normalize each entry x_{ki} such that the total number of copies allocated for a casting k is the same as that desired (r_k), thereby satisfying the equality constraints given in equation 1.

Recombination Operator. The main purpose of a recombination operator is to recombine partial good information of two or more solutions and create an offspring solution. In the current context, a solution can be called good, if the overall utilization of molten metal is close to 100%, so that the corresponding casting strategy causes minimal waste of energy. However, a solution in an intermediate generation may not contain close to 100% utilization in all its heats, but may have close to 100% utilization in some of its heats. These partial solutions, in which a good utilization has been already achieved, are *building blocks* of the problem and should be propagated from parent solutions to their offspring. Emphasizing and propagating such partial good sub-solutions through generations will enable GAs to *jump* towards the optimum by making more and more heats to reach better and better utilization levels. Such a procedure should work fantastically well in linear or linear-like problems, similar to the one described in equation 1.

The population members are ordered randomly and every consecutive pairs of solutions are chosen for recombination, in which the sub-solution (x_{ki} for all castings used in the heat) is simply chosen from that parent which produces a better heat-wise utilization of molten metal. Although this is a simple recombination procedure, the offspring need not necessarily be a feasible solution satisfying all the given constraints, despite the two parent being feasible. We fix such a solution in the mutation operator described below.

Mutation Operators. The main purpose of a mutation operator is to perturb a solution obtained by the recombination operator in the hope of creating an even better solution. In this spirit, we develop two mutation operators (applied to every solution) to make an infeasible solution created by the recombination operator feasible.

In the first mutation operator, the decision variables of a solution are altered so as to satisfy the equality constraints. For a solution, all x_{ki} values for each casting k are added and compared with its demand r_k. If the added quantity is the same as r_k, the corresponding equality constraint is satisfied and no modification is made. If the added quantity is larger than r_k, then the individual normalized utilization of molten metal ($U_i = (W_i - \sum_{k=1}^{K} w_k x_{ki})/W_i$ for the i-th heat) is compared against zero. The heats having an U_i value smaller than zero violates the corresponding inequality constraint. Thus, we have the freedom to

reduce the number of copies to be made from those heats. When there exist more than one heat violating their inequality constraints, any one of the heats can be chosen to reduce the corresponding x_{ki}. Here, we choose to reduce one copy of the k-th casting from the heat ($i_{\min}$) which violates the inequality constraint maximally. Mathematically, $x_{ki_{\min}}$ is reduced by one. The utilization $U_{i_{\min}}$ with the modified number of copies for the $i_{\min}$-th heat is recalculated and the same procedure is repeated till the equality constraint for the k-th casting is satisfied. However, there is one difficulty with this procedure. If $x_{ki_{\min}}$ is equal to zero to start with, the variable $x_{ki_{\min}}$ cannot be reduced any further. In this case, we follow the above procedure with the heat having the second-highest constraint violation, and so on. If $U_i \geq 0$ for all heats, the heat ($i_{\max}$) having maximum U_i is chosen and the corresponding $x_{ki_{\max}}$ is reduced by one, provided $x_{ki_{\max}} > 0$; else the second-highest U_i is chosen and so on. This procedure is continued till the k-th equality constraint is satisfied.

On the other hand, if the added quantity is smaller than r_k then the reverse of the above procedure is followed. First, the heat $i_{\max}$ having maximum U_i (but not violating the corresponding inequality constraint) is chosen and $x_{ki_{\max}}$ is increased by one. After performing these operations for all castings, if the solution is still infeasible, we declare this solution as an infeasible solution by associating with it a constraint violation of $\left|r_i - \sum_{i=1}^{H} x_{ki}\right|/r_i$.

Next, the modified solution is sent to the second mutation operator which attempts to alter the decision variables further without violating the equality constraints in order to satisfy the inequality constraints. The second mutation operator is started using the first heat and continued serially to the final heat. Each heat is checked for its inequality constraint violation. If $U_i \geq 0$, the constraint is satisfied and there is no need to modify the heat. On the other hand, if $U_i < 0$, we look for the heat ($i_{\min}$) which has the minimum feasible utilization. To make U_i feasible, we now have to transfer one copy of the casting k from the i-th heat to the $i_{\min}$-th heat, irrespective of whether $U_{i_{\min}}$ is greater than or lesser than one. Here, we choose a random casting k for which $x_{ki} > 0$. After all infeasible heats are rectified as above, the feasibility of the overall solution is checked. If the solution is found infeasible, a constraint violation equal to the sum of the normalized violations of all infeasible heats is assigned.

Heat-Updates. The objective function is a summation of independent heat utilization values (or efficiencies), divided by the total number of heats. Among all GA operators, the individual heat utilization values are computed only in the mutation operators. Once the individual heat updates are made and the revised utilization values are computed, the overall objective function is easy to calculate. Since one solution may require many such heat-updates in its mutation operators and one solution may not require updates of all its heats to make the solution feasible, the total number of heat-updates can be used as a measure of the computational complexity of the proposed approach. But it should be made clear that it is difficult to compare the total number of heat-updates needed to find an optimum in the proposed approach with the total number of function

evaluations needed for the purpose usually reported using other optimization methods.

However, it is obvious that a large-sized problem involving a large number of heats will naturally require a large number of heat-updates to find the optimum. To compare the performance of the proposed approach with an iterative method which updates all H heats in one iteration, a *normalized heat-update* measure, calculated by dividing the total heat-updates by the total number of heats (H), is used here. This measure will give an estimate of the equivalent number of iterations needed to find the optimum solution by the proposed approach.

4 Simulation Results

Here, we show simulation results of the proposed approach including a scale-up study by varying the number of decision variables in the range 100 to 1,006,750.

4.1 Proof-of-Principle Results

First, we apply the proposed GA to the 100-variable problem solved using the LINGO earlier. With a population size of 10, the proposed GA finds the optimal solution (having an identical efficiency of 95.05%) in an average of only 32.68 normalized heat-updates (total heat-updates divided by the number of heats) from 10 independent runs, requiring only an average of 2.1 generations. One of the GA solutions is shown in Table 3. The average of total number of heat-updates required by the proposed GA is 326.8, which is smaller than the 592 *complete* function evaluations (note that each function evaluation will usually require more than one heat-update) needed by the LINGO.

Table 3. The solution obtained by the proposed GA approach.

Heat	Casting Number										Utilization/	Efficiency
No.	1	2	3	4	5	6	7	8	9	10	Cruc. Size	(%)
1	1	0	1	3	1	0	0	0	0	0	580/650	89.23
2	1	1	0	1	1	1	0	0	0	1	621/650	95.54
3	1	0	1	0	1	1	0	1	0	0	631/650	97.08
4	1	0	0	1	1	0	0	0	3	1	650/650	100.00
5	0	2	1	1	1	0	0	0	0	0	555/650	85.38
6	1	1	0	0	1	0	1	0	1	1	649/650	99.85
7	0	1	0	1	1	1	0	0	2	1	616/650	94.77
8	1	1	1	0	0	0	1	0	1	1	639/650	98.31
9	1	0	1	1	1	0	0	1	0	0	630/650	96.92
10	0	1	1	1	1	2	0	0	0	1	607/650	93.38
	7	7	6	9	9	5	2	2	7	6	Average	95.05

4.2 Generation-Wise Search Effort

In order to investigate the number of generation-wise heat-updates required in a GA run, we have considered a 2,000-variable problem having weight w_k of castings chosen randomly in the range $[50, 200]$ kg. The number of castings is 10 and two crucibles of weights 650 and 500 kg are used alternatively. A population of size 30 is used here. It is observed that a total of 200 heats are necessary to find a solution with 99.83% efficiency. The normalized heat-updates required in each generation are plotted in Figure 1. It can be seen from the figure that more heat-updates are necessary during the early generations. Since the early solutions are usually far from being feasible, a considerable number of heat-updates are necessary to fix the equality and inequality constraints. When solutions approach the most efficient solution towards the latter generations, not much heat-updates are necessary. When all population members become feasible (happened at generation 32 in this case), no new heat-updates are necessary. However, in this particular simulation run, the first feasible solution was found at generation 16. Similar dynamics in the number of heat-updates versus generation number are observed in all experiments performed in this paper.

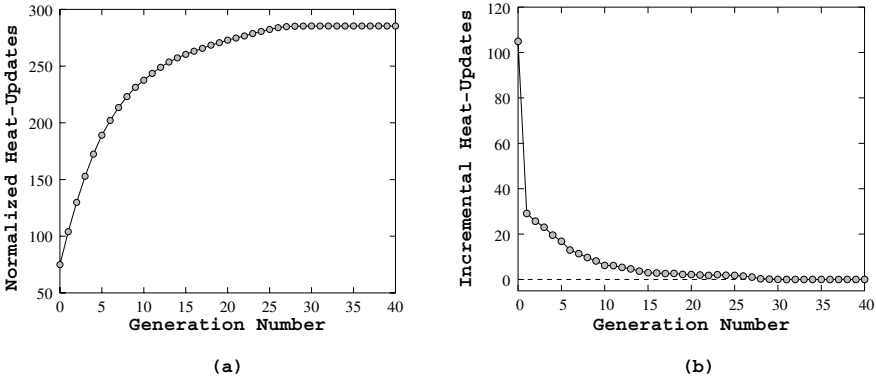


Fig. 1. The number of (a) normalized and (b) incremental heat-updates versus the generation number for a 2,000-variable problem.

In this problem, the total weight of all castings is $\sum_{k=1}^{10} w_k r_k = 113{,}280$ kg. To complete all castings, a total of 17-day schedule with 90 heats of 650 kg crucible and 110 heats of 500 kg crucible are suggested by the obtained solution, thereby amounting to a total of 200 heats. However, with one less heat or with 199 heats the total metal melted would be $(90 \times 650 + 109 \times 500)$ or 113,000 kg, which is 280 kg smaller than the overall required casting weight. Thus, the minimum number of heats required to find a feasible schedule is 200, which is also found to be the optimum by the proposed GA. Interestingly, the proposed GA has found a schedule having a little better objective value (99.83% utilization)

than the mean utilization value, which is $113,280/(90 \times 650 + 110 \times 500)$ or 99.81%.

4.3 Importance of Population Size

Next, we study the effect of population size on the heat utilization. For the same 2,000 variable problem used above, we have used different population sizes, but kept the total number of maximum allowed generations equal to $1,000/N$ (where N is the population size). For each population size, the GA is run to find the minimum number of heats needed to find a feasible solution. These values are shown below:

Population size:	4	8	12	16	20	25	40	60	100
Min. number of Heats:	211	204	201	201	200	200	200	200	200

As discussed earlier, for the 2000-variable problem, there exist no feasible solution with less than 200 heats. The above table shows that for small population sizes (up to 16), the proposed GA cannot find any feasible solution having 200 heats. But, GAs with a population size larger than or equal to 20 are able to find a feasible solution with the minimum possible number of heats (200, in this case). As discussed earlier, an interesting property of this problem is that with the number of heats larger than the optimal number of heats more feasible solutions exist in the search space, thereby making an optimization algorithm easier to solve the problem (but with a lesser efficiency). This is the reason why GAs with small population sizes cannot solve the problem with the optimal number of heats. However, when the population size is adequate, GAs consistently and reliably find feasible solutions with the optimal number of heats.

The corresponding efficiency obtained with each population size is shown in Figure 2. The efficiency obtained in the best solution increases with population size. Once again, the best, average, and the worst efficiencies obtained in each case are shown (but the difference is so small that in most cases they are not visible). These results agree with the past research studies (Deb and Agrawal, 1999; Goldberg, 1989) concluding that for a recombinative GA there exists a critical population size below which the recombination operator does not work well. With an adequate number of population members, the recombination operator has the flexibility to allow good solutions to exchange good partial information. As seen from the Figure 2 that beyond a critical population size there exists a large plateau in which the proposed GA works equally well.

4.4 Importance of Recombination Operator

To investigate the effect of the recombination operator, we have applied the proposed algorithm with and without the recombination operator on the problem having different number of variables. All GA parameters are kept the same for runs with and without the recombination operator. Both GAs are run for a fixed number of generations. Figure 3 shows that for a wide range of decision variables (between 100 to 10,000), the GA with only the mutation operator does

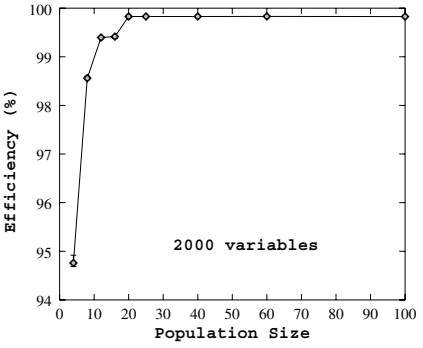


Fig. 2. Efficiencies achieved with different population sizes.

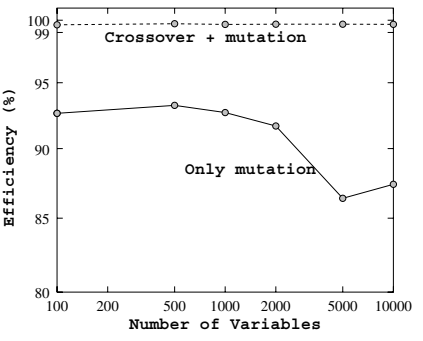


Fig. 3. Efficiencies achieved with different number of variables for GAs with and without the recombination operator.

not perform as good as that with recombination and mutation operators. These results show the importance of the suggested recombination operator in finding solutions with very high efficiencies.

4.5 Scale-up Study

In order to investigate the computational complexity of the proposed approach on increasing number of decision variables, we have solved the problem with variables in the range of a hundred to a million. For a population size of 30, we have taken 10 different runs for each case and run the proposed GA till a feasible solution with an efficiency above 99.65% is achieved. Figure 4 shows the average number of heat-updates over 10 runs for different problem sizes. The figure is plotted in logarithmic scales. A fitted straight line shows a polynomial complexity of heat-updates as $O(n^{0.946})$, where n is the number of integer variables. It is interesting to note that the proposed GA can solve a problem having as much as 1,006,750 integer decision variables in only about $1.6(10^7)$ normalized heat-updates in finding a solution with 99.84% efficiency, which is slightly better than the corresponding mean utilization value of 99.82% for this problem. Figure 5 shows the computational time needed to find the same optimum solutions on a 1.7 GHz Pentium IV processor. The figure shows a sub-quadratic computational time complexity of the proposed algorithm. The log-log plot shows a complexity of $O(n^{1.789})$. For the largest problem, the GA takes only an average of 6.81 hours of computational time. Since problems having 2,000 or less number of variables take less than one second of computational time, we do not show those results in Figure 5.

It is worth mentioning here that to our knowledge this is by far one of the largest problems which a GA has solved successfully and reliably. Where a commonly-used classical mixed-integer programming method has only been

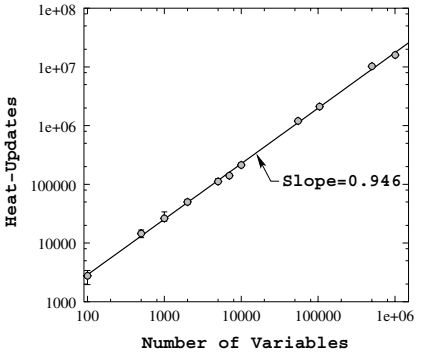


Fig. 4. Average, minimum, and maximum heat-updates needed to find a solution having at least 99.65% utilization of molten metal for different problem sizes.

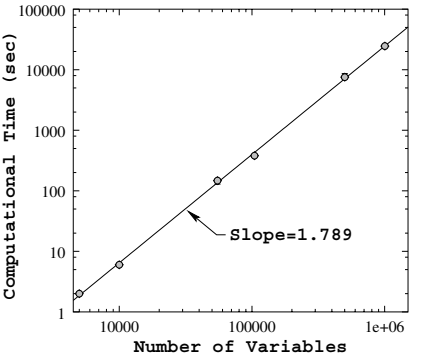


Fig. 5. Average, minimum, and maximum computational time needed to find the optimum solution.

able to solve the MILP problem with at most 200 integer variables, the proposed GA's success to solve the same MILP problem with more than one million integer variables in a reasonable computational effort marks a landmark achievement in the field of search and optimization.

5 Conclusions

Over the years, genetic algorithms (GAs) have been applied to numerous search and optimization problems. However, there exists very few studies where GAs have been tested on very large scale problems, such as problems having hundreds of thousands of variables. In this paper, we have shown that the standard GA implementations have difficulty in solving a real-world casting sequencing problem in a competitive manner. We have highlighted that GA operators designed with problem information must be used in order to solve them in any reasonable computational time. Based on one such set of problem-specific GA operators and an initial population generation procedure, we have been able to solve mixed-integer linear programming problems having as much as 1,006,750 variables in a sub-linear order of function evaluations and a sub-quadratic computational time complexity. Parametric studies with the population size and importance of the recombination operator have also been performed. Here are the outcomes of our study:

1. The proposed GA finds the optimum solution in a computational complexity much smaller than a classical branch-and-bound based LP solver.
2. The computational complexity of the proposed GA is sub-linear in terms of required function evaluations and sub-quadratic in terms of computational time for a wide range $[100, 1006750]$ of decision variables.

3. The new recombination operator developed here is the key operator for the proposed GA's success, although mutation operators suggested here are helpful but are not adequate to solve the problem efficiently by itself.
4. There exists a critical population size beyond which the proposed recombination operator works well.
5. In all problems tried in this paper, very high quality solutions (near 100% utilization of molten metal) are found.

To our knowledge, GAs have not been applied to such large-sized problems (with more than one million integer variables) with success and reliability too often in the past. A careful design of GA operators is the key for using GAs to large-scale search and optimization problems. The customized GA procedure proposed here may also have limited applicability to other more generic integer LP problems. The computational time can be reduced further by using a distributed computing approach. Hopefully, this paper has demonstrated the power of a recombinative GA in solving large-sized problems to optimality and would motivate the design and application of GAs to similar other large-scale real-world optimization problems.

References

Deb, K. (1995). *Optimization for engineering design: Algorithms and examples.* New Delhi: Prentice-Hall.

Deb, K. (2001). *Multi-Objective Optimization Using Evolutionary Algorithms.* Chichester, UK: Wiley.

Deb, K. and Agrawal, S. (1999). Understanding interactions among genetic algorithm parameters. In *Foundations of Genetic Algorithms 5 (FOGA-5)*, pp. 265–286.

Deb, K. and Goyal, M. (1998). A robust optimization procedure for mechanical component design based on genetic adaptive search. *Transactions of the ASME: Journal of Mechanical Design, 120*(2), 162–164.

Glover, F. and Laguna, M. (1997). *Tabu search.* Boston: Kluwer.

Goldberg, D. E. (1989). *Genetic Algorithms for Search, Optimization, and Machine Learning.* Reading, MA: Addison-Wesley.

Holland J. H. (1975). *Adaptation in Natural and Artificial Systems.* Ann Arbor, MI: MIT Press.

Kirkpatrick, S., Gelatt, C. D., Vecchi, M. P. (1983). Optimization by simulated annealing. *Science, 220*, 671–680.

Michalewicz, Z. (1992). *Genetic Algorithms + Data Structures = Evolution Programs.* Berlin: Springer-Verlag.

Mitchell, M. (1996). *Introduction to Genetic Algorithms.* Ann Arbor, MI: MIT Press.

Reklaitis, G. V., Ravindran, A. and Ragsdell, K. M. (1983). *Engineering Optimization Methods and Applications.* New York: Wiley.

Taha, H. A. (1989): *Operations Research.* New York: Macmillan.

Evolutionary Mining for Image Classification Rules

Jerzy Korczak and Arnaud Quirin

LSIIT, CNRS/Université Louis Pasteur,
Pôle API, Boulevard Sébastien Brant F-67400 Illkirch cedex
{korczak,quirin}@lsiit.u-strasbg.fr

Abstract. In this article, an approach for creating image classification rules using evolutionary operators is described. Classification rules, discovered by application of a genetic algorithm on remote sensing data, are able to identify spectral classes with comparable accuracy to that of a human expert. Genetic operators and the fitness function are detailed, and then validated for hyperspectral images (more than 80 spectral bands). Particular attention is given to mutation operators and their efficiency in the creation of robust classification rules. In our case studies, the hyperspectral images contain voluminous, complex and frequently noisy data. The experiments have been carried out on remote sensing images covering zones of Lagoon of Venice and the city of Strasburg, France. It has been shown that the evolution-based process can not only detect and eliminate noisy spectral bands in remote sensing images but also produce comprehensive and simple rules which can be also applied to other images.

Keywords: Remote sensing image, classification rules, high resolution image, hyperspectral image, supervised learning, evolutionary learning, genetic algorithm.

1 Introduction

The design of robust and efficient image classification algorithms is one of the most important issues addressed by remote sensing image users. For many years, a great deal of effort has been devoted to generating new classification algorithms and to refine methods used to classify statistical data sets (Bock, Diday, 1999). At the time of this writing, relatively few workers in the machine learning community have considered how classification rules might be genetically discovered from raw and expertly classified images. In this paper, a new data-driven approach is proposed in order to discover classification rules using the paradigm of genetic evolution.

The unique source of information is a remote sensing image and its corresponding classification furnished by an expert. The images have been registered by various satellites (e.g. SPOT, LANDSAT, DIAS, ROSIS) that use different cameras having various spectral and spatial resolutions (Weber, 1995). These types of remote sensing images generally contain huge volumes of data, for instance an image of DAIS contains 79 bands of each one 2.8 Mbytes. And, sometimes they are very noisy due to coarse spatial resolution or unfavorable atmospheric conditions at the time the images are acquired. In addition, data may be also erroneous due to inexperienced operators of the measurement devices.

P. Liardet et al. (Eds.): EA 2003, LNCS 2936, pp. 153–165, 2004.

The aim of this research is to detail an evolutionary classification method applied to remote sensing images. More about evolutionary classifiers can be found in (DeJong, 1988) and (Ross, Gualtieri et al., 2002). As stated, the approach to discover classifiers is data-driven because the formulated classification rules are generated from data and are able to adapt themselves according to this available data, environment, and the evolution of classes. In remote sensing, the initial population of classification rules is randomly created from raw images and given classes, and then evolved by a genetic algorithm until the acceptable classification accuracy is reached.

In remote sensing literature, several classification approaches are presented, namely:

- pixel-by-pixel, each image pixel is analyzed independently of the others according to its spectral characteristic (Fjørtoft, Marthon et al., 1996),
- zone-by-zone, before classification, the pixels are aggregated into zones, the algorithms detect the borders of the zones, delimit them by their texture, or their repetitive patterns (Kurita, Otsu, 1993),
- by object, this is the highest level of recognition, the algorithms classify semantic objects, detect their forms, geometrical properties, spatio-temporal relations using domain knowledge (Korczak, Louis, 1999).

Our approach uses spectral reflectances; therefore, discovered classification rules are only able to find spectral classes rather than semantic ones. This spectral component of class description is essential to well recognize thematic classes. The approach has been validated using our software environment, called *I See You* (ICU). In this software, the object representation is not too sophisticated but it offers a high degree of freedom in description of symbolic expressions of rules and definition of genetic operators. The goal was to evaluate the capacity of the genetic approach to handle problems of over-generalization and over-fit in highly noisy and complex data. The ICU is a genetic-based classifier, where we have adapted and extended ideas of learning classifier systems, such as XCS (DeJong, 1988; Wilson, 1999), the s-classifiers, and „Fuzzy To Classify System“ (Rendon, 1997). We have also been inspired by the works of Riolo (Riolo, 1988) on gratification and penalization, and of Wilson (Wilson, 1999) on the exploration of the search space.

The paper is structured as follows. The basic concepts of image classification rules are introduced in Section 2. Section 3 details the discovery process of the classification rules. In this Section, the behavior of genetic algorithm functions is explained. Finally, two case studies on real remote sensing data are presented in Section 4.

2 Concept of Classification Rule Extracted from Remote Sensing Images

In general, classification rules are symbolic expressions and describe conditions to be held and actions to be taken if the conditions are satisfied. It must be underlined that in our approach the rules are discovered by an evolutionary process and are not given a priori by a domain expert.

From a functional point of view, a rule represents a piece of knowledge about a class by a conditional expression, such as *if* <*conditions*> *then* <*class*>. The „*conditions*" part described an entry information in the system such as value, color, form, shape, etc, corresponding to conditions that must be fulfilled in order to activate this rule. The „*class*" part defines the class of the instance currently treated by the rule when the appropriate conditions were satisfied. We assert that the evolved rules must be rapidly evaluated and easy to interpret by any user. As a result, condition representation using the concept of an interval could be fully adequate for remote sensing image classification. In terms of machine learning, the rules have to be maximally discriminant generalizations, meaning that they have to cover the maximum pixels belonging to a given class and the minimum pixels belonging to another classes.

Before rule specification, recall that a pixel is encoded as a spectral vector, describing values of reflectance for the n bands of the remote sensing image, i.e. a pixel can be considered as a point in a R^n space :

$$<pixel> := [b_1\ b_2\ b_3 \ldots\ b_n] \tag{1}$$

In our system, the condition for any rule is built on the concept of spectral intervals defining a given band, corresponding to a given class. Such intervals are a pair of integer numbers, between 0 and the maximum possible value for a pixel of a given band (i.e. 65536 for pixels defined on 16 bits). This solution allows to partition the space of the spectral values in two ranges: the first containing the pixel values which corresponds to a given class, and the second containing the remainder.

To precisely specify the class definition, a set of intervals is defined for each band of the remote sensing image. Taking into consideration all bands, the condition part is defined as a set of hyper-rectangles in a R^n space :

$$<condition> := \bigwedge_{i=1}^{n} \bigvee_{j=1}^{k} (m_i^j \le b_i \le M_i^j) \tag{2}$$

where m_i^j and M_i^j are, respectively, the minimal and maximum reflectance values allowed for a pixel belonging to a class C for the band i. k is a parameter which defines the maximum number of disjunctions allowed.

These intervals are not necessarily disjunctive. By experiments, we have found that if we allow the genetic algorithm to create non-disjunctive intervals, instead of merging them, the results of genetic operators are more interesting. We have also noticed that merging intervals significantly diminishes the number of intervals, and in the same time, reduces the possibilities to create more efficient rules. To illustrates the concept of interval merging, $E = [11; 105] \vee [138; 209] \vee [93; 208]$ corresponds after merge operation to $E = [11; 209]$.

To satisfy a rule, a pixel has to match at least one spectral interval for each band. Logically speaking, to associate a pixel to a class, its values have to satisfy the conjunction of disjunctions of intervals that define a condition part of the classification rule.

This representation of the rule has been chosen mainly because of its simplicity, compactness and uniform encoding of spectral constraints. During experimentation, this representation has also demonstrated rapid execution of genetic operators and efficient computing. Of course, one may specify more complex structures using spatial properties of the pixel, with respect to the pixel neighborhood. Also, one may include features resulting from thematic indices or mathematical operators applied to pixel environment. These semantically extensions are interesting, however they not only require more sophisticated genetic operators, but also more powerful computers to perform the calculation in an acceptable amount of time.

3 From the Rule Creation to the Evolution

3.1 Genetic Algorithm

In order to efficiently develop the classification rules, a genetic algorithm initializes interval values according to spectral limits of the classes designated by an expert, for valid zones of the remote sensing image. Initial classification rules are created based on the extreme maximum and minimum values for defined spectral intervals of each class. It should be noted that by this initialization, rule searching is considerably reduced, and initial intervals are very close to the final solution. More about initialization algorithms can be found in (Kallel, Schoenauer, 1997). During the process of evolution, the initial spectral limits are slightly perturbed by adding a random value to lower and upper spectral limits. Hence, the initial population of classification rules is quite diversified.

A Michigan-like approach is used to discover independently a classification rule for each class. A major reason for choosing this approach is the efficiency of computations; that is, the process of rule discovery is not perturbed by other rules.

The quality of classification rules is based on a comparison of these results with the image classified by an expert. If pixels covered by the rule perfectly overlap those indicated by an expert, then the system assigns the highest quality value to the rule; otherwise, in the case of some mismatching, the quality factor is reduced (between 0 and 1). An associated fitness function will be detailed in the next section. During the evolution process, the rules are selected according to the quality for a given class. It should be noted that it is also possible to define global system quality based on rule classification qualities. The process of rule evolution is defined in the algorithm below.

As mentioned before, this algorithm must be designed to run independently for each class. This allows for obtaining rules according to user requirements without the necessity of carrying out computations for all classes with the same level of quality. This also allows to preserve the previously generated rules, as well as to introduce of new ones. Further, the user may define a hierarchy of classes and specialize some rules while respecting newly created sub-classes with different levels of classification quality.

Algorithm 1. Process of rule discovery.

R is a classification rule and P, P' and P'' populations of classification rules.

```
R: = INITIAL_RULE(images)        // Creation of rule according to spectral extremes
P: = INITIALIZATION(R)   // Random perturbation of rules
EVALUATION(P)                    // Calculation of the fitness function for each rule
do while TERMINATION_CRITERION(P) = false
 P' : = SELECTION_X(P)                  // Selection for crossover
 P' : = CROSSOVER(P') U COPY(P)
 P'' : = SELECTION_MUT(P')              // Selection for mutation
 P'' : = MUTATION(P'') U COPY(P')
 EVALUATION(P'')
 P: = REPLACEMENT(P,P'')                // New generation of rules
end_while
```

Result: A print of the classification rule R for a given class, statistics and quality measures for the discovered rule.

3.2 The Evaluation Function

The evaluation function serves to differentiate the quality of generated rules and guide genetic evolution. Usually, this function depends strongly on application domain. In our work, we define a pixel that the rule classifies as being in the class when the expert classifies the pixel as in the class as a true positive. Conversely, we define a pixel that the rule classifies as being not in the class when the expert classifies the pixel as not in the class as a true negative. Other pixels are said to be correctly classified.

We normally use as a quality measure the proportion of pixels that are correctly classified by the rule. In some cases, when classes are under- or over-represented, we care more about one misclassification than another. In these cases, we use $\alpha.p_{tp}+(1-\alpha).p_{tn}$ as a quality measure, where p_{tp} is the proportion of true positive pixel classifications by the rule (called *sensitivity*), p_{tn} is the corresponding proportion of true negatives (called *specificity*), and α is a parameter that lets us adjust the relative weight given to true positives and false negatives. By default the value of the coefficient α is fixed to ½.

The proposed function shows a number of advantages; it is independent of the pixel processing sequence, invariant of the size of classes, and efficient for class discovery with a highly variable number of pixels.

The evolution process converges according to some statistical criteria indicating if the current rule is near to a global optimum or if the population of rules will not evolve anymore. The termination criterion of the algorithm leans on the statistics of rule quality evolution. In our system, we take into consideration not only the evolution of quality of the best discovered rule, but also the minimum acceptable quality defined by a user, the process stability measure and a maximal number of generations to run. If one of these criteria is satisfied, then the process is stopped.

The most difficult question is whether the quality of a rule is not continuing to evolve. To detect stabilization of the quality evolution, instead of taking into account the best rule generated recently we have based our heuristics on statistics regarding quality evolution of the best discovered rules in a time. For example, let Q_k be the quality of the best rule obtained during the last k generation, and Q_o be the quality of the best rule of the current generation. Formally, the algorithm is stopped if the following equation is satisfied:

$$\left| \frac{\sum_{k=1}^{P} Q_k}{P} - Q_0 \right| \leq E \tag{3}$$

where P represents the maximum period of quality stabilization, and E is a maximal variation of this stabilization compared with the current quality.

It is important to have an initial population of rules within the vicinity of the solution to be found. We have proposed two algorithms allowing for the generation of a diversified pool of rules close to the expert hidden classification rule. The first, called MinMax, creates maximum intervals covering all the pixels belonging to a given class, and the second algorithm, called Spectro, integrates the spectral distribution density and interval partitioning.

With respect to software engineering, the genetic algorithm has been structured into layers corresponding to consecutive genetic operations (e.g. selection, mutation, crossover and replacement). This modular approach makes the program maintenance and future extensions much easier.

3.3 Genetic Operators

One of the most important tasks while designing a genetic algorithm is to invent operators that will create new potential solutions. All of our operators have been adapted to the rule representation, and they have been validated on remote sensing images.

Selection of classification rules. In general, selection is the operation of allocating reproductive opportunities to each rule. The reproductive force of a rule is expressed by a fitness function that measures relative quality of a rule by comparing it to other rules in the population. There are many methods for selecting a rule (Blickle, Thiele, 1995). In our system, the selection operator is applied in the following cases:

- choosing the rule to be reproduced for crossing, or muting;
- repetition of the rule, depending on whether it completes the genetic pool after having completed the crossover;
- preservation of a rule from the former genetic pool for the next generation;
- elimination of a rule in a newly created genetic pool based on an assigned rank.

Selection methods are well known: the roulette wheel, ranking, elitism, random selection, the so-called tournament, and eugenic selection. Our experiments have shown that roulette wheel selection is most advantageous for the reproductive phase, but the tournament strategy with elitism is best for the generational replacement scheme.

Crossover of rules. Crossover requires two rules, and cuts their chromosomes at some randomly chosen positions to produce two offspring. The two new rules inherit some rule conditions from each parent rules. A crossover operator is used in order to exploit the qualities of already generated classifiers. Each result of the crossover process has to be validated. Consistency of the various rule attributes (border limits violation, over-passing, etc) is carried out respecting the intervals boundaries. However, merging not only decreases the number of intervals in the rules, but also generates some information loss. In fact, in order to avoid a premature convergence of rules, it is generally important to preserve for the following generation two distinct intervals instead of a single aggregated one. On the other hand, it is also interesting to note that the positive or negative effects of an interval on the quality of the rule can be related to other intervals encoded in the classification rule.

Mutation of rules. The mutation operator plays a dual role in the system: it provides and maintains diversity in a population of rules, and it can work as a search operator in its own right. The mutation processes a single classification rule and it creates another rule with altered condition structure or variables. The mutation operator for several may be applied on three levels: band level, interval level and border level. Figure 1 shows the different variants of mutation as applied to remote sensing images.

Band mutation consists of a deletion of spectral bandwidth in a chosen classification rule. Its interest is twofold; firstly, the *band mutation* allows to simplify and generalize a rule; secondly, it allows to eliminate of noisy bands that frequently appear in hyper spectral images. The existence of noisy bands significantly perturbs the learning process, as well as the process of evolution convergence.

Interval mutation allows for a chosen band to add, eliminate or cut an interval in two spectral ranges. In case of addition, the new rule is completed by a new interval centered randomly with a user-defined spectral width. The cutting of an interval is done by random selection of a cutting point within the interval (for example, the cutting of *[10;100]* can generate two intervals: *[10;15]* and *[16;100]*). Interval mutation such as this allows splitting of continuous spectral ranges. And, this allows for the definition of a spectral tube in which spectral values of the pixels belong to a given class.

Finally, *border mutation* modifies both boundaries of an interval. This mutation refines the idea of targeting spectral tubes carried out by the other types of mutation. It is worthwhile to note that the mutated rules are systematically validated.

In our system, mutation operators are dynamically adapted. Adjustment is related to the probability of each mutation operator according to its current efficiency. Another schemes of mutation can be easily implemented, for instance self-adaptive mutations proposed by (Anglano et al., 1998; Thomsen, Krink, 2002).

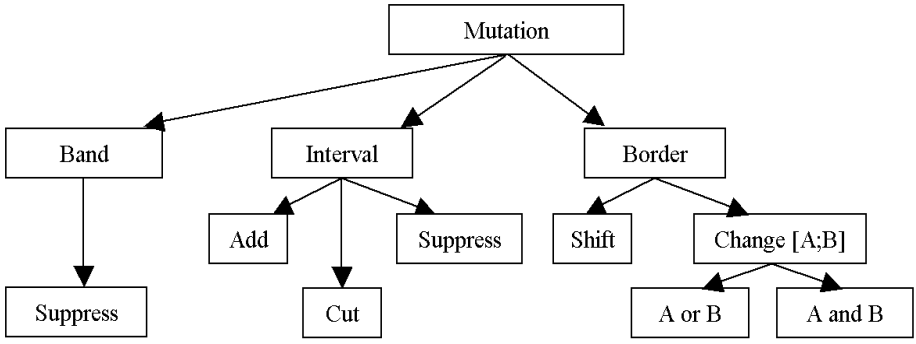


Fig. 1. Mutation operators

Generational replacement. The generational replacement is an operation that determines which of the classifiers in the current population is to be replaced by newly discovered children. According to Algorithm 1, the new generation of rules is created from a population of parents (P) and their children after the crossover and the mutation operations (P''). In our system, the following replacement strategies are applied:

- the revolutionary strategy in which only the population of the children completely replaces the parent population (P),
- the steady-state strategy in which new children are inserted in the new population by replacing the worst, or the oldest rule, or the most similar rules, or by preserving the best rules (elitism).

There exist other replacement strategies integrating, for instance, the strategy where the best rule of the previous population replaces the worst one of the current population or the strategy where the new classifiers having a performance higher than a certain threshold are inserted. However, both these strategies present the risk of having classifiers remain in the population, which is not necessarily a problem except in the case of a weak genetic pool in which some classifiers of average performances that would profit from immunity.

4 Case Studies and Experiments

In this paper, two case studies involving the remote sensing images of Strasbourg and San Felice (Lagoon of Venice) have been chosen. These cases contain hyperspectral data (DAIS 79 bands and ROSIS 80 bands, respectively), with 16 bits per pixel and 3m terrain resolution (Wooding, 2001; Quirin, 2002). The first case study considers a typical problem of classification for urban zones including a high percentage of mixed pixels. The second case demonstrates the performance of rules on very noisy images with closed spectral classes (mostly vegetation classes). Learning was carried out on the lower half of the image (932*184 pixels), and then validation was performed on the whole image (932*368 pixels).

To well understand the formalism of rule representation, let B_i be the reflectance value for the band i of the considered pixel. For instance, the conditional portion of the rule that classifies the instances of a *Limonium Narbonense* class is given below:

$$
\begin{aligned}
&(0 \leq \mathring{a}_0 \leq 65535) \\
\wedge \quad &(461 \leq \mathring{a}_1 \leq 1928) \\
\wedge \quad &\ldots \\
\wedge \quad &(0 \leq B_5 \leq 65535) \\
\wedge \quad &\ldots \\
\wedge \quad &\\
&((522 \leq B_{12} \leq 1895) \vee (6541 \leq B_{12} \leq 39307)) \\
\wedge \quad &\ldots \\
\wedge \quad &(364 \leq B_{79} \leq 2107)
\end{aligned}
$$

It is easy to notice that the band B_0 and B_5 are too noisy (range maximum), and they can be eliminated from the condition. The rules simplification may be also implemented indirectly in the rule evaluation function, promoting the rule simplicity. It should be also noted that after preliminary tests, this method generated many over-generalized rules with relatively weaker performance than that obtained by the simple deletion.

In the following part, the main results of evolutionary data mining are described. Each case study is illustrated by a classified image using the discovered rules, discussions and performance measures.

Image of Strasbourg, Stadium Vauban (Hyperspectral Image, 80 Bands)

Fig. 2. Classified image

Table 1. Parameters of the GA

Parameter	Value
Population	1500 rules
Generations	250
Stabilization length	20
Stabilization error	10^{-4}
Crossover rate	80%
Mutation rate	5%
Rate of eugenic sel.	1%
CPU time (P4 2.5GHz)	16 h
Learned classes	11
Performance	86,06 %

Comments. This complex remote sensing image contains more than 50% noisy bands. Moreover, *Water* data is represented by many small, corrugated lines (signal

has been scrambled by atmospheric conditions) so *Water* and *Shadow* spectral signals are very similar. Therefore, a small spot of *Water* appears in the middle of the city, instead of *Shadow*. The average quality of the best rule for each class is about 86%, which is relatively good performance.

Image of Venice, Lagoon San Felice (Multispectral Image, 4 Bands)

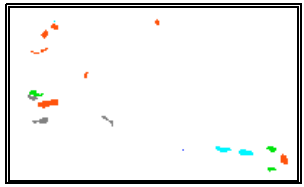


Fig. 3. Expert image

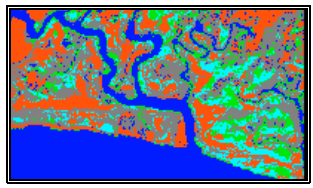


Fig. 4. Classified image

Table 2. Parameters of the GA

Parameter	Value
Population	500 rules
Generations	500
Stabilization length	10
Stabilization error	10^{-4}
Crossover rate	75%
Mutation rate	15%
Rate of eugenic sel.	1%
CPU time (P4 2.5GHz)	2 h
Learned classes	5
Performance	89,03 %

Comments. Fig. 3 presents a typical classified image by an expert using ground truth data. As illustrated the number of classified pixels (shown in color zones) is very low (1.43%). Note that in the whole image only 10 pixels were identified as the *Water* class. In spite of the small training set and the large space of search (spectral value of pixels is represented by 32 bits), the discovered set of rules is able to produce a coherent classified image.

Resistance to the noise of the learning process. The resistance of the classifier system to the noise has been evaluated numerically. Fig. 5 illustrates the protocol used for validation. ($Data_L$,$Expert_L$) are the training data and ($Data_T$,Exp_T) are the validation data. These sets are of the same size and are generated by taking for each one the half of the pixels of the whole image. *Noise(image,rate)* is a function which perturbs *rate%* pixels. *Valid()* is a function which generates the weighted performance of classifiers, according to the discovered rules (*Rules*) and an expert classification (*Expert*) of the remote sensing image (*Data*). *Cycle(rd,re)* computes the performance of the rules learned from a *rd%* perturbed data and a *re%* perturbed expert image. The curves below, C_{data}=*Cycle([0;50],0) and* C_{expert}=*Cycle(0,[0;50])* illustrate the weighted performance of the rules on the noisy training set.

We performed 11 runs for each test (perturbation of the data or the expert image). Standard deviations of the rule performances are : σ_{Cdata} = 0,042 and $\sigma_{Cexpert}$ = 0,026.

The two case studies have demonstrated the high capacity of the evolution-based rules to interpret and classify heterogeneous and complex images (e.g. high dimension, large number of bands and noisy data that provide a computational complexity of

```
Cycle(nd,ne)
  NoRules=50
  NoGenerations=50
  Sampling of data=10%
  Rules=Learning(
   Noise(Data_L , rd),
   Noise(Expert_L , re))
  Perf=Valid(Data_T,Expert_T,Rules)
  return Perf
EndCycle
```

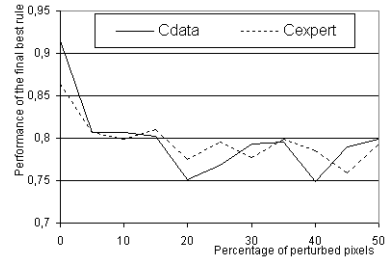


Fig. 5. The validation protocol and the resulting graph

$O(n^3)$, which is quite heavy for a deterministic algorithm). The quality of classification is very high even if there were a high number of noisy bands and mixed pixels. It must be noted that the quality of learning is highly related to the quality of the classified image used for rule discovery. The discovered classification rules are simple and easy to interpret by remote sensing experts. They are also mutually exclusive and maximally specific. The learning time was relatively long due to the large image size and the chosen parameters for the evolution process, but the computing time optimization was not addressed in these experiments.

During the experiments it was observed that the best rules use 0% mutation of bands, 5% mutation of intervals, 41% mutation of borders, and 53% crossovers. In spite of weak mutation rate (5-15%), mutation operators have demonstrated high efficacy. The diagram shows that this evolutionary process is able to admit nearly 5% of noise on the data or the expert image without significant loss of quality. We have observed that by adding more than 5% of noise, the rule quality does not clearly decrease. Rule generalization quality has also been evaluated and it is worthwhile to mention that the best set of rules on high-resolution images can be applied on a 23-times larger image with a loss of quality less than 0.02%.

Finally, high correlation was observed between obtained results and statistics carried out on the remote sensing image (spectrogram statistics, excluding noisy bands). Classified images by the discovered rules have shown that the evolution-based process is able to faithfully reproduce the human expertise.

5 Conclusions and Perspectives

This article has described an evolution-based method applied to remote sensing images. The system has discovered a set of *if … then* image classification rules using the fitness function based on class recognition quality. These rules, which were proven robust and simple to understand for the user, improve the accuracy of classifications proposed by the expert, and are sufficiently generic for reusing them on other portions of remote sensing images.

Taking into consideration image complexity and noisy data, the results of our experiments are very encouraging. Case studies have demonstrated that the obtained rules are able to reproduce faithfully the terrain reality.

The rules are well adapted to recognize large objects on the image (e.g. sport lands), as well as the smaller ones (e.g. trees, shadows, edges of the buildings). The redundant or noisy bands have been successfully identified by our rule representation. The formulation of rule representation has allowed for the modeling of a spectral tube adapted to the granularity of spectral reflectance. The proposed rules initialization seems to be well suited to large volume of data. It has considerably reduced the search space by generating initial rules close to the final solution.

The genetic system developed in this research work, called ICU, is currently available on our web site `http://lsiit.u-strasbg.fr/afd`. A new version of ICU is under development, including a more powerful representation of rules including spatial knowledge, temporal relations, hierarchical representation of objects, and new genetic operators.

Acknowledgements. The authors are grateful to Pierre Gançarski, Cédric Wemmert, Christiane Weber, Anne Puissant, Jélila Labed, Massimo Menenti and Jennifer Ayers for their advice, suggestions and comments during the early stages of this research. We also appreciated the help brought by the suggestions of four anonymous reviewers.

References

ANGLANO C., GIORDANA A., BELLO G. L., SAITTA L. (1998). An experimental evaluation of coevolutive concept learning, [in] *Proc. of ICML'98*, Morgan Kaufmann, San Francisco, pp. 19-27.

BLICKLE T., THIELE L. (1995). A Comparison of Selection Schemes used in Genetic Algorithms, Computer Engineering and Communication Networks Lab, *TIK-Report Nr. 11*, Second Edition, Swiss Federal Institute of Technology, Zurich.

BOCK H. H., DIDAY E. (eds.) (1999). Analysis of Symbolic Data. Exploratory Methods for Extracting Statistical Information from Complex Data, [in] *Studies in Classification, Data Analysis and Knowledge Organization*, vol. 15, Springer-Verlag, Heidelberg.

CARVALHO D. R., FREITAS A. A. (2002). A genetic algorithm with sequential niching for discovering small-disjunct rules, [in] *Proc. of GECCO-2002*, pp. 1035-1042.

DEJONG K. A. (1988). Learning with Genetic Algorithms: An Overview, *Machine Learning*, vol. 3, pp. 121-138.

FJØRTOFT R., MARTHON P., LOPES A., SERY F., DUCROT-GAMBART D., CUBERO-CASTAN E. (1996). Region-Based Enhancement and Analysis of SAR Images, [in] *Proc. of ICIP'96*, vol. 3, Lausanne, pp. 879-882.

KALLEL L., SCHOENAUER M. (1997), Alternative random initialization in genetic algorithms, [in] *Proc. of ICGA'97*, Morgan Kaufmann, San Fransisco, pp. 268-275.

KORCZAK J., LOUIS N. (1999). Synthesis of Conceptual Hierarchies Applied to Remote Sensing, [in] *Proc. of SPIE*, Image and Signal Processing for Remote Sensing IV, Barcelona, pp. 397-406.

KORCZAK J., QUIRIN A. (2003). Evolutionary Approach to Discovery of Classification Rules from Remote Sensing Images, [in] *Proc. of EvoIASP2003*, Essex.

KORCZAK J., QUIRIN A. (2003). Découverte de règles de classification par approche évolutive : application aux images de télédétection, [in] *Proc. of EGC2003*, Lyon.

KURITA T., OTSU N. (1993). Texture Classification by Higher Order Local Autocorrelation Features, [in] *Proc. of Asian Conf. on Computer Vision*, Osaka, pp. 175-178.

QUIRIN A. (2002). Découverte de règles de classification : classifieurs évolutifs, *Mémoire DEA d'Informatique*, Université Louis Pasteur, LSIIT UMR-7005 CNRS, Strasbourg.

RENDON M. V. (1997). Reinforcement Learning in the Fuzzy Classifier System, *Reporte de Investigaci No. CIA-RI-031*, ITESM, Campus Monterrey, Centro de Inteligencia Artificial.

RIOLO R. L. (1988). Empirical Studies of Default Hierarchies and Sequences of Rules in Learning Classifier Systems, PhD Dissertation, Comp. Sc. and Eng. Dept, Univ. of Michigan.

ROSS B. J., GUALTIERI A. G., FUETEN F., BUDKEWITSCH P. (2002). Hyperspectral Image Analysis Using Genetic Programming, [in] *Proc. of the Genetic and Evolutionary Computation Conference (GECCO 2002)*, ed. W. B. LANGDON et al., CA: M. KAUFMANN, pp. 1196-1203.

THOMSEN R., KRINK T. (2002). Self-Adaptive Operator Scheduling using the Religion-Based EA, [in] *Proc. of Parallel Problem Solving from Nature VII (PPSN-2002)*, pp. 214-223.

WEBER C. (1995). Images satellitaires et milieu urbain, Hermès, Paris.

WILSON S. W. (1999). State of XCS Classifier System Research, [in] *Proc. of IWLCS-99*, Orlando.

WOODING M. (2001). *Proceedings of the Final Results Workshop on DAISEX (Digital AIrborne Spectrometer EXperiment)*, ESTEC, Noordwijk.

Ant Algorithm for Detection of Retentive Structures in Coastal Waters

Marc Segond[1], Sébastien Mahler[2], Denis Robilliard[1], Cyril Fonlupt[1], Benjamin Planque[3], and Pascal Lazure[3]

[1] Université du Littoral-Côte d'Opale, France
[2] École des Hautes Études Industrielles, France
[3] IFREMER, France

Abstract. Demography of the anchovy in the Gulf of Biscay is supposed to be linked with the presence of retentive vortices structures, that could retain eggs and larvae in particular environmental conditions. An automatic process for identifying such structures in massive datasets is needed to corroborate this hypothesis with biology statistics. The data we are working on is a large set of vector fields issuing from ten years of daily hydro-dynamical simulations of the stream circulation. Classical vortices detection methods on vector fields do not focus specifically on retentive structures, thus we propose an adaptation of the ant algorithm paradigm for this identification task. We show that this scheme easily achieves retrieval of significant structures.

1 Introduction

1.1 Retentive Structures and Fish Species

The presence and density of animal species in the ocean and coastal waters are often conditioned by the presence of physical structures, such as upwellings, temperature fronts, or vortices. In the case of the anchovy in the Gulf of Biscay, biologists from the Ifremer institute want to investigate the relationship between the presence of such structures and fish demography. Focus is put on so-called retentive structures, that could retain eggs and larvae in specific environmental conditions. Meso-scale vortices-like structures, whose size ranges from tens to hundreds of kilometers, are supposed to be retentive structures of main importance. To verify this hypothesis, one difficulty is efficiently identifying such patterns in massive datasets, in order to match their presence against biologists observations and fisheries statistics. We first have a look at common methods for vortices detection in the next subsection.

1.2 Classical Methods for Vector Field Analysis

The data we study here is a large set of vector fields issuing from ten years of daily hydrodynamic simulations of the stream in the Gulf of Biscay. We do not detail the 3D hydro-dynamical model that has been developed by Pascal

P. Liardet et al. (Eds.): EA 2003, LNCS 2936, pp. 166–176, 2004.

Lazure at Ifremer laboratory, we just note that stream direction and speed are computed for any day of the year on a discrete grid with cell size 10 kilometers in the horizontal plane, and 10 meters in the vertical/depth dimension, then horizontal 2D slices are extracted at 10 meters depth and constitute the data we are working on. These 2D vector fields are not conservative with respect to the stream flow (there are flow sinks and sources), nonetheless they contain enough information for biologists to point out structures that could be retentive for biological agents. Figure 1 illustrates a vector field.

Dealing with vortices in vector fields is a well known matter, that has been investigated notably for purpose of simulation and also for scientific visualization. A classical method is computing the vorticity (or *curl*) in every cell, which can be done on a discrete field by a convolution. Then one can extract the extrema, or use a threshold on this scalar field, as is explained for example by Corpetti et *al.* in [1]. The same authors have proposed in [2] a more precise method based on the Helmholtz decomposition of a vector field $\omega(x, y) = (u(x, y), v(x, y))$ in its solenoïdal part ω_{so} and irrotational part ω_{ir}:

$$\omega = \omega_{so} + \omega_{ir} \, .$$

From the knowledge of $\omega_{so} = (u_{so}(x, y), v_{so}(x, y))$ one can build the orthogonal field $\omega_{so}^{\perp} = (-v_{so}(x, y), u_{so}(x, y))$, and then integrate a scalar potential function ψ such that:

$$\omega_{so}^{\perp} = \nabla\psi \, .$$

The extrema of ψ are characteristic points of the solenoïdal field, and one can extract the vortices centers. The vortices spatial extensions can then be modeled in several possible ways. The interested reader may also refer to [3] for a good review of the state of the art.

Although theoretically sound, these classical schemes do not comply well with the problem at hand. The first method (thresholds of vorticity) may be useful for large scale structures in open ocean, but the shallow depths of coastal waters induce many deviations of the stream circulation, resulting in much "noise" and inaccurate structures shapes. This is illustrated in Fig 2 where the threshold is not low enough to cover the totality of the main structures (upper left and center of the picture), but already too high to avoid the apparition of noise along the coast (upper side of the picture); notice also the erroneous detection of a curling but not retentive structure in the lower right corner. The second technique is excellent at finding vortices centers but no satisfying model is proposed for determining the vortices extensions: the use of Rankine functions only defines unrealistic circular vortices. Moreover we can not assume that the detected vortices always coincide with retentive structures, because the detection is based only on the solenoidal component of the field, thus ignoring a possible irrotational (i.e. radial) movement of the stream that may quickly push eggs and larvae outside of the structure. Indeed the identification of retentive structures needs to take into account the whole information contained in the stream vector field. To tackle this problem, we propose to give a try at a totally different paradigm: ant algorithms.

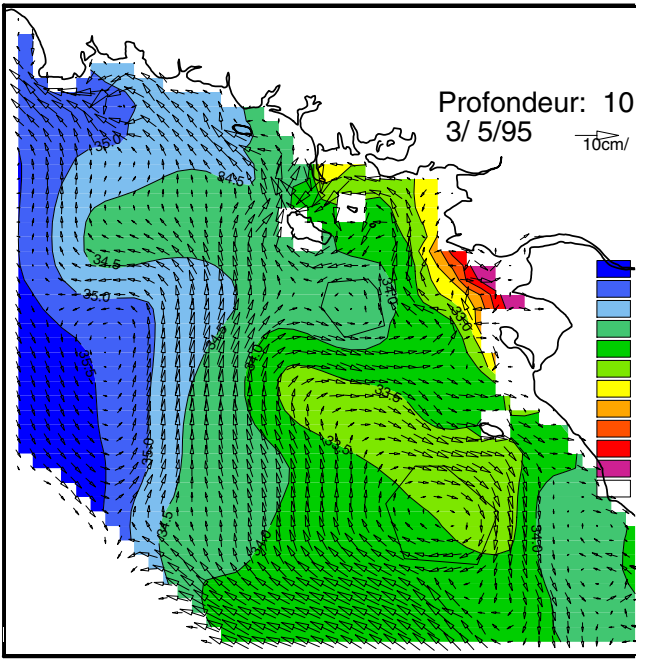


Fig. 1. The circulation in the northern bay of Biscay simulated by a 3D hydrodynamical model at 10m depth. Levels of grey show the salinity field, arrows indicate the speed and direction of the stream, and black lines polygons are suspected retentive structures, tagged by an expert

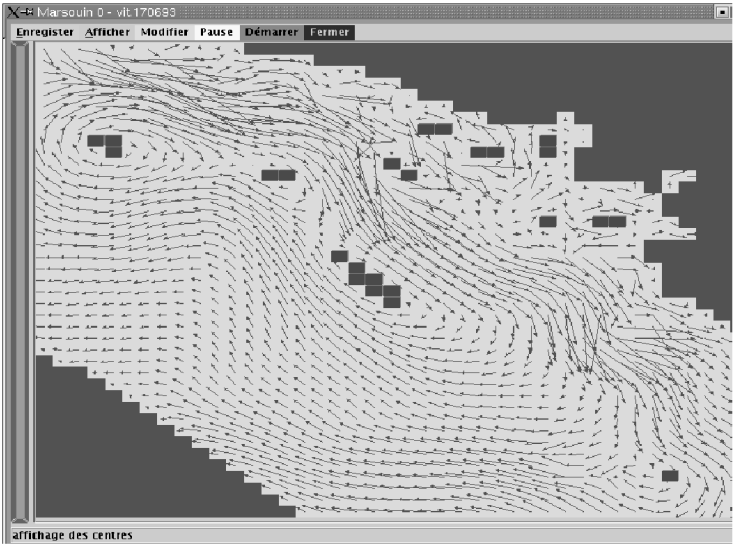


Fig. 2. Detection of vortices by a curl threshold method. Some irrelevant structures are detected, for example in the lower right corner, or along the upper coast-line

1.3 Ant Algorithms

Behavior of social insects like ants, bees, termites and some wasps is characterized by self-organization. This means that interaction is based on local information without explicit references to any global goal. Individuals usually communicate by changing local properties of their environment, and through this limited medium of communication some kind of distributed and collective intelligence emerges [4,5].

Real ants have inspired the "ant algorithms", that were introduced with Marco Dorigo's PhD. The basic idea is to imitate the cooperative behavior of an ant colony in order to solve large combinatorial optimization problems [6,7,8,9]. Primary works have been based on simple model of the ant foraging mechanism, that allows an ant group to find the shortest path between its nest and some food source. During their trip, ants leave a chemical trail of *pheromone* on the ground, whose role is to guide the other ants towards the target point. This chemical substance evaporates and thus has a decreasing action over time, and the quantity left by one ant depends on the amount of food it found. Every ant chooses its path partly at random and partly according to the quantity of pheromone it smells.

This model of communicating ants has been used as a framework for solving combinatorial optimization problems like the TSP [7,10,11], routing problems, load balancing in communication networks [12], numerical optimization [13,14], graph coloring problem [15],... New trends based on the behavior of social insects have also appeared these last years (modeling of ants chemical recognition system, ...) and have led to new topics of research: classification [16], automatic music generation and automatic painting[1] for instance.

In this paper, we explain how to transpose this mechanism to the problem of detecting retentive structures in coastal and oceanic waters. We show that our Marsouin algorithm (French name for porpoise) easily achieves automatic retrieval of retentive vortices, and that the structures' shape is better detected with this technique than with other methods.

The remainder of this paper is organized in the following way. Section 2 describes the main principles used by our ant algorithm and how we adapt the ants paradigm to automatic retentive structure detection. Section 3 describes the experimental results that have been obtained. Section 4 draws some conclusions and discusses about further works.

2 Marsouin: An Ant Algorithm for Retentive Structure Detection

In ant algorithms, new solutions are usually created by virtual ants using some quality function of the local environment, e.g. length of edges for a graph shortest path problem, and some simulation of pheromone communication, e.g. a matrix

[1] `http://www.antsearch.univ-tours.fr/WebRTIC/default.asp?FCT=DP&ID_PAGE=52`

of pheromone concentrations. A typical basic ant algorithm can be summed up as a loop over three main steps:

- generation of solutions by ants according to local and pheromone information
- application of a local search scheme to improve the solutions found (although not inspired from the biological model, this phase is necessary to obtain competitive results on many combinatorial search problems);
- update of the pheromone information

2.1 Ants Population and Behavior

Real ants live in colonies and are able to distinguish both their own trail and that of their colony fellows. In our case, we consider a population of ants divided into several colonies, called *teams*, sharing a specific pheromone that does not attract ants from other teams. This is done in order to help the teams to focus on different vortices and was successfully tested with population size from as small as 10 up to 500, and 1, 5 or 10 teams. These virtual ants are randomly spawned on the cells of a 2-dimensional lattice. Each cell is characterized by speed and direction of the stream, and it also has two arrays of pheromone concentrations, one indexed by ants and the other indexed by teams. This allows an ant to detect its own trail and the composite trail of its own team. Pheromone concentrations are initialized to null value.

Ants iteratively move from one cell to one of its neighbors under an 8-connectivity topology, as depicted in Fig. 3.

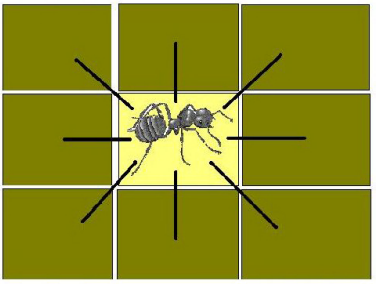


Fig. 3. Ant scheme of cell connectivity

These moves are synchronized and thus all ants move at the same rate. The choice of the next position is determined according by a stochastic rule modulated by four parameters:

1. direction of the stream is the main parameter in the choice of the next move: ants are allowed a maximum 45 degrees deviation from the direction of the stream

2. velocity of the flow: the higher the speed of the stream, the higher the chance to follow the direction of the current
3. level of pheromone dropped by other ants of the same team
4. an individual random directional bias, that is drawn with equal probability at the creation of every ant

A fitness parameter is calculated for each neighbor cells. The destination cell is chosen by performing a "roulette wheel" between each candidate cell. The formula below describes the way this fitness value is calculated for each cell:

$$f(i,\theta,\beta) = 0 \text{ if } \min\left(\left|\left(\theta+\beta.\frac{\pi}{4}\right)-\alpha_i\right|, \left|\left(\theta+\beta.\frac{\pi}{4}\right)-(2\pi-\alpha_i)\right|\right) > \frac{\pi}{4} \quad (1)$$

$$= \frac{1}{3}\left(-\frac{4\left(\min\left(\left|\left(\theta+\beta.\frac{\pi}{4}\right)-\alpha_i\right|, \left|\left(\theta+\beta.\frac{\pi}{4}\right)-(2\pi-\alpha_i)\right|\right)-\frac{\pi}{4}\right)}{\pi} + \frac{\mu_i}{\mu_{max}} + \frac{\phi_i}{\phi_{max}}\right) \text{ else,} \quad (2)$$

with θ ant's direction angle, α_i angle of the candidate cell number i, β ant's bias, μ_i speed in candidate cell number i, μ_{max} the maximum speed in the whole matrix, ϕ_i pheromone concentration in candidate cell number i and ϕ_{max} the maximum pheromone concentration in the whole matrix.

As stream vectors are always tangential to any rotating circulation of the stream, the individual bias has been found useful to allow ants to follow curved trajectories. Notice that this simple set of rules takes into account the whole information of the field: in the presence of both rotating and diverging flow, ants may be efficiently prevented from achieving a loop around a vorticity extremum.

The general course of the algorithm is changed on three occurrences:

- when an ant reaches the coast, it is picked up and dropped elsewhere
- when an ant loops over its own individual trail, the looping part of the trail is examined as a candidate vortex (see next subsections)
- when a given number of moves has been performed the algorithm stops.

2.2 Local Search

Almost all ant algorithms are hybridized with a local search method in order to enhance the search. In our case, there is no true local search scheme, but a simple filter is applied when a candidate vortex has been found: looping paths that are too short to be compatible with that of a meso-scale structure (less than 40 kilometers), and "forward-backward" flat loops, that may happen in places where the stream is chaotic, are discarded.

2.3 Pheromone Update

Many different pheromone updates schemes exist and have been proposed in the literature [9,7,17,18]. They usually consist in two parts: evaporation and

amplification. During the evaporation phase, all pheromone trails are decreased according to a given value (percentage, probability...) while during the amplification phase the best pheromone trails are increased, in analogy with real ants increasing their deposits after finding a food source.

As a candidate vortex can be found at any time step, evaporation in Marsouin is done after every move phase, and not as a dedicated step performed after having built complete solutions as in many other ant algorithms. The evaporation factor ρ is an input parameter, and experiments show that values between 0.1 and 0.5 provide interesting results for the set of maps we used. For every cell (i, j) the following evaporation formula is applied on all pheromone values τ:

$$\tau_{i,j}^{\text{new}} = (1 - \rho)\tau_{i,j}^{\text{old}} .$$

Standard ways of performing amplification can be roughly split up into two sets: either only the "best" ant can improve its trail; or all ants improve their trails in proportion to their current fitness. The intuitive idea is that amplification should take place when a given task has been fulfilled, e.g. completing a circuit through all cities in the classic Traveling Salesperson Problem. Here, our implementation is very close to the natural model: each time an ant enters a cell, it drops some pheromone. The reinforcement effect is computed according to a classical update scheme:

$$\tau_{ij}^{\text{new}} = \tau_{ij}^{\text{old}} + Cst$$

where Cst is a constant fixed by the user (usually above 10.0).

A more involved amplification scheme has also been tried, consisting in an extra amplification of pheromone dropped on looping trails in order to model increased deposits after finding a goal, but it does not bring noticeable improvement over this simple method, although a definitive conclusion is differed until the end of the evaluation phase (see Sect. 3).

2.4 Final Step

Once the ant algorithm is over, we perform a final analysis of the candidate structures:

- structure barycenters are computed
- when the intersection of several structures contains all their barycenters, these structures are merged into a larger one, and a new barycenter is computed.

The largest structure is detected (we mean the structure which has the biggest area) and all the other structures that are totally or partially included in the former are merged together to obtain a unique largest "envelope" of the retentive vortex structure, then the coordinates of the barycenter, the direction of rotation and the area are computed and displayed.

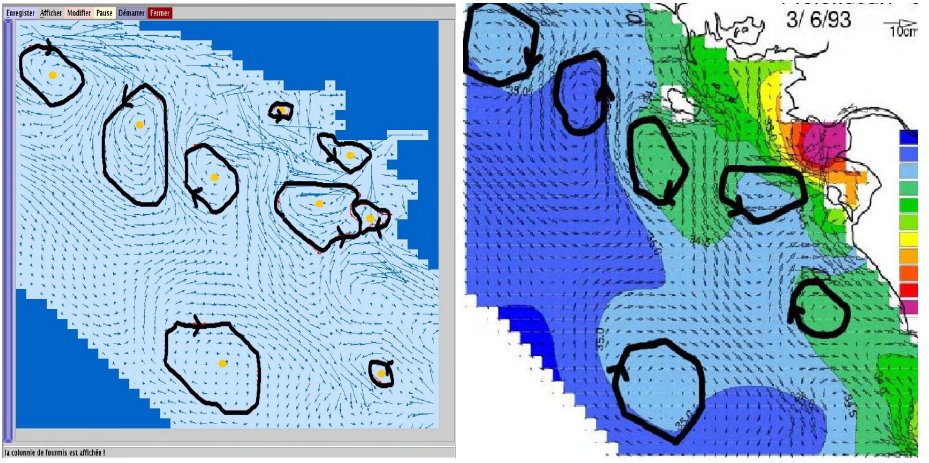


Fig. 4. Results of the *Marsouin* application in the bay of Biscay. Left graph presents vortices found by the *Marsouin* application using 10 teams of 10 ants, while right graph shows expert tagged structures

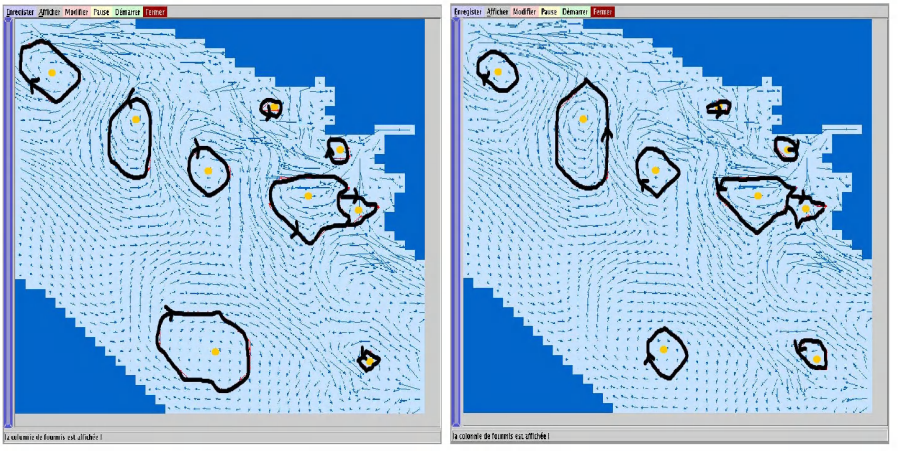


Fig. 5. *Marsouin* detected structures on the same scene, left with 1 team of 20 ants, right with 4 teams of 5 ants

3 Experiments

In this section we present some structure detections performed on a set of maps generated by the Ifremer laboratory. These are to be considered as illustrations, until the results of a quantitative statistical validation phase that is expected to end in fall 2003. This set consists of several "snapshots" of the Gulf of Biscay during the year 1993, the maps varying in size and location. The running time is around 1 minute for Marsouin to analyze a 200×200km map, while vorticity

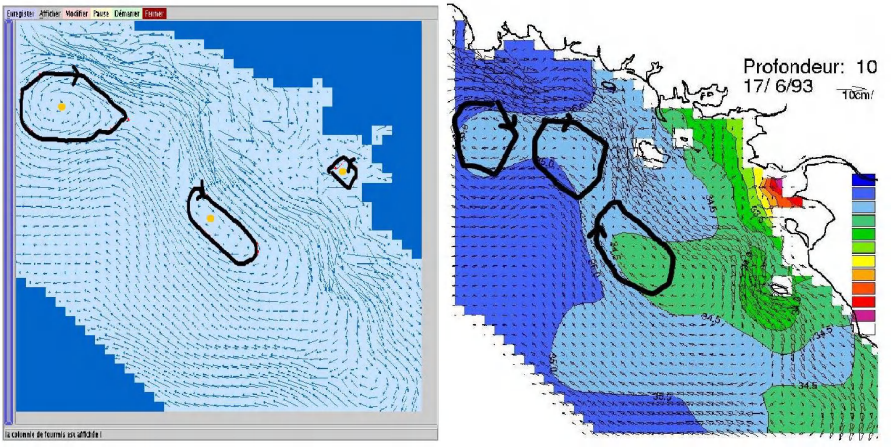


Fig. 6. Detected structures on another scene

based method may use from a few seconds up to 2 minutes (see [3]), depending on the specific method and on the vortex model.

Figure 4 allows a comparison between an automatic detection against structures retained by an expert. Most structures are detected, some small new patterns are found, and one structure proposed by the expert is not chosen by Marsouin (on the right of the picture). Nonetheless for this missing structure we notice that divergence is quite high in the area, preventing ants to loop on. This could be tackled by allowing stronger individual bias to ants, so to render them able to loop around such structures, but through discussion it appears that it might well be the case that the expert has been too optimistic in selecting that particular zone, so the conclusion is delayed until the final word comes from fisheries data.

Figure 5 is a comparison of the same scene with two other population settings, thus illustrating the robustness of the system to parameter changes. The main difference is that with fewer ants, the structures tend to be smaller, and this can be expected as fewer ants provide less exploitation of promising zones.

Figure 6 is an illustration of the search on another map compared to expert judgment. On this scene the central structure is never detected in ten runs, even when changing the number of ants and teams. However, discussions with the expert and a closer inspection of the scene led to the conclusion that this structure should be merged to the one to its left, rather than being considered distinct. However, our algorithm does not yet select the whole extension of the structure.

Marsouin is not available as a free-ware at the moment, but the interested reader might have a look at our web-site[2] for color pictures.

[2] `http://www-lil.univ-littoral/~robillia/Research/Marsouin`

4 Conclusion

Pattern detection tasks occur in many fields related to engineering. New artificial intelligence techniques may be of help when such tasks are not well defined in mathematical or physical terms. In this study, the focus is set on structures with a property of biological retention, and this can not be straightly confounded with the notion of physical vortices as it is used in fluid dynamics. As the problem is defined in terms of retention of biological agents, we experiment the possibility of using convergence of trajectories of virtual ant agents to perform this task.

The model chosen is perhaps closer to its natural counterpart than many ant algorithms, most of them being dedicated to combinatorial optimization problems. The population of ants is split into several colonies, and any particular ant is able to retrieve its own pheromone trail as well as the composite trail left by the whole colony it belongs to. Once an ant has found a looping trajectory, it is interesting to explore around it to try to extend it if possible, in order to know the shape and the dimension of the structure. The pheromone deposit provides the incitement needed by the colony to perform this "exploitation" task. The pheromone update formulas have been kept as simple as possible, and the whole algorithm was very easy to specify and to implement.

This approach is quite successful, and results are well enough in accordance with expert expectations. Experiments showed that reducing the number of ants per team tends to reduce the detected size of structures. On the other hand increasing the number of teams was not necessary on the scenes we worked on. On some pictures the automatic detection led the expert to change its mind, and this was expected because eye strain is a big problem for a human processing several maps. A validation phase is under course to quantify the performance of the algorithm, its sensitivity to parameter changes on a large set of runs and data scenes, and the possible error reduction it can bring versus a human expert in limited time conditions.

This is of course not the only solution that could be tried for this problem. For instance another path could consist in defining a formal model of vortex including a potential function dedicated to take into account the divergence component of the stream. This model should be kept flexible enough to map vortices of variable shape and size, while being moderate on computational resources.

References

1. T. Corpetti, E. Mémin, and P. Pérez. Dense estimation of fluid flows. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(3):365–380, March 2002.
2. T. Corpetti, E. Mémin, and P. Pérez. Extraction of singular points from dense motion fields: an analytic approach. *Journal of Mathematical Imaging and Vision*, to appear in 2003.
3. Thomas Corpetti. *Estimation de l'analyse de champs denses de vitesses d'écoulements fluides.* PhD thesis, Université de Rennes 1, France, 2002. in French.

4. Eric Bonabeau, Marco Dorigo, and Guy Theraulaz. *Swarm Intelligence: From Natural to Artificial Systems*. Oxford University Press, 1999.
5. David Corne, Marco Dorigo, and Fred Glover, editors. *New Ideas in Optimisation*. Mc Graw-Hill, 1999.
6. M. Dorigo, V. Maniezzo, and A. Colorni. Positive feedback as a search strategy. Technical Report 91-016, Politecnico di Milano, Milano, Italy, 1991.
7. M. Dorigo, V. Maniezzo, and A. Colorni. The ant system: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Mans, and Cybernetics*, 1(26):29–41, 1996.
8. M. Dorigo and L. Gambardella. Ant colonies for the traveling salesman problem. Technical Report TR/IRIDIA/1996-3, IRIDIA, Université Libre de Bruxelles, Belgium, 1996.
9. M. Dorigo and L. Gambardella. Ant colony system: A cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computation*, 1(1):53–66, 1997.
10. L. Gambardella and M. Dorigo. ANT-Q: A reinforcement learning approach to the traveling salesman problem. In *Proceedings of the twelth International Conference on Machine Learning*, pages 252–260, 1995.
11. M. Dorigo and L. Gambardella. A study of some properties of ant-q. In *International Conference on Evolutionary Computation: the 4th Conference on Parallel Problem Solving from Nature*, pages 656–665, 1996.
12. G. Di Caro and M. Dorigo. Antnet: A mobile agents approach to adaptive routing. Technical Report IRIDIA/97-12, IRIDIA, Université Libre de Bruxelles, Belgium, 1997.
13. G. Bilchev and I. Parmee. The ant colony metaphor for searching continuous design spaces. *Lecture Notes in Computer Science*, (993):24–39, 1995.
14. Nicolas Monmarché, Mohand Slimane, and Gilels Venturini. on how *Pachycondyla apicalis* ans suggest a new search algorithm. *Future Generation Computer Systems*, 16(8):937–946, 2001.
15. D. Costa and A. Hertz. Ants can color graphs. *Journal of Operation Research Society*, (48):295–305, 1997.
16. Nicolas Monmarché, Mohand Slimane, and Gilles Venturini. L'algorithme antclass : classification non supervisée par une colonie de fourmis artificielles. *Extraction des Connaissances et Apprentissage : Apprentisage et évolution*, 1(3):131–166, 2001.
17. T. Stutzle and H. Hoos. the MAX-MIN ant system and local search for the traveling salesman problem. In *International Conference on Evolutionary Computation*, pages 308–313, 1997.
18. T. Stutzle and H. Hoos. Improvements on the ant system: Introducing the MAX-MIN ant system. In *Proceedings of the third International Conference on Artificial Neural Networks and Genetic Algorithms*, 1997.

Air Traffic Controller Keyboard Optimization by Artificial Evolution

Daniel Delahaye[1] and Stephane Puechmorel[2]

[1] CENA ⋆⋆⋆
[2] ENAC†
delahaye@tls.cena.fr
puechmor@recherche.enac.fr
CENA-ENAC
7, Avenue Edouard Belin
31055 Toulouse, France

Abstract. The annual number of daily flights in France has increased from about 3500 in 1982 to about in 8000 in 2000. The number of flights simultaneously present on the radar screen of the controller has also increased. Usually controllers manage about 15 aircraft on their position and sometime this number reach a maximum of 20. On the radar screen, aircraft are represented by spots (with some previous positions and their speed vector) and the associated label which give the flight ID, the speed and the altitude of the aircraft. The controller in charge of the controlled area, has to be able to select any aircraft in order to manipulate some parameters of the flight such as heading, speed, altitude etc. Aircraft selection is done by the mean of a virtual keyboard where the controller pressed the keys of the flight ID. This ID is composed by a sequence of three letters (maximum) which represents the airline code, followed by the flight number. When such a selection is done, the associated flight is made highlighting on the radar screen. Depending of the flight ID distribution on a control position, the virtual keyboard can be optimized in order to speed up the aircraft selections and to improve the work of the controllers mainly when the sectors are overloaded. This keyboard optimization problem may be addressed like a pure assignment problem which is NP_Hard. This paper shows how artificial evolution has been used for solving such a problem with very good results on real instance associated to the Roissy departure sector.

1 Introduction

When an aircraft flies from a city A to a city B, it has to be managed by air traffic controllers in order to avoid collisions with others aircraft. Everyday, about 8000 aircraft fly in the French airspace, inducing a huge amount of control workload. Such a workload, is then spread by the mean of the airspace sectoring.

⋆⋆⋆ Centre d'Études de la Navigation Aérienne
† École Nationale de l'Aviation Civile

P. Liardet et al. (Eds.): EA 2003, LNCS 2936, pp. 177–188, 2004.

The airspace is divided into geometrical sectors, each of them being assigned to a controller team. When a conflict between two (or more) aircraft is detected, the controller changes their routes (heading, speed, altitude) in order to keep a minimum distance between them during the crossing. All flying aircraft are then monitored during their navigation and so from the departure till the destination. This monitoring is helped by mean of the flight ID. This ID is a code associated to the flight composed by several letters related to the airline, followed by a number. For instance, the ID TWA810 is associated to the flight Boston-Paris operated by the TWA airline. All airlines have a normalized code given by the ICAO[1] authority. This code has a maximum length of 3 letters. Air France has the code "AFR", British Airways "BA" and so on. In order to proceed a flight, the pilot has first to produce a flight plan which is a kind of summary of the preferred navigation route. This flight plan gives the flight ID and a list of navigation segments connected by normalized way points (points in the airspace extracted from an official set produced by the civil aviation authority). For all those points (2D), the pilot must produce the associated altitude or the flight level[2] of the aircraft when it will be above this point.

The flight ID selection is done by mean of a numerical keyboard for which letters are associated to numbers like in a telephone keyboard. The fact that two aircraft belonging to the same airline being in the same controlled area at the same time, is very rare and the flight ID selection is done only on the letter associated to the airline. When such uncommon event happen, the selection is extended with the number.

Nowadays, the controller enters all letters of the flight ID. Instead of using a physical keyboard, we propose to use a virtual one for which the alphabetic association to the numeric keyboard will be optimized in order to speed up the selection process.

For all controlled areas, it is easy to determine the number of Air France flights, the number of TWA flights etc. and to build statistics about the airline codes. This counting is done on the year base. An example of such a statistic is given in the following table:

Number of flights	Airline ID	Airline	Percentage
44	AFR	Air France	23
24	BA	British Airways	12
21	LF	Lufthansa	11
...	...	...	...

Our problem is to synthesize a dedicated virtual keyboard to any controlled area in order to minimize the average selection time of aircraft IDs. This keyboard

[1] International Civil Aviation Organization

[2] A flight level is a measure of altitude, given in hundred of feet by an altimeter referenced to the 1013 hpa altisurface (average pressure at the sea level). For instance if an altimeter measures a pressure of 164 hpa, there is a difference of 164-1013=-848 hpa which gives an altitude difference of 848∗23=19504ft ($1hpa \Rightarrow 23ft$); then the associated flight level is 195.

is based on the numerical keys like the telephone keyboard but the distribution of the letters on those keys have to be optimized. The numerical keys has to be kept because such a keyboard is also used for others numerical tasks of the controller work.

In a first part, a refined description of the problem is given for which the performance and the constraints of the virtual keyboard are given. The second part proposes an associated mathematical modeling of the problem. The third and the fourth parts show how artificial evolution has been applied to such a problem. Finally the fifth part presents results on some real traffic sets.

2 Problem Description

Based on the airline statistics associated to the considered controlled area, one has to find the optimal letter assignment on a numerical keyboard. The numerical distribution of the keys on a telephone keyboard is given Figure 1.

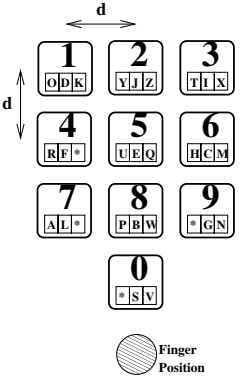


Fig. 1. Distribution of the letter on a telephone keyboard

The three boxes under each number are the potential positions of the letters; so there are 30 positions where the airline ID letters can be included (a random association has been represented in this example). Such a keyboard will generate a translation between the alphabetic code (airline descriptor) and the resulting numerical code. Having more boxes (30) than alphabetic letters (26), 4 null characters will be included in such alphabet for completion. This association of letters to the numerical keyboard has to optimize the following two criteria:

1. *the total number of pressed keys;*
2. *the total distance on the keyboard.*

Depending on the letter distribution, it is possible to produce automatic completion without ambiguity. As a matter of fact, suppose the following unrealistic

airline code set is given :{AAA 50%;GGG 50%}. It says that 50% of the aircraft has the airline code "AAA" and the other 50% has the code "GGG". It is easy to understand that the first letter is enough to remove the ambiguity between the two possible airline codes subject that the two letters are located on two different keys. In such example, the number of pressed key reductions is given by : (50%.2 + 50%.2).total number of aircraft. When the system has not ambiguity anymore, it can highlight the associated aircraft.

For the key sequence which have to be pressed on the keyboard, the associated distance has also to be minimized. The keyboard will try to put as close as possible the letter associated to the most frequent letter codes.

Such optimization has to avoid also collisions between the resulting numerical codes. As a matter of fact, different sequences of letters have to be coded by different sequences of numbers in order to avoid ambiguities. For instance, in the previous example, if the two letters "A" and "G" are coded by the same number, the system will not be able to distinguish between the two airlines.

Some previous related works may be found in the reference [16] but the objective is to compare the performances of several virtual keyboards without optimization.

The problem that has to be solved is very closed to the one of information compression as it appears in the information theory [6]. In such compression process, a file composed of different characters has to be reduced in size in order to be transmitted or stored on any data support. Initially, all the characters are coded with the same binary digit sequence length. The idea consists in coding with short binary digit sequences the most frequent characters and by long ones the less frequent ones (Huffman algorithm [12]). Such approach is not adapted for our problem for the two following reasons :

1. to applied the Huffman algorithm to such a problem, every sequence of letters has to be considered as a "super-letter" of an alphabet composed of $27 \times 27 \times 27 = 19683$ characters (26 alphabetic letters plus the null character). The Huffman algorithm will then produce a transcoding without any meaning for the controller. For instance all the Air France (AFR) flights would be coded by the sequence 24, the British Airways (BA) by the sequence 39. The two initial alphabetic codes have a common letter, but the produced numerical codes have no intersection;
2. the less frequent sequences could be coded with longer numerical sequences compared to the maximum length of three in the alphabetic sequence which is also very disturbing for the controller.

Instead of using such information theory tools which are not adapted enough to our problem, we propose to address this problem from the combinatoric optimization [13] point of view. Before presenting the method which has been used, a mathematical model is first presented.

3 Mathematical Modeling

The problem that has to be solved consists in finding an optimal assignment of the alphabetic letters to the numerical keyboard with a maximum of three letters per key (see Figure 1). This assignment represents the state space of our problem.

If the initial alphabet is extended with 4 null letters (stars on Figure 1), a point in our state domain can be considered as a permutation of those 30 letters (26 alphabetics plus 4 nulls). If numbers are assigned to all alphabetic letter (A $\rightarrow$ 0; B $\rightarrow$ 1; C $\rightarrow$ 2; ...; Z $\rightarrow$ 25; Null $\rightarrow$ 26; ...; Null $\rightarrow$ 29), state points are pure permutations among those 30 integers. In order to evaluate such a point in the state space, an objective function has to be defined. As it has been mentioned in the introduction, the optimization process must minimize the number of pressed keys and the distance covered on the numerical keyboard during code entering process. To compute this objective, the first step consists in computing the list of numerical codes ($\tilde{N}$) builded from the alphabetic sequences ($\tilde{L}$) and the given permutation (P). Based on this initial list, each numerical sequence is checked with the other ones in order to determine the last superfluous digits which can be removed in the final numerical sequences. Those final sequences will be the ones which will be effectively pressed on the keyboard. The system will then produce the automatic completion and will highlight the associated flight. Those final sequences will be also called *compressed sequences* in the following. In order to compute the distance related to the pressed keys, a distance matrix has been introduced. To build such a matrix, it has been supposed that only one finger presses the keys on the numerical keyboard and the initial position of this finger is under the "0" key as it can be seen on Figure 1.

The associated matrix is given by (the numbers expressed in this matrix are in "d" unit):

$$D = \begin{bmatrix} 0 & 1 & \sqrt{17} & 4 & \sqrt{17} & \sqrt{10} & 3 & \sqrt{10} & \sqrt{5} & 2 & \sqrt{5} \\ 1 & 0.5 & \sqrt{10} & 3 & \sqrt{10} & \sqrt{5} & 2 & \sqrt{5} & \sqrt{2} & 1 & \sqrt{2} \\ \sqrt{17} & \sqrt{10} & 0.5 & 1 & 2 & 1 & \sqrt{2} & \sqrt{5} & 2 & \sqrt{5} & \sqrt{8} \\ 4 & 3 & 1 & 0.5 & 1 & \sqrt{2} & 1 & \sqrt{2} & \sqrt{5} & 2 & \sqrt{5} \\ \sqrt{17} & \sqrt{10} & 2 & 1 & 0.5 & \sqrt{5} & \sqrt{2} & 1 & \sqrt{8} & \sqrt{5} & 2 \\ \sqrt{10} & \sqrt{5} & 1 & \sqrt{2} & \sqrt{5} & 0.5 & 1 & 2 & 1 & \sqrt{2} & \sqrt{5} \\ 3 & 2 & \sqrt{2} & 1 & \sqrt{2} & 1 & 0.5 & 1 & \sqrt{2} & 1 & \sqrt{2} \\ \sqrt{10} & \sqrt{5} & \sqrt{5} & \sqrt{2} & 1 & 2 & 1 & 0.5 & \sqrt{5} & \sqrt{2} & 1 \\ \sqrt{5} & \sqrt{2} & 2 & \sqrt{5} & \sqrt{8} & 1 & \sqrt{2} & \sqrt{5} & 0.5 & 1 & 2 \\ 2 & 1 & \sqrt{5} & 2 & \sqrt{5} & \sqrt{2} & 1 & \sqrt{2} & 1 & 0.5 & 1 \\ \sqrt{5} & \sqrt{2} & \sqrt{8} & \sqrt{5} & 2 & \sqrt{5} & \sqrt{2} & 1 & 2 & 1 & 0.5 \end{bmatrix}.$$

It can be noticed that when a key is pressed two times, the associated distance is not zero but half d. So, for a given numerical code, the distance of the overall path is given by the summation of the individual distances between keys.

$$D(\tilde{N}'_i) = \sum_{j=1}^{|\tilde{N}'_i|-1} \left[d_{\tilde{N}'_{ij},\tilde{N}'_{ij+1}} \right]$$

where $|\tilde{N}'_i|$ represents the length of the considered compressed code ($\tilde{N}$ represents the induce code by a permutation and $\tilde{N}'$ after compression). For instance

the numeric code 753 has the distance $\sqrt{5}+\sqrt{2}+\sqrt{2}$. This distance being computed on the compressed numerical codes (code without superfluous last digits), its minimization meets two objectives (minimization of the number of pressed keys and the remaining distance). In order to take into account the statistics associated to each code, the distance is weighted by the number of flight having this code. So, a permutation P is evaluated the following way :

$$O(P) = \sum_{i=1}^{i=|N'(P)|} D(\tilde{N}'_i).N_B(\tilde{N}'_i)$$

where $N'(P)$ is the set of numerical codes associated to P after compression; and $N_B(\tilde{N}'_i)$ the number of flights with the code $\tilde{N}'_i$.

Having now defined the state space and the objective function, one has to identify the associated constraints of this problem. The main constraint can be summarized the following way: *two different alphabetic sequences have to produce two different numerical sequences*. For a given permutation P, the associated constraint violation criterium is computed by counting the identical compressed numerical code induced by a permutation P. The number of constraint violations is then given by :

$$N_v(P) = \sum_{i=1}^{i=|N'(P)|} \sum_{\substack{j=1 \\ j=i}}^{j=|N'(P)|} \delta[\tilde{N}'_i(P), \tilde{N}'_j(P)]$$

where

$$\delta[\tilde{N}'_i(P), \tilde{N}'_j(P)] = \begin{cases} 1 & \text{si} \quad \tilde{N}'_i(P) = \tilde{N}'_j(P) \\ 0 & \text{sinon.} \end{cases}$$

This number is taken into account in the optimization process by relaxation. Then, the constraint factor will be stronger at the end of the optimization process than at the beginning.

Having now defined a mathematical model for our problem one must identify the associated complexity. As it has been previously shown, one has to find an optimal assignment of 26 letters to 30 boxes. The number of possibilities is given by[3]: $\frac{30!}{4!}$.

The problem we have to solved is an assignment problem which is known to be NP_Hard and artificial evolution is quite adapted to address such a problem.

4 Artificial Evolution

Artificial evolution is an optimization algorithm belonging to the class of stochastic optimization techniques. Simulated annealing [8], taboo search [4], branch and probability bound [17] etc..., belong also to such class. Such techniques usually

[3] The first letter has 30 choices, the second 29 and so on: $30.29 \ldots 5 = \frac{30!}{4!}$ (the number 4 comes from the 4 null letters which have been included).

addressed problem with strong complexity for which the objective function has no specific feature such as convexity, continuity etc. The main feature of those stochastic optimization techniques is to randomly move in the state domain in order to improve the objective criterium. Evolutionary Algorithms (EA) [5,7, 3,11,9,2,14] are problem solving systems based on principles of evolution and heredity. An EA maintains a population of individuals,$P(t) = x_1, x_2, \ldots, x_n$ at iteration t. Each individual represents a potential solution to the problem at hand and is implemented as some (possibly complex) data structure S. Each solution x_i is evaluated to give some measure of fitness. Then a new population at iteration $t + 1$ is formed by selecting the fitter individuals. Some members of the new population undergo transformation by means of genetic operators to form new solutions. There are unary transformations m_i (of mutation type), which create new individuals by a small change of a single individual and higher order transformations c_i(crossover type), which create new individuals by combining parts from several (two or more) individuals. For example, if parents are represented by a five-dimensional vector $(a_1, a_2, a_3, a_4, a_5)$ and $(b_1, b_2, b_3, b_4, b_5)$, then a slicing crossover of chromosomes after the second gene produces offspring $(a_1, a_2, b_3, b_4, b_5)$ and $(b_1, b_2, a_3, a_4, a_5)$. The control parameters for genetic operators (probability of crossover and mutation) need to be carefully selected to provide better performance. The intuition behind the crossover operation is information exchange between different potential solutions. After some number of generations the program converges - the best individual hopefully represents the optimum solution.

The structure of the EA which has been used for our experiments is given Figure 2. To move from generation k to generation $k + 1$, the population is first

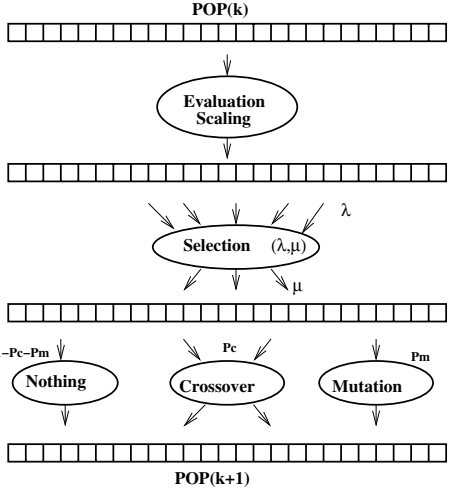


Fig. 2. Structure of the Evolutionary Algorithm

evaluated and the associated fitness is also scaled using a classical power law scaling principle [5]. Afterward, the best individuals are selected with a tournament selection $(\lambda; \mu)$ and then undergo the recombination process. Then, individuals can be put straightly in the next population ("nothing" operator), crossed over (crossover operator) or mutated (mutation operator). Those operators are randomly selected based on their associated probabilities ($(1 - P_m - P_c) \rightarrow$ nothing; $P_c \rightarrow$ crossover; $P_m \rightarrow$ mutation). The probability of mutation is self adapted to the problem using the Reichenberg rule with a constraint interval limiting this probability between $[0.5, 0.7]$. This scheme has been successfully used for our keyboard optimization experiments.

5 Application to Keyboard Optimization

5.1 Coding

In order to make run such Evolutionary algorithm, a coding of the state space has been developed. Two potential codings have been proposed. The first one is based on the pairwise decomposition property of any permutation of size N. As a matter of fact, any permutation of size N can be decomposed into a list of exchanges between pair of positions, subject that the minimum number of swaps is at least $N \log(N)$. From the mathematical point of view, this property can be formulated the following way. Suppose that a permutation P of size N and a pairwise swapping operator T are given. When the permutation P is applied to an initial list L_1, it produces a new list noted L_2. This means that $L_2 = P(L_1)$. This list L_2 can also be reach by the successive application of the operator T on some pairs of position: $L_2 = T_1 \circ T_2 \circ \ldots \circ T_K(L_1)$ subject that K is greater than $N \log N$ (where $\circ$ is the composition operator). The structure of such a coding is given Figure 3. This coding is convenient because crossover like slicing

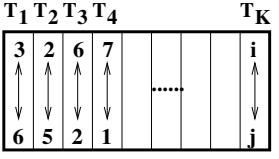


Fig. 3. Coding based on pairwise permutation operator T

or uniform can be straightly applied to any chromosome without checking the integrity of the children (as a matter of fact, the application of such a crossover insures that the produced permutation will be valid (one letter is assigned to only one numerical key)). In the some way, it is very easy to develop a mutation operator by randomly choosing a locus in the chromosome and put a new random pairwise permutation.

Unfortunately, this coding produced poor results in our experiments and a more straight one has been used without crossover operator. This new coding is just an array of integers describing the overall permutation as it can be shown Figure 4.

i	1	2	3	4	5		28	29	30
P(i)	17	24	9	27	12		1	3	15

Fig. 4. Straight coding of the permutation P

In this example the 17th letter is assigned to the first position, the 24th to the second one and so on.

The mutation operator consists in exchanging two random positions (i, j) in the chromosome like it can be seen Figure 5 (in this example the size of the chromosome is 12 instead of 30 in our problem).

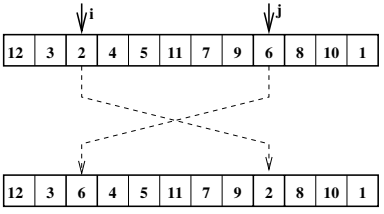


Fig. 5. Mutation operator

The associated fitness evolution takes into account the constraint criterium (N_v) and the overall distance induced by the keyboard. As it has been previously noticed, this distance minimization takes into account the code compressions (when a flight can be selected only by the first digits in the numerical code, the final ones has not to be pressed on the numerical keyboard). The fitness associated to a chromosome C is then given by :

$$Fitness(C) = e^{-\lambda . N_v(C)} . \frac{1}{\left(\frac{D(C)}{N_F}\right)}$$

where $\lambda = \frac{gen}{nb\_gens}$ (gen current generation number; nb_gens: total number of generations), $D(C)$ the total distance associated to the chromosome C and N_F the total number of flights. To compute $D(C)$ and $N_v(C)$, the keyboard is build from the permutation associated to C. Then all alphabetic codes are then converted into numerical codes and compressed when it is possible. Based on this set

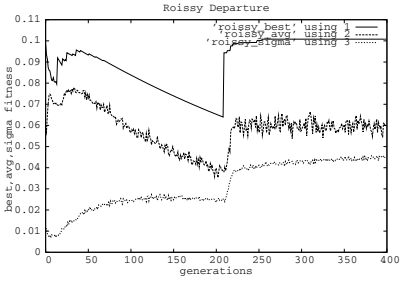
of numerical codes, the number of collisions is then computed (N_v: the number of numerical codes which are similar) which is the number of constraint violations. The overall distance on the proposed keyboard is then computed using the distance matrix. This distance takes into account the number of flights associated to a given numerical code.

6 Results

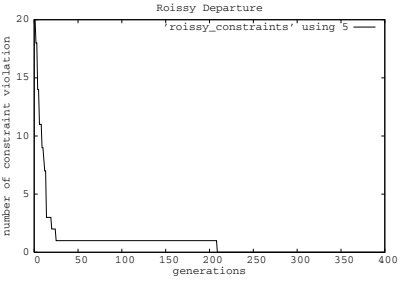
This method has been applied to the departure sector of Roissy Airport. During one year this sector is crossed by about 300000 aircraft. The set of alphabetic codes is given on the following table (only the beginning of the table is represented).

Airline ID	AFR	AF	DLH	BAW	FPO	AZA	SAS	BCY	BMA	...
Nb Flights	97071	23689	14324	9165	7359	7332	4862	4328	3944	...

The parameters of the evolution were the followings: population size 300; number of generations 400; $\lambda = 7$, $\mu = 2$ and initial probability of mutation 0.6. The evolution of the best fitness, the average and the associated dispersion are given Figure 6(a). This fitness has to be maximized and the observed decreasing is due to the constraint relaxation for which the constraint violation criterium becomes stronger and stronger. The discontinuities of the fitness appear when the number of constraint violations decreases (see 6(b)). The associated keyboard distance reduction and the number of compressions are represented on figure 7.



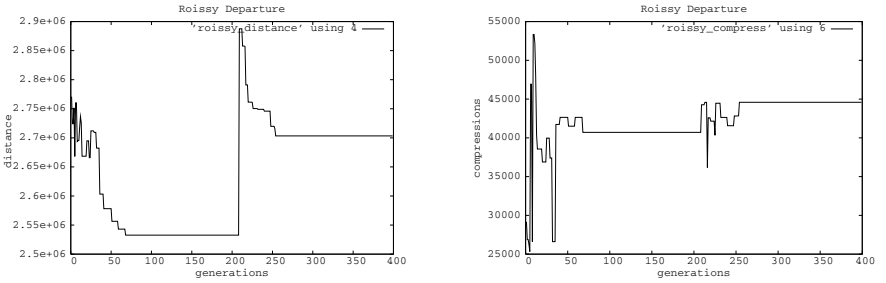
(a) Best, Avg, Sigma



(b) Number of constraint violations

Fig. 6. Fitness and constraint violations evolution

As it can be seen on the figure, the distance has a smaller value at the beginning of the evolution than at the end but it is also the period where the constraint are not respected. The final distance is about $2.7.10^6$ which is really smaller to the one induced by the actual keyboard: $3.9.10^6$. The associated reduction of pressed keys is about 45.10^3. Finally the synthesized keyboard is represented Figure 8. The optimization process has reduced significantly the distance of the most frequent airline codes like AFR, DLH, BAW (closest to the finger position).



(a) Distance minimization (b) Compression maximization

Fig. 7. Distance and compression evolution

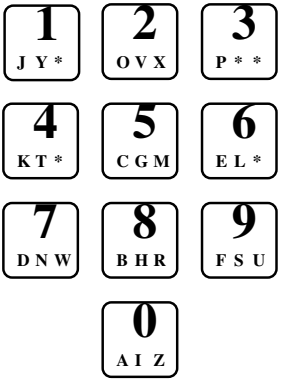


Fig. 8. Synthesized keyboard

7 Conclusion

This paper has presented a real word application of the Artificial Evolution technique. Results proposed by the algorithm really improve the selection performance of the flight ID on the control position. The paper has first described the operational context of the flight ID selection and the underlying keyboard optimization problem. The objective function, the state space and the associated constraints have been presented with the associated mathematical formulation. Due to the induced complexity of such a problem, stochastic optimization has been supposed to be a good candidate to solve this problem. Artificial evolution has been presented and adapted to the keyboard optimization problem. Finally, the algorithm has been tried on a real instance of the problem related to departure sector of Roissy Airport. The given results really improved the flight ID selection with a large reduction of pressed key. In a near future, this tool will be adapted to several control positions with different flight IDs in order to check if the produced results are as good as the ones given for the departure sector of Roissy Airport.

Acknowledgements. We would like to thank Herve Damiano for having proposed such a beautiful combinatoric optimization problem.

References

1. D Colin De Verdière. Le système français de contrôle du trafic aérien : Le CAUTRA. Technical report, CENA, Toulouse France, Aout 1992.
2. D.B Fogel. *Evolutionary Computation. Toward a new Philosophy of Machine Intelligence*. IEEE press, 1994.
3. L.J Fogel, A.J Owens, and M.J Walsh. *Artificial Intelligence Through Simulated Evolution*. Wiley and sons. NY, 1966.
4. A. Glover. *Taboo Search*. Kluwer Academic Publishers, 1997.
5. D.E Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Reading MA Addison Wesley, 1989.
6. D Hankerson, G.A Harris, and P Johnson. *Introduction to Information Theory and Data Compression*. CRC Press, 1997.
7. J.H Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan press, 1975.
8. S Kirkpatrick, C.D Gelatt, and M.P Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.
9. J.R Koza. *Genetic Programming*. MIT press, 1992.
10. G Maignan. *Le Contrôle de la Circulation Aérienne*. Presse Universitaire de France, 1991.
11. Z Michalewicz. *Genetic algorithms + Data Structures = Evolution Programs*. Springer-Verlag, 1992.
12. M Nelson and J.L Gailly. *The Data Compression Book*. Hungry Minds, Inc, 1995.
13. C.H Papdimitriou and K Steiglitz. *Combinatorial Optimization : Algorithms and Complexity*. Prentice-Hall, New York, 1982.
14. H.P Schwefel. *Evolution and Optimum Seeking*. Wiley, New York, 1995.
15. P.L Tuan, H.S Procter, and G.J Couluris. Advanced productivity analysis methods for air traffic control operation. Technical report, Stanford Research institute, Menlo Park CA 94025, December 1976.
16. S. Zhai, B.A Smith, and M Hunter. Performance optimization of virtual keyboards. Human-Computer Interaction, 2001.
17. A. A. Zhigljavsky. *Theory of Global Random Search*. Kluwer Academic Publishers, 1991.

Post Docking Filtering Using Cartesian Genetic Programming

A. Beatriz Garmendia-Doval, S. David Morley, and Szilveszter Juhos

RiboTargets Ltd, Granta Park, Cambridge, England, CB1 6GB
beatrizagd@yahoo.co.uk, d.morley@ribotargets.com szilva@computer.org

Abstract. Structure-based virtual screening is a technology increasingly used in drug discovery. Although successful at estimating binding modes for input ligands, these technologies are less successful at ranking true hits correctly by binding free energy. This paper presents the results of initial attempts to automate the removal of false positives from virtual hit sets, by evolving a post docking filter using Cartesian Genetic Programming.

1 Introduction

Structure-based virtual screening (see [1]) is an increasingly important technology in the hit identification and lead optimisation phases of drug discovery. It provides a fast, low-cost alternative to experimental High Throughput Screening (HTS) of large compound libraries and promises to help reduce the time and effort required to identify candidate drug molecules for further clinical development.

The goal of structure-based virtual screening is to identify a set of small molecules (ligands) that are predicted to bind to a defined target macromolecule (protein or nucleic acid). Through the combination of fast molecular docking algorithms, empirical scoring functions and affordable computer farms, it is possible to virtually screen hundreds of thousands or even millions of ligands in a relatively short time (a few days). The output from the docking calculation is a prediction of the geometric binding conformation of each ligand along with a score that represents the quality of fit for the binding site. Only a small fraction of the top-scoring virtual hits (typically up to 1000) then are selected for experimental assay validation. If successful, this virtual hit set will be significantly enriched in bioactive molecules relative to a random selection and will yield a number of diverse starting points for a medicinal chemistry 'hit-to-lead' programme.

Although many factors contribute to the success of virtual screening, a critical component is the scoring function employed by the docking search algorithm. During the last ten years a variety of docking programs have been implemented and published in the literature (e.g. FlexX [2] and GOLD [3,4]). The majority of these use so-called empirical scoring functions to estimate the binding energy of a ligand conformation in terms of well-recognised physicochemical interactions

P. Liardet et al. (Eds.): EA 2003, LNCS 2936, pp. 189–200, 2004.

such as hydrogen bonding, ionic, and hydrophobic interactions. Whilst reasonably effective at reproducing the binding geometries of known ligands, empirical scores are less successful at ranking true hits correctly by binding free energy. This is a natural consequence of the many approximations made in the interests of high throughput and, as such, all virtual hit sets contain false positives to a greater or lesser extent. One contributing factor is that empirical scoring functions are calibrated against known target-ligand structures that, by definition, only represent observed interactions. Some of the false positives contain non-observed combinations of molecular interactions that are not explicitly penalised by the scoring function (see [5]). Many of these can be removed manually by visual inspection of the predicted binding geometries by an expert computational chemist, but this is a time consuming process.

There have been previous studies that used Genetic Algorithms to improve the coefficients of the scoring function (see [6]). Also Böhm ([5]) developed an empirical postfilter for the docking program FlexX using penalty functions. This paper presents the results of our initial attempts to apply Genetic Programming (GP) techniques to automate the removal of false positives from virtual hit sets provided by our in-house virtual screening platform, rDock.

2 rDock Virtual Screening

Here at RiboTargets rDock ([7]) and its predecessor RiboDock ([8]) have been developed as docking platforms that can rapidly screen millions of compounds against protein and RNA targets. It also has "pluggable" modules that are designed to give a more reliable estimation of the binding mode and energy at the expense of CPU time. The program outputs binding modes, score components, and also many additional descriptors that can be used during post-processing for filtering purposes.

rDock is able to screen libraries with millions of compounds and return a list of virtual hits of several thousands. This list is still too large and it should be further reduced before selecting the compounds that will be assayed experimentally. This process is left to be done by the computational chemists that use all their expertise to reduce the amount of virtual hits while maintaining real hits in the set.

During docking rDock tries to minimise the total score:

$$S_{total} = S_{inter} + S_{intra} + S_{restraint}$$

where S_{inter} stands for the sum of all the intermolecular scoring functions, S_{intra} is the ligand intramolecular term and $S_{restraint}$ is a penalty term that considers the deviation from certain restraints, for instance when part of the ligand is outside the docking cavity.

Using this score rDock searches for the best conformation for a given ligand over a given docking site. At the end, rDock stores the ligands for which a conformation with a low enough score has been found. These are the ligands that will be considered virtual hits.

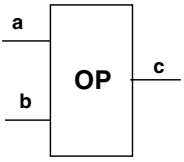


Fig. 1. CGP Cell

Once all the hits are found, the value of the score is no longer meaningful. The score is good enough to compare two different conformations of a given ligand, but not good enough to accurately rank order different ligands.

rDock outputs the score and its constituents. rDock also outputs additional descriptors for both the ligand and the target docking site such as molecular weight, number of aromatic rings, charge, number of atoms, etc., that are not used directly during docking. This information is used in an ad hoc manner by the computational chemists to filter out manually the virtual hits, often on a per-target basis, for example to ensure a desired balance between polar and apolar interaction scores. We have explored the use of Genetic Programming techniques to automatically evolve more complex, target-independent post-docking filters.

3 Cartesian Genetic Programming

In standard GP ([9], [10]) a program is implemented as a tree, where the terminal nodes (or leaves) are inputs (variables and constants) and the branching nodes are functions to be applied to the terminals. During program evolution, crossover and mutation operators can affect the depth of the tree and the functions applied at the nodes. Each tree represents an evolved program explicitly.

In Cartesian Genetic Programming (CGP), developed by Julian Miller ([11]), each function is represented by a cell as shown in figure 1. In this case the cell represents c = a op b with op representing one of the valid operations for the given problem. If op only asks for one input, the value of b can be ignored, as in c = op a.

Each cell is represented by a string of integers. In this example, by three integers, the first two representing the input arguments and the third the operator. The inputs can refer to inputs of the program (i.e. variables and constants) or to other cells, i.e. the output value of another cell. The cells can then be chained together to form a program. Figure 2 represents a CGP program. The inputs of the program are two variables, represented in the left by 0 and 1. In this case there are 6 cells which are given the numbers 2-7. The output is the result given by cell 6. There are four operators represented by the integers 0-3. They represent respectively addition, subtraction, multiplication and division. Cell 6 is represented by the integers 2 5 3. This means take the output of cell 2 and divide it by the output of cell 5. Cell 5 is represented by the integers 0 3 2. This means take the output of 0 (simply the value of the input variable 0), and multiply it by the output of cell 3. And so on.

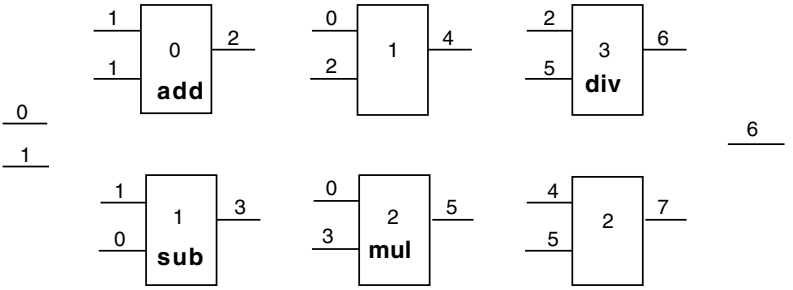


Fig. 2. CGP Program

Given two inputs, 6 cells, and four operations, the CGP program is coded by the string

1 1 0 1 0 1 0 2 1 0 3 2 2 5 3 4 5 2 6

and it represents the function

$$(v1 + v1)/(v0 * (v1 - v0)) \ .$$

The string has 19 integers and although cells 7 and 4 do not play a role in this case, they are also represented in the string. If the last value of the string is changed from 6 to 7:

1 1 0 1 0 1 0 2 1 0 3 2 2 5 3 4 5 2 7

the output of the program will be taken from cell 7 instead, and all the cells except 6 will play a role. In that case the string will represent the function

$$(v0 - (v1 + v1)) * (v0 * (v1 - v0)) \ .$$

With two inputs, one output, four operations and a structure of 3 by 2 cells, it is possible to use a string of 19 integers to represent quite different functions. By using a string all the normal Genetic Algorithm operators are available. As the structure is fixed, the resulting functions cannot go above a given size. The cells that are not used for a given function, act as introns. A mutation or crossover can turn them from inactive to active. The consequence of this approach is that the genotype (string of integers) is completely separated from the phenotype (the function represented)([12]). This is in contrast to standard GP where a tree is both the phenotype and the genotype.

A common problem of standard implementations of GP is the increase of size for the average program as the population evolves, known as bloating. Much of this bloating is due to code that can be eliminated without modifying the behaviour of the program, so called "junk code" (see [13]). As the average size increases, more memory and CPU time is needed. Also, smaller programs tend to generalize better. These reasons make it necessary to control bloating when implementing a GP, normally by not allowing the trees to grow further than a maximum size.

Bloating is not a problem in CGP. Although the phenotype, the function, can vary greatly in size, the genotype has a fixed size, the length of the string. Julian Miller also has found that the amount of junk code is greatly reduced or even non-existent (see [14]). We decided to implement a CGP for all these reasons. As we have not tried the standard GP on this problem, this paper does not pretend to be a study or comparison between standard GP and CGP. On the other hand, it must be said that at least for this problem, CGP has proven to be a good choice as bloating was indeed absent.

4 Implementation

The GP implemented is a standard CGP. The structure is a matrix with 1 row and 200 columns. A cell can use as inputs any cell 100 before itself, counting the input variables as the first cells. All cells have three inputs and one output. The operations implemented are

Addition

$$+$$

Subtraction

$$-$$

Multiplication

$$*$$

Division

$$div(a,b) = \begin{cases} a & \text{if } |b| < 0.000001 \\ a/b & \text{otherwise} \end{cases}$$

Logarithm

$$\log(a) = \begin{cases} 0 & \text{if } |a| < 0.000001 \\ \log(|a|) & \text{otherwise} \end{cases}$$

Exponential

$$\exp(a) = \begin{cases} 0 & \text{if } a < 200 \\ \exp(200) & \text{if } a > 200 \\ \exp(a) & \text{otherwise} \end{cases}$$

Conditional

$$if(a,b,c) = \begin{cases} b \text{ if } a > 0 \\ c \text{ otherwise} \end{cases}$$

Random constant. First time this command is called for a given cell, it will create a new random constant. That will be the value of the cell for the rest of the program, unless a mutation operator changes the operation of the cell.

The input to the program is the data returned by rDock. There are components to S_{inter}, S_{intra}, and $S_{restraint}$, ligand descriptors and docking site descriptors. Some of these descriptors are explained in detail in table 2.

Apart from the input variables, there is also a pool of 15 constants introduced as input. Each time the CGP is run, 13 of them will be created at random. The other 2 are the constants 0.0 and 1.0. A random constant is equal to $a * 10^b$ where a is a float (with just one decimal) number between -10 and 10 and b is an integer between -5 and 5.

In total there were 66 input variables, although a given filter did not have to use all of them. In average 10 to 15 variables were used by individual filters.

The evolutionary algorithm implemented was also taken from Miller's work ([11]). It is a simple form of $(1 + \lambda)$ Evolutionary Strategy with λ equal 4, no crossover and a mutation rate of 8%. The algorithm can be summarised as:

```
Initialize random population
while no max number of generation
  - choose fittest individual
  - mutate the fittest individual enough times to fill up the
    rest of the population
```

The genetic algorithm described in [11] was also tried but, as the results obtained were no better than those presented here and required more CPU time, this method was discarded. Also, a series of experiments were conducted in which system parameters such as the structure of the matrix, number of cells, number of constants, and mutation rate were varied. The fitness function of the CGP is based on the result of applying the current filter over the training set. As this is a classification problem, our aim is to maximize the classification accuracy over the test set. Being able to find the global optimum for the training set is not our goal as this will almost surely be equivalent to overfitting and will produce a filter that will perform poorly over new data. Because of this, once a system capable of finding good local optima was identified, the system parameters were fixed.

5 Application

5.1 Training Set

We have assembled a set of 163 targets, for each of which there is a structure available of the target bound to its native ligand.

Each of the 163 ligands have been docked against each of the targets. If the scoring function used in docking were perfect, then the lowest (best) score for each target would be obtained for the native ligand bound.

As our current ability to calculate physical properties is quite limited, the native ligand only ranks first in a few cases. Therefore, this cross-docking set contains a large number of false positives. These can be used to drive a genetic program to evolve appropriate filters based on various calculated properties.

Table 1. Results of Best Filter for training set and test set

		Correctly Classified	Incorrectly Classified
Training Set			
	Native Ligands	28	2
	Non-Native Ligands	36	21
Test Set			
	Native Ligands	89	44
	Non-Native Ligands	1946	2417

From the targets for which the corresponding native ligand ranks in the 9th position or higher, 30 were chosen at random. The training set is then these 30 targets, where the native ligands are considered hits and the ligands with a higher rank are considered misses.

5.2 Fitness Function

The CGP implemented evolves numerical functions. For each input (i.e., docking score components of a given ligand over a given docking site, together with the ligand and docking site descriptors), a float number is returned. It was decided to interpret these numbers in the following manner:

$f < 0$ represents a hit
$f >= 0$ represents a miss

For each protein in the training set there is one hit (native ligand) and a list of size between 0 and 8 with misses (ligands that score better than the native ligand).

The fitness function counts the number of proteins for which the hit was recognised as hit ($f < 0$) and at least $\frac{2}{3}$ of the misses were recognised as misses ($f <= 0$).

5.3 Test Set

The test set corresponded to the rest of the cross-docking matrix, i.e., the 133 proteins left after removing the 30 ones used for training. The reason for the test set being so much larger than the training set was due to the fact that only half of the matrix was available at the beginning of the project. Once the other half was made available, it was added directly to the test set.

The CGP was run several hundred times and the filters that performed best over the test set were chosen. These were then further tested over our validation set.

5.4 Seeded Libraries

Seeded libraries are one of a number of methods that have been developed to assess the performance of virtual screening programs. A seeded library for a given target is a library of drug-like compounds that includes several native ligands known to bind to that target. All the compounds (both native and non-native) are docked into the given target using the docking platform. The output conformations are then sorted by the score. The ability of the virtual screening software to distinguish between the native ligands and the non native ones can then be measured, typically by calculating the percentage of native ligands found in the top x%.

We tested the best filters over four seeded libraries, and the most promising was chosen. This filter can be seen in figure 4. The variables used by this filter are described in table 2 and the results obtained in the training set and the test set can be seen in table 1.

Figure 3 shows the top 1% (blue), top 2% (green) and top 5% (brown) completeness for these four seeded libraries. We had 10 native ligands for proteins 1 and 3, and 17 for proteins 2 and 4. The first column shows the results of rDock. The second column shows the results of applying the figure 4 filter.

For protein 1, there is not a great improvement using the filter, as rDock already produces very good results and is therefore difficult to improve upon them.

Protein 2 shows a nice improvement as some of the native ligands that were previously found on the top 2% are now found on the top 1%. Similarly for protein 4, all the native ligands found before in the top 2% are now found in the top 1%.

Finally protein 3 gives the best results as more native ligands are found in the top 1% (40% vs. 30%), more in the top 2% (80% vs. 60%) and again more in the top 5% (100% vs. 90%) where all the native ligands are found.

5.5 Best Filter

There were several filters that worked well for the cross-docking matrix but most of them did not generalise well for the seeded libraries. They either filtered out native ligands or, most commonly, they filter almost nothing out. The following filter on the other hand generalises quite well.

It should be emphasised that the cross-docking matrix is a quite different experiment from the seeded libraries. The fact that this filter is able to filter out true misses while maintaining most of the true hits in both experiments is quite encouraging and is relatively safe to infer that somehow it has found some general trends in the data.

Although it is difficult to understand exactly what the filter is doing, the filter combines intermolecular score components (as used during docking) with both protein and ligand properties in a chemically meaningful way. For example, highly strained conformations (SCORE.INTRA.VDW.raw) and steric clashes

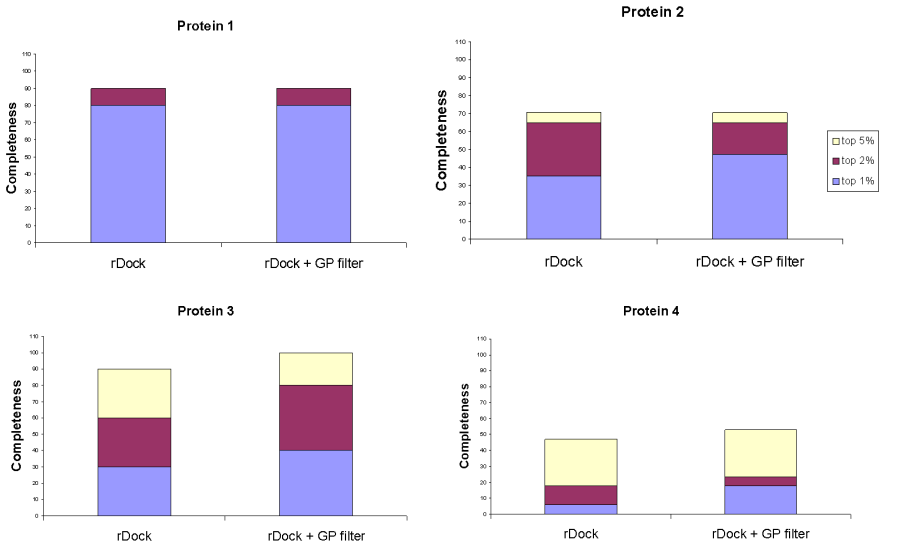


Fig. 3. Seeded Libraries

between ligand and target (SCORE.INTER.VDW.nrep) are more likely to be rejected.

Finally it should also be noted that the only simplifications done over the original filter output by the CGP program and this filter are replacing the expression $\exp(-0.0087)$ for 0.9913 and the expression $-(900 * -0.76)$ for 684. Some parenthesis that were not necessary were also removed to make it more readable. As reported in [14], in all the programs found by CGP for this problem, there were "either very weak program bloat or *zero bloat*[1]".

6 Results with Real Data

All the previous results shown were obtained over idealised test sets used routinely to measure docking performance. As a final validation we have applied the filter in figure 4 to real virtual screening data from docking campaigns performed at RiboTargets, specifically against an oncology target protein, HSP90.

From an initial docking library of around 700000 compounds, a total of around 40000 virtual hits were identified over several docking campaigns against HSP90. Around 1500 of the virtual hits were selected by a computational chemist for experimental assay using a variety of ad hoc post filters, and knowledge and experience of the target protein, in a process taking around a week. Thirty of the assayed compounds were confirmed as real hits, in that they showed significant activity against HSP90.

[1] emphasised in the original

```
 log(SCORE.INTRA.VDW.raw - 0.9913) *
     exp(SCORE.INTER.AROM.narom *
         exp(SCORE.INTER.POLAR.nhbd) * LIG_POS_CHG)
+ 684 *
     if  SCORE.INTER.REPUL.nhba $>$ 0   then
         LIG_NEG_CHG
     else
         SCORE.INTER.VDW.nrep
         --------------------
         SITE_PERC_AROMATOMS
     end
-
  if  (SCORE.INTER.POLAR.nhbd -
       SCORE.INTRA.REPUL.raw + LIG_TOT_CHG )  > 0   then
      SITE_NLIPOC
  else
      exp(SITE_NEG_CHG) - log(LIG_NHBD)
  end
```

Fig. 4. Best filter found to date

The figure 4 filter was applied to the virtual hits (see Table 3) and was able to remove 29% of the original unfiltered hits, whilst only removing 4% of the compounds manually selected for assay. Three of the true actives were also removed.

The GP-derived filter therefore shows very good agreement with the manual filtering process, in that the filter passes almost all of the original assayed compounds, but is able to reduce automatically the initial size of the data set by almost 30%. This provides further evidence that the filter is generalising across docking targets quite distinct from those in the training and test sets.

The filter is currently being used and tested with each new docking campaign, with very good results. It promises to be a useful additional tool in the computational chemist's armoury of post-docking filters.

7 Conclusions

Removal of false positives after structure-based virtual screening is a recognised problem in the field. This paper describes what we believe is the first attempt at using Genetic Programming to evolve a post-docking filter automatically.

The cross docking matrix used for training and evolving post-docking filters is quite different from the seeded libraries and the HSP90 data. The post-docking filter chosen from the ones found by the GP platform is filtering out consistently bad compounds in all cases, while retaining interesting hits. We can say that it is generalising over the data. The HSP90 data is the first real data on which the filter has been tested and the results are very promising. This filter is now being

Table 2. Descriptions of the Variables used by the best filter

variables	description
SCORE.INTRA.VDW.raw	sum of the intramolecular van der Waals forces
SCORE.INTRA.REPUL.raw	repulsive polar contacts between the ligand and the target
SCORE.INTER.AROM.narom	number of aromatic rings involve in aromatic interactions
SCORE.INTER.POLAR.nhbd	number of ligand hydrogen bond donors involved in polar interactions
SCORE.INTER.REPUL.nhba	number of ligand hydrogen bond acceptors involved in repulsive polar interactions
SCORE.INTER.VDW.nrep	number of ligand atoms with overall repulsive van der Waals interactions (steric clash)
LIG_NEG_CHG	sum of formal negative charges of the ligand
LIG_NHBD	number of hydrogen bond donors in ligand
LIG_TOT_CHG	total formal charge of the ligand
SITE_PERC_AROMATOMS	percentage of atoms that are aromatic in the target site
SITE_NLIPOC	number of non-polar carbons in the site
SITE_NEG_CHG	sum of formal negative charges of the site

Table 3. HSP90

	Manual post-docking hits	+GP post-docking filter	Reduction
rDock virtual hits	39908	28374	−29%
Compounds assayed	1467	1409	−4%
True actives	30	27	−10%

used as standard in all the projects in the company. Early results confirm its usefulness.

The GP platform offers immediately a pragmatic, automated post-docking filter for cleaning up virtual hit sets. It can be easily applied again for different descriptors or scoring functions.

Longer-term the filters found may offer a way of "boot-strapping" docking scoring function improvements, by identifying non-obvious, yet systematic, defects in the scoring function.

This technique is also not specific to docking programs, and we plan to apply it in the near future for other problems where a list of variables and descriptors is available and there is a need for a generic filter.

Acknowledgements. We thank Mohammad Afshar for his support on this project, Xavier Barril for his help in collecting and reporting the HSP90 data and Rod Hubbard for his helpful comments in preparing this manuscript.

References

1. Lyne, P.D.: Structure-based virtual screening: an overview. Drug Discovery Today **7** (2002) 1047–1055.
2. Kramer, B., Rarey, M., Lengauer, T.: Evaluation of the flexx incremental construction algorithm for protein-ligand docking. PROTEINS **37** (1999) 228–241.
3. Jones, G., Willett, P., Glen, R., Leach, A., Taylor, R.: Development and validation of a genetic algorithm for flexible docking. J. Mol. Biol. **267** (1997) 727–748.
4. Nissink, J., Murray, C., Hartshorn, M., Verdonk, M., Cole, J., Taylor, R.: A new test set for validating predictions of protein-ligand interaction. PROTEINS **49** (2002) 457–471.
5. Stahl, M., Böhm, H.J.: Development of filter functions for protein-ligand docking. J. Mol. Graphics Mod. **16** (1998) 121–132.
6. Smith, R., Hubbard, R., Gschwend, D., Leach, A., Good, A.: Analysis and optimization of structure-based virtual screening protocols (3). new methods and old problems in scoring function design. J. Mol. Graphics Mod. **22** (2003) 41–53.
7. Morley, S., Juhos, S., Garmendia-Doval, A.: rDock: a fast and accurate docking software for virtual screening. in preparation (2003).
8. Afshar, M., Morley, S.: Validation of an empirical rna-ligand scoring function for fast flexible docking using ribodock. in preparation (2003).
9. Crammer, N.L.: A representation for the adaptive generation of simple sequential programs. In Grefenstette, J.J., ed.: Proceedings of an International Conference on Genetic Algorithms and their Applications. (1985) 183–187.
10. Koza, J.R.: Genetic Programming: On the Programming of Computers by Means of Natural Selection. MIT Press, Cambridge, MA, USA (1992).
11. Miller, J.F., Thomson, P.: Cartesian genetic programming. In Poli, R., Banzhaf, W., Langdon, W.B., Miller, J.F., Nordin, P., Fogarty, T.C., eds.: Genetic Programming, Proceedings of EuroGP'2000. Volume 1802 of LNCS., Edinburgh, Springer-Verlag (2000) 121–132.
12. Miller, J.F.: What is a good genotype-phenotype mapping for the evolution of computer programs? Technical Report 364, University of Hertfordshire, Computer Science, University of Hertfordshire (2002).
13. Altenberg, L.: Emergent phenomena in genetic programming. In Sebald, A.V., Fogel, L.J., eds.: Evolutionary Programming — Proceedings of the Third Annual Conference, World Scientific Publishing (1994) 233–241.
14. Miller, J.: What bloat? cartesian genetic programming on boolean problems. In Goodman, E.D., ed.: 2001 Genetic and Evolutionary Computation Conference Late Breaking Papers, San Francisco, California, USA (2001) 295–302.

GUIDE: Unifying Evolutionary Engines through a Graphical User Interface

Pierre Collet[1] and Marc Schoenauer[2]

[1] Laboratoire d'Informatique du Littoral, Université du Littoral - Côte d'Opale
[2] Projet Fractales, INRIA Rocquencourt
Pierre.Collet@univ-littoral.fr, Marc.Schoenauer@inria.fr

Abstract. Many kinds of Evolutionary Algorithms (EAs) have been described in the literature since the last 30 years. However, though most of them share a common structure, no existing software package allows the user to actually shift from one model to another by simply changing a few parameters, e.g. in a single window of a Graphical User Interface. This paper presents GUIDE, a *Graphical User Interface for DREAM Experiments* that, among other user-friendly features, unifies all kinds of EAs into a single panel, as far as evolution parameters are concerned. Such a window can be used either to ask for one of the well known ready-to-use algorithms, or to very easily explore new combinations that have not yet been studied. Another advantage of grouping all necessary elements to describe virtually all kinds of EAs is that it creates a fantastic pedagogic tool to teach EAs to students and newcomers to the field.

1 Introduction

As Evolutionary Algorithms (EAs) become recognised as practical optimisation tools, more and more people want to use them for their own problems, and start reading some literature. Unfortunately, it is today very difficult to get a clear picture of the field from papers or even from the few books that exist. Indeed, there is not even a common terminology between different authors.

Many papers (see e.g. [5]), refer to the different "dialects" of Evolutionary Computation, namely Genetic Algorithms (GAs), Evolution Strategies (ESs), Evolutionary Programming (EP) or Genetic Programming (GP).

From a historical perspective, this is unambiguous: broadly speaking (see [10] for more details), GAs were first described by J. Holland [14] and popularised by D.E. Goldberg [12] in Michigan (USA); ESs were invented by I. Rechenberg [20] and H.-P. Schwefel [22] in Berlin (Germany); L. Fogel [11] proposed Evolutionary Programming on the US West Coast; J. Koza started the recent root of Genetic Programming [16].

However, when the novice reader tries to figure out the differences between those dialects from a scientific perspective, she/he rapidly becomes lost: from a distance, it seems that GAs manipulate bitstrings, ESs deal with real numbers and GP handles programs represented by parse-trees. Very good —so the difference seems to lie in *representation*, i.e. the kind of search space that those dialects search on.

P. Liardet et al. (Eds.): EA 2003, LNCS 2936, pp. 203–215, 2004.

But what about EP then? Original EP talks about Finite State Automata, but many EP papers deal with parametric optimisation (searching a subspace of $\mathbf{R}^n$ for some $n \in \mathbf{N}$) and many other search spaces; and the "real-coded GAs" also do parametric optimisation, while inside the GA community, the issue of representation is intensively discussed [18,26,23], and many *ad hoc* representations are proposed for instance for combinatorial problems [19]. Some ESs also deal with combinatorial representations [13] or even bitstrings [3]; and even in the field of GP, which seems more clearly defined by the use of parse-trees, some linear representations are currently used [17].

So our patient and persevering newcomer starts delving into the technical details of the algorithms, and thinks he has finally found the fundamental differences: those dialects differ by *the way they implement artificial Darwinism.*

- Indeed, GAs use proportional or tournament-based selection, generate as many offspring as there are parents, and the generation loop ends by replacing all parents by all offspring.
- In ESs, each of the μ parents generates a given number of offspring, and the best of the λ offspring (resp. the λ offspring + the μ parents) are deterministically selected to become the parents of the next generation. The algorithm is then called a $(\mu, \lambda) - ES$ (resp. $(\mu + \lambda) - ES$).
- EP looks very much like a $(P+P)-ES$, except that competition for survival between parents and offspring is stochastic.
- In GP, only a few children are created at each generation from some parents selected using tournament-based selection, and they replace some of the parents that are chosen using again some tournament. But wait, this way of doing is precisely called ... Steady-State GA —so this is a GA, then !

And indeed, GP started as a special case of GA manipulating parse-trees and not bitstrings (or real numbers or ...), and became a field of its own because of some other technical specificities, but also, and mostly, because of the potential applications of algorithms that were able to create programs and not "simply" to optimise.

"Ah, so the differences come from the applications then ?" will ask the newcomer. And again, this will only be part of reality, as there is in fact no definite answer: all points of view have their answer, historical, technical, "applicational" or even ... political (however, this latter point of view will not be discussed in this paper, devoted to scientific issues).

But what does the newcomer want to know ? She/He wants to be able to use the best possible algorithm to solve her/his problem. So she/he is certainly not concerned with historical differences (apart from curiosity), and she/he wants to find out first what exactly is an Evolutionary Algorithm, and what different parameters she/he can twiddle to make it fit her/his needs.

Starting from the target application, a newcomer should first be able to choose any representation that seems adapted to the problem, being informed of what "adapted to the problem" means, in the framework of Evolutionary Computation: the representation should somehow capture the features that seem important for the problem at hand. As this is not the central issue of this paper,

we refer the user to the literature, from the important seminal work of Radcliffe [18] to more recent trends in the choice of a representation [6].

The only other thing for which the user cannot be replaced is of course the choice (and coding) of the fitness function to optimise. But everything else that a user needs to do to run an Evolutionary Algorithm should be tuning some parameters, e.g. from some graphical interface: many variation operators can be automatically designed [24,23], and most implementations of artificial Darwinism can be described in a general framework that only requires fine tuning through a set of parameters.

This paper addresses the latter issue, with the *specification of any evolution engine, unified within a single window* of a Graphical User Interface named *GUIDE* (where *evolution engine* means the way artificial Darwinism is implemented in an EA in the selection and replacement steps). In particular, this paper will **not** mention any representation-specific feature (e.g. crossover or mutation), nor any application-specific fitness function.

Section 2 briefly recalls the basic principles of EAs, as well as the terminology used in this paper. Section 3 presents the history of GUIDE, based on the specification language *EASEA* [9]. Section 4 details the Evolution Engine Panel of GUIDE, demonstrating that it not only fulfills the unification of all historical "dialects," but that it also allows the user to go far beyond those few engines and to try many more yet untested possibilities. Finally, section 5 discusses the limitations of this approach, and sketches some future issues that still needs to be addressed to allow a wide dissemination of Evolutionary Algorithms.

2 Basic Principles and Terminology

Due to the historical reasons already mentioned in the introduction, even the terminology of EAs is not yet completely unanimously agreed upon. Nevertheless, it seems sensible to recall here both the basic skeleton of an EA and the terminology that goes with it. This presentation will however be very brief, as it is assumed that the reader is familiar with at least some existing EAs, and will be able to recognise what she/he knows. Important terms will be written in boldface in the rest of the paper.

2.1 The Skeleton

The goal is to optimise a **fitness function** defined on a given search space, mimicking the Darwinian principle that *the fittest individuals survive and reproduce*. A generic EA can be described by the following steps:

- **Initialisation** of *population* Π_0, usually through a uniform random choice over the search space;
- **Evaluation** of the individuals in Π_0 (i.e. computation of their fitnesses);
- **Generation** i builds population Π_i from population Π_{i-1}:
  - **Selection** of some parents from Π_{i-1}, biased by the fitness (*the "fittest" individuals reproduce*);

  - Application (with a given probability) of **variation operators** to the selected parents, giving birth to new individuals, the **offspring**; Unary operators are called **mutations** while n-ary operators are called **recombinations** (usually, $n = 2$, and the term **crossover** is often used);
  - **Evaluation** of newborn offspring;
  - **Replacement** of population Π_i by a new population that is created from the old parents of population Π_{i-1} and the newborn offspring, through another round of Darwinian selection (*the fittest individuals survive*).
- Evolution stops when some predefined level of performance has been reached, or after a given number of generations without significant improvement of the best fitness.

An important remark at this point is that such a generic EA is made of two parts that are completely orthogonal:

- the *problem-dependent* components, including the choice of the search space, or space of **genomes**, together with their initialisation, the variation operators and of course the fitness function.
- on the other hand, the **evolution engine** implements the artificial Darwinism part of the algorithm, namely the selection and replacement steps in the skeleton given above, and should be able to handle populations of objects that have a fitness, regardless of the actual genomes.

2.2 Discussion

This is of course a simplified view, from which many actual algorithms depart. For instance, there are usually two search spaces involved in an EA: the space of genomes, or **genotypic space**, where the variation operators are applied, is the space where the actual search takes place; and the **phenotypic space**, where the fitness function is actually evaluated. The mapping from the genotypic space onto the phenotypic space is called the decoding of solutions, with the implicit assumption that the solution the user is looking for is the phenotypic expression of the genome given by the algorithm. Though the nature of this mapping is of utter importance for the success of the algorithm, it will not be discussed at all here, as GUIDE/EASEA only considers genotypes, hiding the phenotypic space in the fitness.

Other variants of EAs do violate the pure Darwinian dogma, and hence do not enter the above framework: many variation operators are not "blind," i.e. do bias their actions according to fitness-dependent features; conversely, some selection mechanisms do take into account some phenotypic traits, like some sharing mechanisms involving phenotypic distances between individuals [21].

Nevertheless, we claim that the above generic EA covers a vast majority of existing algorithms[1], and most importantly, is a mandatory step for anyone intending to use Artificial Evolution to solve a given problem. In that context,

[1] It even potentially covers Multi-Objective EAs, as these "only" involve specific selection / replacement steps. However, MOEAs are not yet available in GUIDE, but will be soon.

the existence of a user-friendly interface allowing anyone with little programming skills to design an Evolutionary Algorithm following this skeleton is a clear dissemination factor for EC as an optimisation technique: Such were the motivations for EASEA [9] and, later, for GUIDE.

3 GUIDE Overview

GUIDE is designed to work on top of the EASEA language (*EAsy Specification of Evolutionary Algorithms*) [9]. The EASEA language and compiler have been designed to simplify the programming of Evolutionary Algorithms by providing the user with all EA-related routines, so that she/he could concentrate on writing the code that is specific to the application, as listed in section 2, i.e.: the genome structure and the corresponding initialisation, recombination, mutation and evaluation functions.

On the evolution engine side, a set of parameters allows the user to pick up existing selection/replacement mechanisms, and was designed to supersede most existing combinations, while allowing new ones. One of the most important features of EASEA is that it is library-independent: from an EASEA source code, the compiler generates code for a target library, that can be either the C++ libraries *GALib* [25] and EO [15], or, more recently, the Java library JEO [1]. The resulting code is then compiled using the routines from the corresponding library —but the user never has to dive into the intermediate complex object-oriented code.

Unfortunately, whereas the goal of relieving the user from the tedious task of understanding an existing EC library was undoubtedly reached since the very first versions of EASEA, back in 1999, the specification of the evolution parameters implicitly supposes some deep knowledge of existing evolution engines. Moreover, the lack of agreed terminology makes it even difficult for EC-advanced users to pick up their favorite algorithm: GA practitioners, for instance, have hardly heard of the replacement step, because in "standard" GAs, the number of generated offspring is equal to the number of parents, and all offspring simply replace all parents (generational replacement). The need for a graphical interface was hence felt necessary quite early in EASEA history.

Such a graphical interface was eventually developed as part of the DREAM European project IST-1999-12679 [2] (Distributed Resource Evolutionary Algorithm Machine —hence the name GUIDE), whose evolutionary library JEO [1] is one of the possible targets for EASEA compilation.

GUIDE: A Quick Tour

In a programming environment, the Graphical User Interface is the entry point at the highest possible level of interaction and abstraction. In the GUIDE/EASEA programming environment, the idea is that even a non-expert programmer should be able to program an EA using one of the underlying libraries without even knowing about it ! GUIDE must therefore at least allow the user to:

1. specify the numerous parameters of any evolutionary engine by way of a point-and-click interface,
2. write or view the user code related to problem-specific operators,
3. compile the experiment by a simple click,
4. run the experiment, and visualise the resulting outputs in some window(s).

While the second to fourth points above are merely a matter of implementation, and by no way could justify a scientific paper in a conference, the first point not only is original, but also clearly (graphically) highlights the common features of most existing EC paradigms: the user can specify *any* evolution engine within a single window.

The structure of GUIDE reflects this point of view, and offers four panels to the user (see the tags on Figure 1):

Algorithm Visualisation Panel to visualise and/or modify problem-dependent code. It contains a series of text windows, each window referring to the equivalent "sections" of EASEA source code [9] that the user has to type in. This is where the user writes the code for genetic operators such as initialisation, mutation, recombination, and most importantly for the evaluation function.

Evolution Engine Panel for the Darwinian components. This panel will be extensively described in section 4.

Distribution Control Panel to define the way different islands communicate in a distributed model [2]. This panel will soon be adapted to ParaDisEO [7], the recent Parallel Distributed version of EO.

Experiment Monitor Panel to compile and run the experiment. At the moment, only compilation is possible from there. The user has to run the program from outside GUIDE.

GUIDE obviously also offers (see Figure 1) a top-menu bar from where the user can save/load previous sessions to `Files`, choose the `Target Library` and `Build` the executable file —and, as usual, some of these actions can be fired by some icons from the toolbar.

4 Evolution Engine Panel

This section describes the most innovative feature of GUIDE: the panel where the user can specify the evolution engine, either from pre-defined "paradigmatic" engines, or by designing new combinations of selections and replacements fitting her/his taste.

4.1 Evolution Engines

The concept of evolution engine designates the two steps of the basic EA that implement some artificial Darwinism: **selection** and **replacement**. Basically, both steps involve the "selection" of some individuals among a population. However,

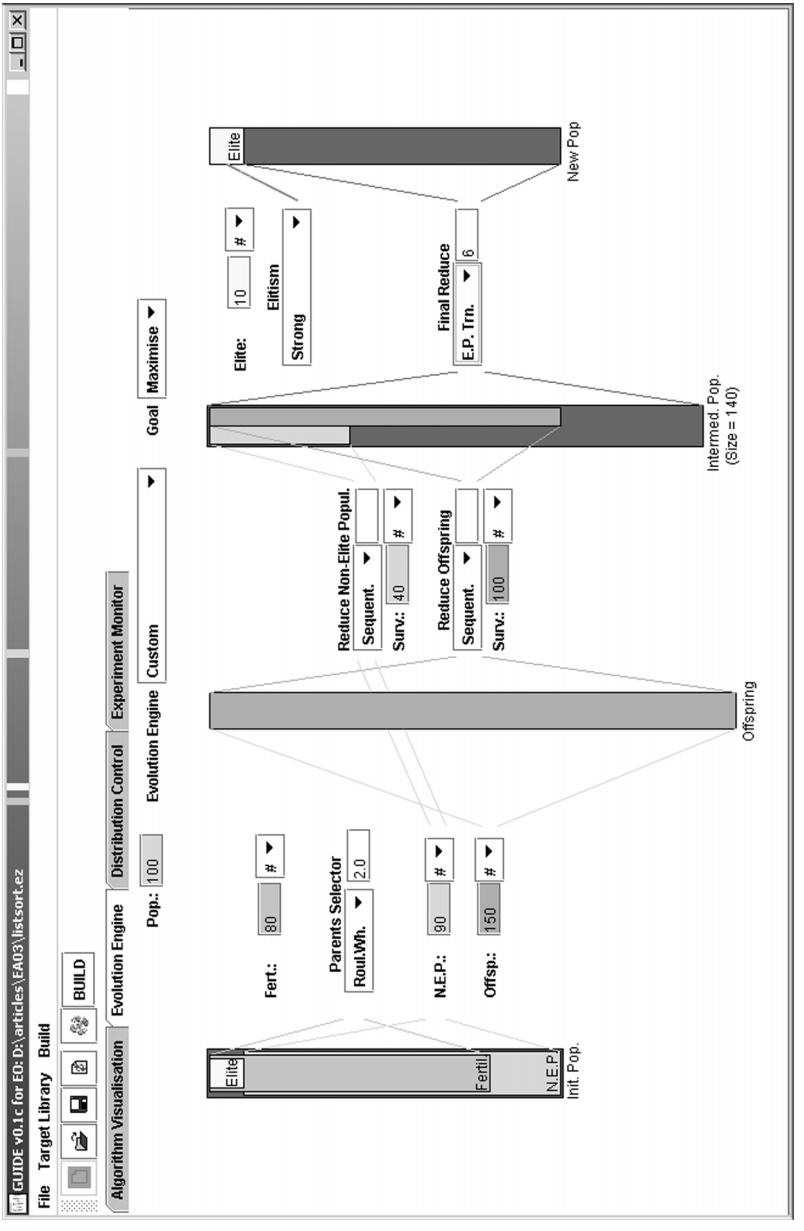


Fig. 1. GUIDE: Evolution Engine Panel

at least one important difference between those steps makes it necessary to distinguish among them, even when only one is actually used (like in traditional GAs or ESs): an individual can be selected *several times* for *reproduction* during the selection step, while it can be selected *at most once* during the *replacement* step (in this case, the non-selected individuals simply disappear).

Another important feature is implemented in a generic way in GUIDE: **elitism**. It will be discussed separately (section 4.2).

4.2 Panel Description

Figure 1 shows the Evolution Engine Panel of GUIDE. On top of the panel, the user can specify the population size (here 100), whether the fitness is to be maximised or minimised, and which type of Evolution Engine should be used: this type can be any of the existing known engines, described in next section 4.3, or set to `Custom`, as in Figure 1.

Below are some vertical bars, representing the number of individuals involved in different steps of a single generation: the left part of these bars describes the selection step (including, implicitly, the action of the variation operators, but not their description), and starts with the initial parent population —the leftmost bar, labeled `Init. Pop`. The right two bars specify the replacement step, and end with the `New Pop.` that will be the `Init. Pop.` of next generation —and hence has the same height (number of individuals contained) that the `Init. Pop.` bar.

Parameters input. There are two kinds of user-defined parameters that completely specify the evolution engine in GUIDE: the sizes of some intermediate populations, and the type of some selectors used inside the selection and the replacement steps. When a parameter is a type from a predefined list, a pull-out menu presents the possible choices to the user.

All sizes in GUIDE can be set either graphically, using the mouse to increase or decrease the size of the corresponding population, or using the numeric pad and entering the number directly. In the latter case, the number can be entered either as an absolute value, or as a percentage of the up-stream population size —the choice is determined by the small menu box with either the "#" or the "%" character.

Selection. The selection step picks up individuals from the parent population `Init. Pop.` and handles them to the variation operators to generate the `Offspring` population. In GUIDE, the parameters for the selection step are:

- the type of selector that will be used to pick up the parents, together with its parameters (if any). Available selectors (see e.g. [8] for the definitions)are `Roulette wheel` and (linear) `ranking` (and an associated selection pressure), `deterministic tournament` (and the associated tournament size), `stochastic tournament` (and the associated probability), plus the trivial `random` (uniform selection) and `sequential` (deterministic selection selecting from best to worse individuals in turn).

- The number of `Fertile` individuals: non-fertile individuals do not enter the selection process at all. This is somehow equivalent to truncation selection, though it can be performed here before another selector is applied among the fertile individuals only.
- The size of the `Offspring` population.

The last field in the "selection" area of the panel that has not been discussed here is the `N.E.P.`, or Non Elite Population, whose size is that of the population minus that of the `Elite` and that will be discussed in section 4.2 below. At the moment, consider that this population is the whole parent population.

In the example of Figure 1, only the 80 best individuals will undergo roulette wheel selection with selection pressure 2.0, and 150 offspring will be generated. (Although roulette wheel is known to have several weaknesses, notably compared to the Stochastic Universal Sampling described by Baker[4] it is still available in GUIDE, mainly because all underlying libraries implement it.)

Replacement. The goal of the replacement procedure is to choose which individuals from the parent population `Init. Pop.` and the offspring population `Offspring` will build the `New Pop.` population. In GUIDE, the replacement step is made of three **reduction** sub-steps: a reduction is simply the elimination of some individuals from one population according to some Darwinian **reduce** procedure:

- First, the `Non Elite Population` (the whole `Init. Pop.` in the absence of elitism) is reduced (the user must enter the type of reducer to be used, and the size of the reduced population).
- Second, the `Offspring` population is reduced (and again, the user must set the type of reducer and the size of the reduced population).
- Finally, both reduced populations above are merged into `Intermed.Pop.` (for intermediate population). The corresponding bar is vertically divided into two bars of different colors: the survivors of both populations. This population is in turn reduced into the final `New Pop.`, whose size has to be the size of the parent population: hence, only the type of reducer has to be set there.

The available types of reducers include again the `sequential` and `random` reductions, as well as the deterministic and stochastic `tournaments` (together with their respective parameters), that repeatedly eliminate bad individuals. An additional option is the `EP tournament` (together with the tournament size T): in this stochastic reducer, each individual fights against T uniformly chosen opponents, and scores 1 every time it is better; the best total scores then survive.

Note that many possible settings of those parameters would result in some unfeasible replacements (e.g. asking for a size of `Intermed. Pop.` smaller that the `Pop.` size). Such unfeasible settings are filtered out by GUIDE ... as much as possible.

In the example of Figure 1, the best 40 individuals from the `N.E.P.` (the initial population in the absence of elitism) are merged with the best 100 offspring, and the resulting intermediate population is reduced using an EP tournament of size 6.

Elitism. All features related to elitism have been left aside up to now, and will be addressed here. There are two ways to handle elitism in EC literature, termed "strong" and "weak elitism" in GUIDE. The user first chooses either one from the menu in the replacement section, and then sets the number of `Elite` parents in the corresponding input box (setting the `Elite` size to 0 turns off all elitism).

Strong elitism amounts to put some (copy) of the best initial parents in the `New Pop.` *before* the replacement step, without any selection against offspring whatsoever. The remaining "seats" in the `New Pop.` are then filled with the specified replacement. Note that the elite population nevertheless enters the selection together with the other parents —and that, of course, only the `N.E.P.` competes in the reduction leading to the parent part of the `Intermed. Pop`.

Weak elitism, on the other hand, only takes place *after* normal replacement, in the case when the best individual in the `New Pop.` is worse than the best parent of the `Init Pop`. In that case, all parents from the `Elite` population that are better than the best individual of the `New Pop.` replace the worst individuals in that final population. This type of elitism is generally used with an `Elite` size of 1.

For instance in Figure 1, elitism is set to `strong` and the number of elite parents to 10: the best 10 parents will anyway survive to next generation[2]. Only the 90 worse individuals undergo the reduction toward the `Intermed. Pop.` — the 40 best out of these 90 worse will survive this step— and only 90 seats are available in the `New Pop.`, the 10 first seats being already filled by the elite parents.

4.3 Specifying the Main Evolutionary Paradigms

The example in Figure 1 is typically a `custom` evolution engine. This section will give the parameter settings corresponding to the most popular existing evolution engines —namely GAs (both generational and Steady-State), ES and EP. Note that though those names correspond to the historical "dialects," they are used here to designate some particular combination of parameters, regardless of any other algorithmic component (such as genotype and variation operators). However, going back to those familiar engines by manually tuning the different parameters is rather tedious. This is the reason for the `Evolution Engine` pull-out menu on top of the panel: the user can specify in one click one of these five "standard" engines, and instantly see how this affects all the parameters.

[2] Such a strategy is generally used, together with a weak selection pressure, for instance when the fitness is very noisy, or is varying along time.

After choosing one of the pre-defined engines, the user can still modify all parameters. However, such modification will be monitored, and as soon as the resulting engine departs from the chosen one, the pull-out menu will automatically turn back to `Custom`. The predefined engines are:

Generational GA. In that very popular algorithm, let P be the population size. P parents are selected and give birth to P offspring that in turn replace the P parents: any selector is allowed, `Fertile` size is equal to `Pop.` size, `Offspring` size is set to `Pop.` size, reduced `N.E.P.` size to 0 (no parent should survive) and `Intermed. Pop.` size to `Pop.` size. In fact, none of the reducers is actually active in this setting (this is a generational replacement).

Furthermore, weak elitism can be set (generally with size 1). `Fertile` size can be reduced (this is equivalent to *truncation selection*) without leaving the `GGA` mode.

Steady-State GA. In Steady-State GA, a single offspring is created at each generation, and replaces one of the parent, usually chosen by tournament. Again, any selector is allowed, `Fertile` size is equal to `Pop.` size, but `Offspring` size is set to 1 and the parents are now reduced by some tournament reducer, to `Pop.` size minus one, while the final reducer is not active[3].

The number of offspring can be increased without leaving the `SSGA` mode: there is no clear limitation of this mode, though the number of offspring should be kept small w.r.t. `Pop.` size. And of course, setting any type of elitism here is a misconception.

Evolution Strategies. There are two popular evolution engines used in the traditional Evolution Strategies algorithms, the $(\mu, \lambda) - ES$ and the $(\mu + \lambda) - ES$: in both engines, there is no selection step, and all μ parents give birth to λ parents. The μ individuals of the new population are deterministically chosen from the λ offspring in the $(\mu, \lambda) - ES$ and from the μ parents *plus* the λ offspring in the $(\mu + \lambda) - ES$. Both these evolution engines set the selection to `Sequential`, the `Offspring` size and the `Reduce offspring` size to λ (no reduction takes place there) and the final reducer to `sequential`. Therefore, choosing $(\mu, \lambda) - ES$ sets the `Reduce N.E.P.` size to 0 while choosing the $(\mu + \lambda) - ES$ sets it to `Pop.` size.

The $(\mu + \lambda) - ES$ engine already is strongly elitist. The $(\mu, \lambda) - ES$ engine is not, and elitism can be added —but it then diverges from the original ES scheme.

Evolutionary Programming. In traditional EP algorithms, P parents generate P offspring (by mutation only, but this is not the point here). Then the P parents plus the P offspring compete for survival. Setting `EP` mode in the Evolution Engine menu sets the `Offspring` size to `Pop.` size, both reduced sizes for `N.E.P.` and `Offspring` to `Pop.` size as well (no reduction here) and the fi-

[3] An "age" tournament should be made available soon, as many SSGA-based algorithms do use age as the criterion for the choice of the parent to be replaced.

nal reducer to `EP tournament`. Note that early EP algorithms sometimes used a deterministic choice for the survivors —this can be achieved by choosing the `Sequential` final reducer.

Here again, elitism can be added, but this switches back to `Custom` engine.

5 Discussion and Perspectives

As already argued in section 2.2, the very first limitation of the GUIDE evolution engine comes from the chosen EA skeleton, that does not allow weird evolution engines. However, we firmly believe that most existing EA application use some evolution engine that falls in this framework.

The forthcoming improvements of this part of GUIDE are concerned with adding new selector/reducer options, more specifically selection procedures based on other criteria. The `Age tournament` has already been mentioned in the SSGA context. But all multi-objective selection procedures will also be added (with additional options in the `Maximise/Minmise` pull-out menu.

Going away from the evolution engine, the asymmetry in terms of flexibility between the Evolution Engine and the Algorithm Visualisation Panels cries out for a graphical interface allowing the user to specify the genome structure and the variation operators. Such interface is not as utopian as it might seem at first sight: the genome structure could be specified from basic types, and basic constructors (e.g. heterogeneous aggregations, homogenous vectors, linked lists, trees, ...). And there exist some generic ways to design variation operators for such structures [24].

Of course, last but not least, after the Distribution Panel has been adapted to ParaDisEO, the Experiment Monitor Panel should be redesigned such that the user can graphically plot the evolution of any variable in that window.

As a conclusion, the Evolution Engine Panel of GUIDE is only a first step towards a widely available dissemination tool for EAs. But it already achieves the demonstration that all evolutionary algorithms are born equal if the user is provided with enough parameters to tune. GUIDE Evolution Engine Panel is a visual and pedagogical tool that allows one to understand the intricacies of the different evolutionary paradigms. While, quite often, Graphical User Interfaces reduce flexibility, GUIDE offers at the Evolution Engine level readability, simplicity of use and an easy way to experiment with complex parameters.

References

1. M.G. Arenas, P. A. Castillo, B. Dolin, I. Fdez de Viana, J. J. Merelo, and M. Schoenauer. JEO: Java Evolving Objects. In *GECCO'02*, pp 991–994, 2002.
2. M.G. Arenas, P. Collet, A.E. Eiben, M. Jelasity, J.J. Merelo, B. Paechter, M. Preuß, and M. Schoenauer. DREAM: A Framework for Distributed Evolutionary Algorithms. In J.J. Merelo et al., eds., *PPSN VII*, pp 665–675. Springer-Verlag, LNCS 2439, 2002.
3. T. Bäck and M. Schütz. Intelligent mutation rate control in canonical GAs. In Z. W. Ras and M. Michalewicz, eds, *ISMIS '96*, pp 158–167. Springer Verlag, 1996.

4. J. E. Baker, Reducing bias and inefficiency in the selection algorithm. Proceedings of the Second International Conference on Genetic Algorithms, L. Erlbaum Assoc. Eds (Hillsdale), 1987.
5. Th. Bäck, D.B. Fogel, and Z. Michalewicz, eds. *Handbook of Evolutionary Computation*. Oxford University Press, 1997.
6. P. J. Bentley and S.Kumar. Three ways to grow designs: A comparison of embryogenies for an evolutionary design problem. In *GECCO'99*, pp.35-43, 1999.
7. S. Cahon, N. Melab, E-G. Talbi, and M. Schoenauer. ParaDisEO-based design of parallel and distributed evolutionary algorithms, this volume, 2003.
8. U. Chakraborty, K. Deb, and M. Chakraborty. Analysis of selection algorithms: A Markov chain approach. *Evolutionary Computation*, 4(2):133–168, 1996.
9. P. Collet, E. Lutton, M. Schoenauer, and J. Louchet. Take it EASEA. In M. Schoenauer et al., eds, *PPSN VI*, LNCS 1917, pp 891–901. Springer Verlag, 2000.
10. D.B. Fogel. *Evolutionary Computing: The Fossile Record*. IEEE Press, 1998.
11. L. J. Fogel, A. J. Owens, and M. J. Walsh. *Artificial Intelligence through Simulated Evolution*. New York: John Wiley, 1966.
12. D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison Wesley, 1989.
13. Michael Herdy. Self-adaptive population size and stepsize in combinatorial optimization problems:solving magic squares as an example. In *Proc. GECCO-2002 Workshops*. Morgan Kaufmann, 2002.
14. J. H. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, 1975.
15. M. Keijzer, J. J. Merelo, G. Romero, and M. Schoenauer. Evolving Objects: a general purpose evolutionary computation library. In P. Collet et al., eds, *Artificial Evolution'01*, pp 229–241. Springer Verlag, LNCS 2310, 2002.
16. J. R. Koza. *Genetic Programming: On the Programming of Computers by means of Natural Evolution*. MIT Press, Massachusetts, 1992.
17. Peter Nordin and Wolfgang Banzhaf. Evolving turing-complete programs for a register machine with self-modifying code. In L. J. Eshelman, ed., *ICGA'95*, pp 318–325. Morgan Kaufmann, 15-19 1995.
18. N. J. Radcliffe. Forma analysis and random respectful recombination. In R. K. Belew and L. B. Booker, eds, *ICGA'91*, pp 222–229. Morgan Kaufmann, 1991.
19. N. J. Radcliffe and P. D. Surry. Fitness variance of formae and performance prediction. In L. D. Whitley and M. D. Vose, eds, *Foundations of Genetic Algorithms 3*, pp 51–72. Morgan Kaufmann, 1995.
20. I. Rechenberg. *Evolutionstrategie: Optimierung Technisher Systeme nach Prinzipien der Biologischen Evolution*. Fromman-Hozlboog Verlag, Stuttgart, 1973.
21. B. Sareni and L. Krähenbühl. Fitness sharing and niching methods revisited. *Transactions on Evolutionary Computation*, 2(3):97–106, 1998.
22. H.-P. Schwefel. *Numerical Optimization of Computer Models*. John Wiley & Sons, New-York, 1981. 1995 – 2^{nd} edition.
23. P.D. Surry. *A Prescriptive Formalism for Constructing Domain-specific Evolutionary Algorithms*. PhD thesis, University of Edinburgh, 1998.
24. P.D. Surry and N.J. Radcliffe. Formal algorithms + formal representations = search strategies. In H.-M. Voigt et al., eds., *PPSN IV*, LNCS 1141, pp 366–375. Springer Verlag, 1996.
25. M. Wall. Overview Matthew's Genetic Library. http://lancet.mit.edu/ga/.
26. D. Whitley, S. Rana, and R. Heckendorn. Representation issues in neighborhood search and evolutionary algorithms. In D. Quadraglia et al., eds., *Genetic Algorithms and Evolution Strategies in Engineering and Computer Sciences*, pp 39–58. John Wiley, 1997.

ParaDisEO-Based Design of Parallel and Distributed Evolutionary Algorithms

S. Cahon[1], N. Melab[1], E.-G. Talbi[1], and M. Schoenauer[2]

[1] Laboratoire d'Informatique Fondamentale de Lille
Université des Sciences et Technologies de Lille
59655 - Villeneuve d'Ascq Cedex
[2] Projet Fractales – INRIA Rocquencourt – 78153 Le Chesnay Cedex
{cahon, melab, talbi}@lifl.fr, marc.schoenauer@inria.fr

Abstract. ParaDisEO is a framework dedicated to the design of parallel and distributed metaheuristics including local search methods and evolutionary algorithms. This paper focuses on the latter aspect. We present the three parallel and distributed models implemented in ParaDisEO and show how these can be exploited in a user-friendly, flexible and transparent way. These models can be deployed on distributed memory machines as well as on shared memory multi-processors, taking advantage of the shared memory in the latter case. In addition, we illustrate the instantiation of the models through two applications demonstrating the efficiency and robustness of the framework.

1 Introduction

There are (at least) two broad categories of optimization problems that requires heavy computational resources: large combinatorial problems, and complex numerical engineering problems.

Large combinatorial problems are continuously evolving in terms of requirements, constraints, etc. Therefore, one needs flexible and adaptable algorithms to solve them. Furthermore, these problems are often NP-hard, characterized by a complex landscape and large instances with many decision variables.

Combinatorial optimization algorithms fall into two categories: exact methods and metaheuristics. Exact methods allow to find optimal solutions but they are often inefficient and unpractical. On the opposite, the metaheuristics aim at finding efficiently near-optimal solutions. Metaheuristics include Evolutionary Algorithms (EAs), local search methods, and the like.

Similarly, the advances of both modelization techniques and numerical simulation algorithms have increased the demand for numerical optimization algorithms. And the practitioner is faced with the following dilemma: either use some deterministic method to precisely optimize a simplified model, or use a stochastic method to approximately optimize a complex model. Evolutionary Algorithms are a good choice in the latter situation.

Unfortunately, EAs applied to real-world problems, either large combinatorial problem or complex numerical optimization applications, are known to

P. Liardet et al. (Eds.): EA 2003, LNCS 2936, pp. 216–228, 2004.

be time consuming. On the other hand, the proliferation of networks/clusters of workstations and parallel machines offers more and more facilities to the users: EAs are also known to allow efficient Parallelization/distribution and multi-threading, achieving high performance and robustness in reasonable time ... at the cost of a sometimes painful apprenticeship of parallelization techniques.

Many implementations of EAs have been proposed and some of them are available on the Web. Code developers are often tempted to reuse them to save time. However, understanding and reusing a third party code is generally a long, tedious and error prone task. Indeed, one needs to examine the internal working of the code and make several modifications to adapt it to a new problem. A better reuse approach consists in using a framework, such as ParaDisEO [4] or MALLBA [1], dedicated to the design and development of EAs. Good frameworks should be well documented and tested. But their success strongly depends on their ability to allow flexible and adaptable design. For instance, adaptation should simply require to parametrize some existing EAs and at least to write the fitness function. The same is true for the parallelization/distribution of EAs: In order to facilitate this step for those who are unfamiliar with parallel mechanisms, the frameworks must integrate the up-to-date parallelization techniques and allow their exploitation and deployment in a transparent manner.

This paper focuses on the ParaDisEO framework [1], a flexible approach for the transparent design and deployment of parallel and distributed evolutionary algorithms. ParaDisEO is basically an extension of Evolving Objects (EO) [9], an open source framework based on C++ templates allowing a flexible design of EAs. The extensions include local search methods (descent search, simulated annealing and tabu search for combinatorial optimization, gradient-based search for numerical optimization), hybridization mechanisms (coupling local search and global evolutionary search) and parallel/distributed models. This paper will concentrate on the latter topic: the different models supported by ParaDisEO will be presented, together with their flexible and transparent parameterization. The user is relieved from the burden to explicitly manage the communication and concurrency. Furthermore, the models can be efficiently deployed both on shared memory multi-processors and on distributed networks.

The reader is referred to [2] for a state of the art about parallel and distributed evolutionary algorithms, their design, many techniques of hybridization, and a full taxonomy of applications done until now. In this paper, the use of ParaDisEO will be illustrated on two test cases: the spectroscopic data mining [10] and the network design optimization [11]. The results show that they allow an efficient and robust deployment.

[1] This work is a part of a current French joint grid computing project ACI-GRID DOC-G (Challenges in Combinatorial Optimization on Grids)

The paper is organized as follows: Section 2 highlights the major requirements for the design of a useful framework of EAs and rapidly surveys existing frameworks freely available on the Web. Section 3 presents the ParaDisEO framework and details the major parallel/distributed supported models. In Section 4, we identify the main issues regarding the implementation and deployment of the models on shared memory multi-processors and parallel/distributed machines. Section 5 is dedicated to experimentation and evaluation of the models through the two applications quoted above. In Section 6, we conclude with some perspectives of this work.

2 Design Requirements and Frameworks

Using a framework can be successful only if some important user criteria are satisfied. The major of them regarding the previous problem statement are the following:

- **Ease of use**: the framework has to be relatively user-friendly. However, this obviously depends on the skill level of the programmer.
- **Flexibility/adaptability**: the integrated evolutionary algorithms can be adapted to a large variety of problems by just parameterizing or specializing them.
- **Openness**: the platform must allow the design and integration of new algorithms by reusing (parts of) existing ones.
- **Portability**: in order to satisfy a large number of users the framework must support different material architectures and their associated operating systems.
- **Performance and robustness**: as the optimization applications are often time-consuming the performance issue is crucial. Moreover, the execution of the algorithms must be robust to guarantee the reliability of the results.

Many frameworks have been realized in the combinatorial/numerical optimization area tempting to meet these design requirements. However, very often only some of them are deeply developed. We present below a non-exhaustive list of existant open source frameworks. They are in some way relatively outstanding to the best of our knowledge.

- **The MALLBA Project**[1]: The library is a set of common *software skeletons* including both metaheuristics and exact methods. Skeletons are generic classes allowing ease of use in some way and flexible design. Robustness is achieved by strong and weak hybridization mechanisms. The performance issue is addressed by providing parallel models deployable on LAN/WAN environments. Communications are based on *NetStream*, a flexible and simple OOP message passing service, upon the Message Passing Interface. Portability is ensured by the utilization of the C++ language and standards such as MPI.

- **The DREAM Project**[3]: The software infrastructure is devoted to support infohabitants evolving in an open and scalable way. It considers a virtual pool of distributed computing resources, where the different steps of an E.A. are automatically processed. It is coupled with the GUIDE, EASEA, and JEO tools. The integrated architecture allows the user to choose his/her specification interface according to his/her skill level. The main focus of DREAM is the ease-of-use and flexibility. Portability is enabled by the use of the Java language. It is drastically limited regarding the performance and robustness requirements.
- **ECJ** (`http://www.cs.umd.edu/projects/plus/ec/ecj/`): EJC is a rich Java-based portable evolutionary computation and genetic programming library. It is highly flexible as nearly all the classes are dynamically determined at runtime by a user-provided parameter file. In addition, all structures are arranged to be easily modifiable. Furthermore, parallelism and multi-threading allow efficient executions. Many other features are present including multi-objective optimization, checkpointing possibilities, etc. However, local search methods and hybridization mechanisms are not addressed.
- **JDEAL** (`http://laseeb.ist.utl.pt/sw/jdeal/`): JDEAL is another portable Java framework for E.As. Both local and parallel/distributed models are allowed to deal with the performance requirement. Other features include the paradigm freedom, facilities for the reuse and extension of existing code, and the being of both a documentation and a tutorial. Therefore, the ease-of-use and flexibility can be in some way achieved. However, as JDEAL local search methods and hybridization mechanisms are not addressed.
- **GALOPPS**[8]: The GALOPPS system is only dedicated to the design of serial or parallel genetic algorithms. A PVM extension allows the deployment of some cooperative island G.A. Moreover, some techniques of checkpointing/restarting are implemented so that a trace could be stored during the execution path. It is very limited regarding the quoted design requirements.

3 The ParaDisEO Framework

The design factors identified in the previous section are our main objectives in the design of EO/ParaDisEO. In order to meet them the following choices have been made:

- **Object-Oriented technology**: the evolutionary algorithms that are provided are software skeletons. They are implemented as templates allowing to factor out the common behaviors of evolutionary algorithms in generic classes, to be instantiated by the user. The object-oriented technology is important to meet the three first requirements.
- **Transparent parallelism and distribution**: parallelism and distribution are two important ways to achieve high performance execution. In ParaDisEO, parallelism and distribution are implemented so that the user can for instance add parallelism to his/her sequential algorithm very easily and transparently.

– **Hybridization**: the hybridization paradigm allows one to obtain robust and better solutions. Here again, the OO technology facilitates its design and implementation.

The "EO" part of ParaDisEO means Evolving Objects. Before ParaDisEO feature are presented let us give a brief description of EO.

3.1 EO

EO is a C++ open source framework downloadable from `http://eodev.sourceforge.net`. The framework is originally the result of an European joint work [9]. EO includes a paradigm-free Evolutionary Computation library (EOlib) dedicated to the flexible design of EAs through evolving objects superseding the most common dialects (Genetic Algorithms, Evolution Strategies, Evolutionary Programming and Genetic Programming). Flexibility is enabled through the use of the object-oriented technology. Templates are used to model the EA features: coding structures, transformation operators, stopping criteria, etc. These templates can be instantiated by the user according to his/her problem-dependent parameters. The OO mechanisms such as inheritance, polymorphism, ... are powerful ways to design new algorithms or evolve existing ones. Furthermore, EO integrates several services making it easier to use including visualization facilities, on-line definition of parameters, application checkpointing, etc.

Moreover, a Graphical User Interface has been developed for EO. This GUI, called *GUIDE* (*Graphic User Interface for DREAM Experiments*), and based on the specification language for EAs *EASEA* [5]. *GUIDE* was developed during the European project *DREAM* [3], and provides a friendly graphical interface allowing the user to specify his/her own EA. One panel is dedicated to the specification of the problem-specific parts of the algorithm (the genome, and the corresponding operators). Another panel is devoted to the specification of the evolution engine (the implementation of artificial Darwinism). And a third panel is entirely devoted to specifying the distribution mechanism: as it was initially conceived for the DREAM framework, this panel only knows about the island model (see section 1). But within this model, the distribution panel of GUIDE allows the user to specify a topology, a number of migrants, selection and replacement mechanisms for the migrants, ... i.e. all the parameters needed for the ParaDisEO island model.

3.2 ParaDisEO

In its original version, EO does not enable the design of local search methods such as descent search or hill-climbing, simulated annealing, tabu search or gradient-based search. Moreover, it does not offer any facility for either parallelism and distribution, or hybridization. Sticking out those limitations was the main objective in the design and development of ParaDisEO. In the following, we will focus only on parallel/distributed mechanisms.

Note, however, as far as hybridization is concerned, that the two levels (High and Low) and the two modes (Relay and Teamwork) of hybridization identified in [13] are available in ParaDisEO: High level hybridization consists of making self-contained EAs cooperate. Their internal work is not considered, thus their coupling is weak. At the contrary, the Low level addresses the functional composition of an EA. For example, a transformation operator can be replaced by a local search method. On the other hand, the Relay mode means that a set of EAs are applied in a pipelined way. The Teamwork mode allows a concurrent cooperation between EAs.

Going back to parallelism, three major parallel models are implemented in the platform: the island asynchronous cooperative model, the parallel/distributed population evaluation and the distributed evaluation of a single solution. These models will now be detailed in turn (again, detailed descriptions ot those models as well as a comprehensive survey of existing models is given in [1]).

3.3 Model 1 – Island Asynchronous Cooperative Model

Different EAs are simultaneously deployed and then cooperate to compute better and robust solutions (fig. 1). They exchange in an asynchronous way genetic stuff to diversify the search. The objective is to allow to delay the global convergence, especially when the EAs are heterogeneous regarding the variation operators. The migration of individuals follows a policy defined by few parameters: the migration decision criterion, the exchange topology, the number of emigrants, the emigrants selection policy, and the replacement/integration policy.

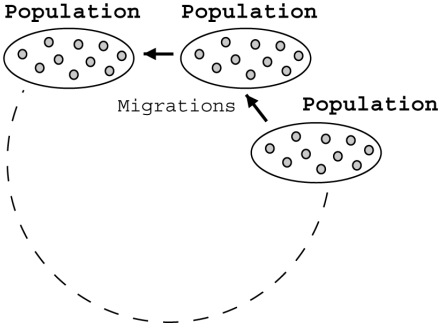


Fig. 1. The cooperative island evolutionary algorithm

– **Migration decision criterion**: Migration can be decided according to either *blind* or *intelligent* criterion. The blind can be either periodic or probabilistic. Periodic decision making consists in periodically performing the migration by each EA. The user has to set the value of the migration frequency. Probabilistic migration is performed at each generation with a user-defined probability. On the other hand, the criteria for the intelligent approaches are here driven by some quality improvement criterion. The user has to specify

a threshold for the improvement, and migration will take place every time the improvement of the best fitness in the population during two successive generations will be below that threshold.
As in the rest of EO, all the quoted migration criteria are predefined in ParaDisEO as class methods. Hence the user can easily either use them directly, or combine them as building blocks to design his/her own specific criterion.

- **Exchange topology**: The topology specifies for each island its neighbors with respect to migration, i.e. the other islands to which it will send its emigrants, and the ones from which it will receive immigrants (both lists can be different. Two well-known topologies are predefined: ring and hypercube. And of course, the user has the possibility to specify his/her own topology according to problem-dependent or machine-dependent features.
- **Number of emigrants**: This emigrants parameter is defined either as a percentage of the population size, or as a fixed number of individuals.
- **Emigrants selection policy**: The selection policy indicates in a deterministic or stochastic way how to select emigrant individuals from the population of a source EA. Different selection policies are already defined in EO, and used for instance to select the parents undergoing variation operators (e.g. roulette wheel, ranking, stochastic or deterministic tournaments, uniform sampling, ...). The flexibility of the library makes it easy to reuse these policies for emigrants selection.
- **Replacement/integration policy**: Symmetrically, the replacement policy defines in a deterministic or stochastic way how to integrate the immigrant individuals in the population. Again, the replacement strategies defined in EO are used here, including tournaments, EP-like stochastic replacement, elitist and pure random replacements.

3.4 Model 2 – Parallel/Distributed Population Evaluation

The parallelization evaluation step of an EA is required as it is in general the most time-consuming. The parallel evaluation follows the centralized model (fig. 2). The farmer applies the following operations: selection, transformation and replacement as they require a global management of the population. At each generation, it distributes the set of new solutions between different workers. These evaluate and return back the solutions and their quality values. An efficient execution is often obtained particularly when the evaluation cost of each solution is costly.

The model can be deployed either on a shared-memory multi-processor or a distributed memory machine. In the first case, the user has to indicate the number of parallel evaluators (workers). The different workers pick up individuals from a shared list-based population. In the distributed model the user must give the size work unit size (grain). The parameter represents the number of individuals to be sent to a worker at a time. At each generation an evaluator may to process one or several work units.

The model described above is the synchronous master/slave model [2]. The two main advantages of the asynchronous model over the synchronous model are first the fault tolerance of the asynchronous model, an second the robustness in case the fitness evaluation can take very different computation times (e.g. for nonlinear numerical optimization). Whereas some time-out detection can be used to address the former issue, the latter one can be partially overcome if the grain is set to very small values, as individuals will be sent out for evaluations upon request of the slaves.

The asynchronous evaluation process will be soon released in ParaDisEO, so that both breeding and evaluation step could be done concurrently. The farmer manages the evolution engine and two queues of individuals, each one with a given fixed size: individuals to be evaluated, and awaiting solutions being evaluated. The first ones wait for a free evaluating node. When the queue is full the process blocks. The second ones are assimilated into the population as soon as possible.

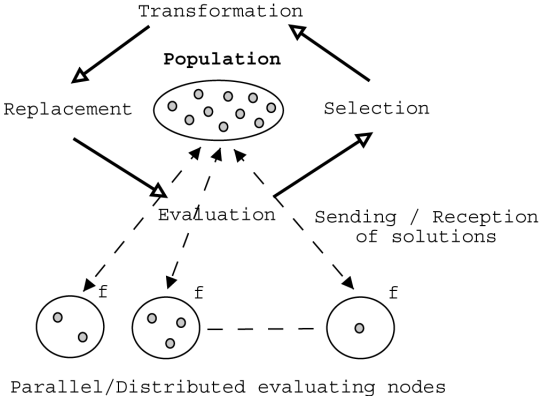


Fig. 2. The parallel/distributed evaluation of a population

3.5 Model 3 – Distributed Evaluation of a Single Solution

The quality of each solution is evaluated in a parallel centralized way. That model is particularly interesting when the evaluation function can be itself parallelized as CPU time-consuming and/or IO intensive. In that case, the function can be viewed as an aggregation of a certain number of partial functions. For instance, if the problem to be solved is multi-objective each partial function evaluates one objective. The partial functions could also be identical if for example the problem to deal with is a data mining one. The evaluation is thus data parallel and the accesses to data base are performed in parallel. Furthermore, a reduction operation is performed on the results returned by the partial functions. As a summary, for this model the user has to indicate a set of partial functions and an aggregation operator of these (fig. 3).

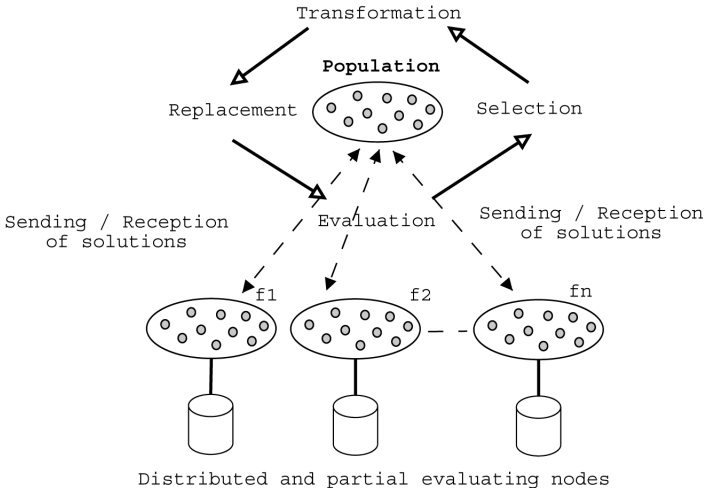


Fig. 3. The distributed evaluation of a single solution

4 Implementation and Transparent Deployment

One has to note here that the implementation and deployment of the presented parallel models is transparent for the user. Indeed, the user does not need to manage the communications and threads-based concurrency. We present below some implementation and deployment issues with regards to the underlying parallel hardware support: shared memory multi-processors and distributed memory machines.

– The migration decision maker represents the kernel of the island asynchronous cooperative model. It uses the user-defined parameters to take migration decisions, perform migrations according to the selection and replacement policies. The implemented migration algorithm is the following:

```
1. If there are any waiting immigrants integrate them into the
local population, using the replacement operator.
2. Process the received immigration requests if any. For each
request, choose some individuals according to the selection
operator, then send them as requested.
3. Check if according to the migration decision maker a
migration operation is necessary. If so participate to the
migration process.
```

The deployment of the algorithm requires to choose the technical means to be used to perform the migrations of the individuals according to the hardware support. On shared memory multi-processors the implementation is multi-threaded and the migrations are memory copy-driven. Threads access in a concurrent way the populations of the different EAs stored on the shared

memory. On distributed memory machines the mechanism is communication-driven. Migrations are performed by explicit message exchange.
- In the parallel/distributed population evaluation model the population is always stored on a single processor. However, the assignment of the individuals to the processors for evaluation depends on the hardware architecture. On shared memory multi-processors threads pick up (copy) individuals from the global list in a concurrent way. On distributed memory processors the population is divided into sub-populations according to the user-defined granularity parameter. These sub-populations are then assigned to the processors using communication messages. When an evaluator returns its results to the farmer, that evaluator will immediately be sent another sub-population – if there still are individuals to be evaluated. The performance of the model hence strongly depends on the grain size. Very fine-grained sub-populations induce a huge amount of communication, leading to a small acceleration over sequential programs. Conversely, very coarse-grained sub-populations imply a lower degree of parallelism, and thus again leads to a smaller acceleration. Some compromise has to be found between both extreme cases.
- The distributed evaluation of a single solution model is well-suited for distributed memory machines. The data of the evaluation function have to be distributed among different processors. A distribution strategy has to be defined. One has to note here that the problem is application-dependent. Up to date, in ParaDisEO it is supposed that the data is already distributed. The computation i.e. the user-specified partial functions are sent in explicit messages to the workers which apply them on their local data.

Finally, to meet the portability objective the implementation of the parallel/distributed and hybridization mechanisms is based on different standards. Indeed, the *de facto* communication libraries PVM and MPI allow a portable deployment on networks of heterogeneous workstations. Furthermore, the Posix Threads multi-programming library contributes to enable a portable execution on shared memory multi-processors.

5 Applications

ParaDisEO has been experimented with several applications including academic ones such as TSP and graph coloring, and industrial ones. In this section, we will address two real-world problems: the NIR spectroscopic data mining and the mobile telecommunication network design.

- **Near InfraRed spectroscopic data mining**: the problem consists in discovering, from a set of data samples, a predictive mathematical model for the concentration of sugar in beet. Each data sample contains a measure of the concentration of sugar in one beet sample, and a set of its absorbances to 1024 NIR wavelengths. According to the Beer Lambert law the problem is linear. The Partial Least Square (PLS) statistical method is known to be well-suited to deal with such problem. However, the number of wavelengths

in the resulting predictive model is high, decreasing the understandability of the results. Well, but the analysis of the data allow to highlight that many absorbances are correlated between them (redundancy problem) and less correlated with the concentration (irrelevance problem). In order to withdraw both irrelevant and redundant wavelengths a feature selection has to be performed. As feature selection is an NP-hard problem we use a genetic algorithm (GA) to solve it. The PLS method is used as the fitness function of the GA. More exactly, the prediction error returned by the PLS method is the fitness value of the individuals (selections of wavelengths).
The hybridization GA-PLS is CPU time consuming. The fitness evaluation is particularly costly as the PLS method handles large matrices. Parallelism is required to get results at short notice. The models 1 and 2 have been exploited in a straightforward way. The experimentation has been done on an IBM cluster of SMP. The model is composed of four distributed islands, each of them deployed on a shared memory multi-processor (10 processors) dedicated to evaluate selections of wavelengths. The database is replicated on each cluster node and shared between the processes of the same node. Thus, four populations of 200 individuals are evolving and migrations occur when no improvement has been noticed during the last ten iterations. Both selection and replacement operators are performed by using the stochastic tournament operator with a rate pressure fixed to 0.8.
The obtained results are encouraging since over 90% of wavelengths proved to be useless or harmful. Therefore, the processing time required to compute the concentration of sugar in a new sample would be divided by ten. In addition, the prediction accuracy is increased by 37% compared to the PLS without feature selection.

– **Mobile telecommunications network design**: One of the major problems in the engineering of mobile telecommunication networks is the design of the network. It consists in positioning base stations on potential sites in order to fulfill some objectives and constraints [11]. More exactly, the multi-objective problem is to find a set of sites for antennas from a set of pre-defined candidates sites, to determine the type and the number of antennas, and their configuration parameters (tilt, azimuth, power, ...).
It is a hard practical problem with many decision variables. Tackling it in a reasonable time requires to use of parallel heuristics. A parallel distributed GA is investigated to solve the problem. The three parallel models have been exploited. The islands in the model 1 evolve with heterogeneous variation operators. Migrations are performed according to a ring topology. Emigrants are randomly selected with a fixed percentage of 5% of the population. In the model 2, the offspring are distributed with a chunk fixed to one. Furthermore, the fitness evaluation of a network is not trivial as it handles large precalculated databases. Each node contributing in the deployment of the model 3 is assigned a subpart of the geographical map.

Experimentations on the hardware platform quoted above have shown that the parallel models are efficient. Indeed, they allow to obtain robust and high

quality solutions. In addition, as Figure 4 illustrates it they permit a quasi-linear parallel execution speed-up for the data mining application. For the network design application, the speed-up is quasi-linear on over 16 processors. However, the scalability is not achieved as the granularity i.e. (a subpart of the geographical map) of parallelism decreases while increasing the number of processors. The processing time required by the aggregation of the partial results widely exceeds the computation load performed by each participant processor.

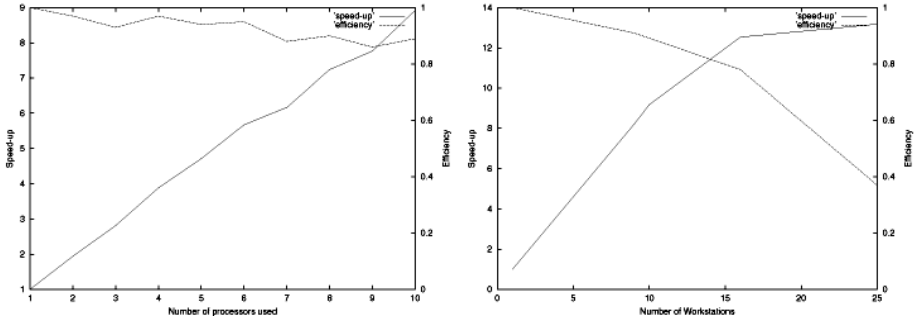


Fig. 4. Here are the measures of speed-up and efficiency obtained for the two described applications according to the respective number of processors/workstations used

6 Conclusion and Future Work

We have presented the ParaDisEO approach for the flexible and transparent design of parallel and distributed evolutionary algorithms. Flexibility and transparency are achieved through parameterizable templates. Three major complementary models have been presented. Their implementation is based on two communication libraries and the Posix threads multi-programming library. These standards allow a portable execution of the models and their deployment on as well shared memory multi-processors as distributed memory machines.

As many problems are multi-objective in practice we are currently investigating the ParaDisEO-based design of multi-objective EAs. As the framework is flexible the design is straightforward. Moreover, the models 1 and 2 do not require any design change. The model 3 could be exploited by evaluating in parallel the different objectives of the fitness function. In addition, we are addressing some issues related to the deployment of the three models in adaptive environments. As workstations are continuously leaving and joining the network, ParaDisEO must provide dynamic task stopping and restarting mechanisms. Furthermore, it has to integrate checkpointing in order to deal with the fault tolerance issue. To do that we are investigating a new version of ParaDisEO on top of the Condor-PVM [12] resource manager. Finally, to allow a deployment on grid-based environments we focus on grid-oriented algorithmics and implementation

issues. To meet that objective we are planning to use Condor-G [7]. The latter grid execution support is a coupling of Condor and Globus [6], a toolkit that allows a large scale multi-domain execution of parallel and distributed applications.

References

1. E. Alba and the MALLBA Group. MALLBA: A library of skeletons for combinatorial optimisation. In R.F.B. Monien, editor, *Proceedings of the Euro-Par*, volume 2400 of LNCS, pages 927–932. Springer-Verlag, 2002.
2. E. Alba and M. Tomassini. Parallelism and Evolutionary Algorithms. *IEEE Transactions on Evolutionary Computation*, 6(5):443–462, October 2002.
3. M.G. Arenas, P. Collet, A.E. Eiben, M. Jelasity, J.J. Merelo, B. Paechter, M. Preuß, and M. Schoenauer. A framework for distributed evolutionary algorithms. In *Proceedings of PPSN VII*, september 2002.
4. S. Cahon, E-G. Talbi, and N. Melab. ParadisEO: A Framework for Parallel and Distributed Biologically Inspired Heuristics. In *Nature Inspired Distributed Computing Workshop, in IEEE IPDPS2003 (Int. Parallel and Distributed Processing Symposium)*, page 201. Nice, France, IEEE Press, April 2003.
5. P. Collet, E. Lutton, M. Schoenauer, and J. Louchet. Take it easea. In PPSN2000editors, editor, *PPSN2000*, LNCS 1917, pages 891–901, 2000. http://sourceforge.net/projects/easea.
6. I. Foster and C. Kesselman. Globus: A Metacomputing Infrastructure Toolkit. *ntl J. Supercomputer Applications*, 11(2):115–128, 1997.
7. J. Frey, T. Tannenbaum, M. Livny, I. Foster, and S. Tuecke. Condor-G: A Computation Management Agent for Multi-Institutional Grids. *Cluster Computing*, 5(3):237–246, 2002.
8. E. Goodman. An introduction to galopps – the "genetic algorithm optimized for portability and parallelism" system. Technical report, Intelligent Systems Laboratory and Case Center for Computer-Aided Engineering and Manufacturing, Michigan State University, November 1994.
9. M. Keijzer, J.J. Merelo, G. Romero, and M. Schoenauer. Evolving Objects: A General Purpose Evolutionary Computation Library. *Proc. of the* 5^{th} *Intl. Conf. on Artificial Evolution (EA'01), Le Creusot, France*, October 2001.
10. N. Melab, S. Cahon, E-G. Talbi, and L. Duponchel. Parallel genetic algorithm based wrapper feature selection for spectroscopic data mining. In *BioSP3 Workshop on Biologically Inspired Solutions to Parallel Processing Problems, in IEEE IPDPS2002 (Int. Parallel and Distributed Processing Symposium)*, page 201. Fort-Lauderdale, USA, IEEE Press, April 2002.
11. H. Meunier, El-Ghazali Talbi, and P. Reininger. A Multiobjective Genetic Algorithm for Radio Network Optimization. In *Congress on Evolutionary Computation*, volume 1, pages 317–324. IEEE Service Center, July 2000.
12. J. Pruyne and M. Livny. Interfacing condor and pvm to harness the cycles of workstation clusters. In *Future Generations of Computer Systems*. P. Sloot, to appear.
13. E-G. Talbi. A Taxonomy of Hybrid Metaheuristics. *Journal of Heuristics, Kluwer Academic Publishers*, Vol.8:541–564, 2002.

A Coarse-Grained Parallel Genetic Algorithm Employing Cluster Analysis for Multi-modal Numerical Optimisation

Yong Yang[1], Jonathan Vincent[2], and Guy Littlefair[1]

[1] Faculty of Technology, Southampton Institute.
East Park Terrace, Southampton, SO14 OYN, UK.
[2] School of Design, Engineering and Computing, Bournemouth University.
Talbot Campus, Poole, Dorset, BH12 5BB, UK.

Abstract. This paper describes a technique for improving the performance of parallel genetic algorithms on multi-modal numerical optimisation problems. It employs a cluster analysis algorithm to identify regions of the search space in which more than one sub-population is sampling. Overlapping clusters are merged in one sub-population whilst a simple derating function is applied to samples in all other sub-populations to discourage them from further sampling in that region. This approach leads to a better distribution of the search effort across multiple sub-populations and helps to prevent premature convergence. On the test problems used, significant performance improvements over the traditional island model implementation are realised.

1 Introduction

Genetic algorithms (GAs) [1] are an established technique for the solution of complex optimisation problems. They are not guaranteed to find an optimal solution and their effectiveness is determined largely by the population size [1]. As the population size increases, the GA has a better chance of finding the globally optimum solution, or other acceptable solution, but the computational cost also increases as a function of the population size. As the scale and complexity of target applications increases, run-time becomes a major inhibitor. Parallel genetic algorithms (PGAs) have therefore become an important area of research.

Adopting the classification of Alba and Troya [2], four common classes of parallel genetic algorithm (PGA) can be identified; global, diffusion, island, and hybrid. The primary purpose of these models is the effective distribution of the time consuming objective function evaluations across a number of physical processing nodes. Considerable progress has been made in understanding the mechanics and performance implications of these techniques, see for example [3]. This paper considers coarse-grained parallelism in the context of numerical optimisation. Of the four classes of parallel genetic algorithm, only the island model is specifically targeted at coarse-grained processing platforms.

The island model PGA is based on a division of the genetic algorithm's population into a number of co-operating sub-populations, each capable of being

P. Liardet et al. (Eds.): EA 2003, LNCS 2936, pp. 229–240, 2004.

executed on a separate processing node. These sub-populations may communicate occasionally to share good solutions, through a mechanism referred to as migration. Sub-populations typically consist of some tens to hundreds of chromosomes and the approach is therefore highly suited to coarse-grained processing platforms. When the rate of migration is low, good speed-ups can be achieved. Although primarily motivated by speed-up, it has long been recognised that parallel implementations may search better than serial algorithms on complex multi-modal search spaces, see for example [4]. The reason for this is the inherent niching effect that occurs when sub-populations are isolated for a period (between migrations) during which their search trajectories can independently evolve. This can lead to a reduced probability of premature convergence.

Migration is also sometimes said to contribute to the avoidance of premature convergence by improving diversity, though this is questionable as it can also lead to increased effective selection pressure and disruption of the niching behaviour. The independence of the sub-populations may be deemed poorly controlled as it depends on chance - that is, where the initial samples are taken - and the effect of migrations. In practice, sub-populations partially overlap in their search of particular areas of the search space, either discovering fit regions independently or sharing them through migration. This lack of discrimination has previously motivated the development of an alternative approach, referred to here as the partition model [5][6][7]. The partition model not only distributes the objective function evaluations, but also partitions the search space so that each sub-population is responsible for its own unique region of the space. In that way, the independence of sub-populations is taken to an extreme, and empirical results suggest that this is useful. However, despite encouraging results on small-scale problems, the partition model has certain scalability problems, discussed in [8].

In this paper a new multi-modal numerical optimisation technique employing cluster analysis is described. The intention is to replicate the beneficial behaviours of the partition model using more scalable techniques. Essentially an extension of the island model, it tries to find clusters of samples that overlap between different sub-populations and merges these overlapped clusters in one of the sub-populations. Other sub-populations are discouraged from re-entering the region of these clusters by applying a simple derating function to their fitness evaluations. The resultant reduction in search overlap means that sub-populations are free to explore other areas of the search space. This has the beneficial effect of reducing premature convergence, widely recognised as a major difficulty in population-based search.

The technique described does not require the *a priori* knowledge of the fitness landscape often needed by other clustering approaches. The algorithm is shown to perform significantly better than the island model on the problems used whilst being only marginally worse than the partition model; a situation that is more than compensated for by the improved scalability of the approach.

The rest of the paper is organised as follows. Section 2 describes the parallel models considered in this study. Section 3 details the clustering algorithm. Section 4 provides a performance comparison, and Section 5 concludes.

2 Parallel Models

This section reviews the coarse-grained PGAs considered in this study, and establishes the basis for the clustering approach.

2.1 Island Model

The configuration of island-model PGAs can be divided into two main parts: population sizing and migration. The size of the population is typically the parameter that has the most influence on performance (although other parameters, such as mutation and migration, are inevitably linked). The spatial allocation of resources (i.e. the decision on the number and the size of sub-populations) has been solved for extreme cases [9]. Migration is a complex operator, affected by several interrelated parameters (including, migration timescale, migration rate and topology) as well as the characteristics of the fitness landscape. The impact of these parameters on the quality of search has been investigated by Cantú-Paz in [3], who provides rational design guidelines.

Migration may introduce diversity into a sub-population. The introduction of a new sample, geometrically distant from the existing samples, tends to cause the sub-population to investigate the intervening space. Thus it can possibly locate further optima in the intervening space and lead to improved performance. However, migration also has a potentially negative effect. It can increase the convergence speed by replicating fit structures and cause multiple sub-populations to converge to the same region of the search space. There is no guarantee that different sub-populations will converge on different optima even when there is no communication between sub-populations. Thus, there is an indiscriminate overlap in the work effort between processing nodes. This overlap can be beneficial, particularly on problems that are not deceptive, because the more samples taken from an optimum region the more accurate the final solution will be. However, on deceptive problems this behaviour is a disadvantage as it leads to overlooking large areas of the search space in which globally fit solutions may be found in favour of the more readily identified areas of lesser fitness.

2.2 Partition Model

To ascertain whether the removal of the indiscriminate overlap between sub-populations is beneficial, the partition model was proposed by Vincent [5]. The main concept is the division of the search space into a number of (initially equal size regions) that are separately optimised. When the fitness distribution of the search space is irregular, as is typical with optimisation problems, equal sized regions are not of equal worth and should not therefore be subject to the same degree of sampling. Depending on the nature of the fitness landscape, this computational load imbalance could severely reduce efficiency. To address this, a load-balancing algorithm was used to adjust the decomposition (i.e. position of partitions) of the search space according to the fitness distribution - algorithm details can be found in [6].

It has been shown that the partition model is feasible for problems of low-dimensionality and that is offers substantially superior performance, in terms of both rapid convergence and final solution quality, when compared with the island model on a number of randomly generated multi-modal numerical optimisation problems. However, the basic concept is not scalable [8], as the division of the search space results in a minimum of 2^n regions, where n is the number of parameters being optimised. Thus, on problems of practical interest, with some tens to hundreds of parameters, it will not be possible to partition every dimension due to the limitation of number of processing nodes available and overheads involved. One solution is simply to select a subset of parameters to partition, but this requires *a priori* knowledge of the search space in order to select significant parameters. Thus, the partition model may fail to effectively split the optimisation workload between sub-populations and will not therefore offer the superior performance found on small-scale problems.

2.3 Cluster Model

The natural behaviour of a genetic algorithm can be informally described as comprising three phases. Following initialisation, a brief period of highly explorative search occurs, driven primarily by crossover. Quite quickly, though, the selection pressure will cause clusters to begin forming around fit regions of the search space. These regions may represent individual peaks or just a generally fit part of the space. During this period, explorative search gradually diminishes and the utility of crossover is lessened, until the population is largely categorised by a number of relatively tight clusters of samples plus a scattering of a few randomised samples throughout the space. At this point, unless an operator is employed to inject diversity into the population, the course of the optimisation, excepting some fortuitous mutation, is set. The final phase consists of the further tightening of clusters and an elimination process whereby the fittest clusters receive increasing numbers of samples until low fitness clusters are removed entirely. Ultimately, the population will converge to a single cluster.

In practice, this clustering behaviour can be exploited to identify regions of the search space in which sub-populations overlap, provided that a reliable and efficient clustering algorithm is used. The cluster model proposed here operates in a similar manner to an island model, but periodically performs a cluster analysis on each sub-population. The clusters found in each sub-population are compared to identify those that overlap. Overlapping clusters are merged, and clusters redistributed amongst sub-populations so that only one is optimising in that region of the search space. Further, the objective function can be modified by composition with some derating function to discourage other sub-populations from re-entering that region of the space. If the overlap of work effort can be removed it means that in each case $(n-1)$ sub-populations can be freed to further explore the search space, except the overlapped region, where n is number of sub-populations that contain samples from the overlapped region. It is possible to locate further optima in other regions of the search space and this leads to a higher probability of locating the global optimum or other satisfactory solution.

Migration can be used in the initial stages of the cluster model when it can help with exploration, but can be switched off when overlapping clusters appear to prevent the reintroduction of overlap. Naturally, eliminating overlap means that the number of samples that can be focussed around any one peak is limited by the sub-population size. So, when the algorithm enters the final phase, however this may be identified, migration could be switched back on to then promote overlap in order to focus sampling on the best of the discovered optima and increase the final solution quality. The effect of these variations is still to be fully investigated. The overall effect of the cluster model, however, is similar to that of the partition model.

Naturally, this approach bears some similarity with niching methods, such as fitness sharing first introduced by Goldberg and Richardson [10] [11], but the motivation is different. Their experiments showed that a modified GA is able to maintain stable clusters of samples on significant peaks, with the cluster size being roughly proportional to the fitness of the peak. One of the main problems in applying fitness sharing and other related techniques is the determination of the niche radius, which is directly related to the properties of the search space and cannot be assumed to be known *a priori*. Here, such *a priori* knowledge is largely avoided (see the next section).

3 Clustering Algorithm

The basic concept is to find the overlapping regions between slaves during the run and redistribute the samples from each overlapping region to one slave in order to force other slaves to further explore the search space. Given that the pattern of samples within a sub-population at any point in time can be characterised by randomised scatter points and clustered groupings, the objective of the cluster algorithm is to reliably recognise these groupings so that comparisons between the clusters formed by different sub-populations can be made. Cluster analysis methods are normally categorised into either hierarchical or non-hierarchical approaches [12]. In this study, a non-hierarchical algorithm, which classifies the data into a single partition of k clusters, is applied to cluster all samples in a sub-population into $k \leq N$ niches, where N is the number of samples in the sub-population. Clustering requires a distance metric and the Euclidean distance is used here. One sample from each cluster is chosen to be the "representative" sample. In this algorithm, the best sample is chosen since that will be the closest to the true optimum about which the cluster has formed. Note that this might not be the centre of the cluster itself, and the determination of overlap is therefore somewhat fuzzy. Given the notation: Q is a sequence of representative samples; P is the population of samples; $D(i, j)$ is the Euclidean distance between sample i and j; C is a sequence of clusters, with each element being a sequence of one or more samples; $\#x$ is the number of elements in sequence x; $x[i]$ is the i^{th} element of x; and $^\frown$ is the concatenation operator, the algorithm can be specified as follows:

//step 1 : initialise cluster diameter threshold
1. $D_{\text{threshold}} = D^0_{\text{threshold}}$
//step 2 : determine representative samples, Q
2. sort population in descending order of fitness, $P' = sort(P)$
3. take best sample as first representative sample, $Q = Q^{\cap}P'[1]$
4. for each sample $i = 2 \ldots \#P$
5. if

$$\min_{j=1}^{\#Q} \{D\left(P'[i], Q[j]\right)\} > D_{\text{threshold}}$$

6. then treat sample i as new representative sample, $Q = Q^{\cap}P'[i]$
//step 3 : classify all samples in P' using partition clustering based on Q
7. for each sample $i = 1 \ldots \#P$
8. find $j, j = 1 \ldots \#Q$, such that $D(P'[i], Q[j])$ is a minimum
9. assign sample to relevant cluster, $C[j] = C[j]^{\cap}P'[i]$
// step 4 : reset cluster diameter threshold based on actual size of largest cluster
10.

$$D_{\text{threshold}} = \max_{c=1}^{\#C} \left\{ \max_{i=1}^{\#C[c]} \{D\left(C[c][i], Q[c]\right)\} \right\}$$

// step 5 : repeat step 2 to step 4 for next cluster analysis

The value of the initial cluster diameter threshold, $D^0_{\text{threshold}}$, is not critical as it is only used for the first clustering. The value should be related to the size of the space, and a useful starting point is to take the bounds of the space in the longest dimension and divide by the number of sub-populations.

The implementation used in this study comprises a master-slave configuration, where each slave implements a sub-population and the master performs the centralised operations associated with the processing of clusters. The clustering algorithm is executed by each slave and therefore the serial part of the implementation where the master process is executing is minimal. Further distribution may be possible, but has not yet been investigated. The clustering algorithm is executed periodically, after each epoch of E generations. Each slave classifies its samples into different clusters and sends the set of representative data (Q) to the master. The master then determines whether there are any overlapping clusters. Overlap is determined by a parameter, $\sigma_{\text{threshold}}$. If the distance between the representative data of two clusters from different slaves is less than $\sigma_{\text{threshold}}$ the two clusters are considered to overlap. This threshold can be related to the cluster diameter threshold without using *a priori* knowledge. Values in the range $D_{\text{threshold}} \times 0.8 \leq \sigma_{\text{threshold}} \leq D_{\text{threshold}} \times 1.2$ have been found appropriate. A value of $\sigma_{\text{threshold}} = D_{\text{threshold}}$ is used here.

The samples of overlapping clusters must be redistributed so that only one slave has samples in that region. This redistribution affords the opportunity to perform some form of load balancing so that the mean fitness is roughly equally across sub-populations. Thus, samples from overlapping clusters are merged and sent to the sub-population with the worst fitness. This process is repeated for all overlapped clusters.

Each slave, on receiving the results of the cluster comparison from the master, can modify the fitness function by composition with some derating function to discourage further sampling in areas of overlap. The approach adopted here makes the overlapping regions flat and of minimum fitness for every slave except that which has received the merged cluster. Naturally, samples removed from a sub-population by merging overlapped clusters must be replaced. In [7], in connection with the partition model, results were presented for two forms of regeneration - random and evolved. Evolving the replacement samples from those remaining valid samples was shown to perform better, and did not show any signs of tending to premature convergence, and is therefore also used in the cluster model. If there are insufficient samples remaining for the GAs selection and reproduction operators, replacements are generated at random until there are sufficient to evolve the remainder.

4 Empirical Investigation

To evaluate the performance of the cluster model, experiments from [5] and [7], which compared the serial, island model and partition model approaches, are repeated here. Two experiments are reported. The first examines the performance on a simple problem with a variable degree of deception, using two sub-populations, as per [5]. The second examines the performance on a larger problem using ten sub-populations, as per [7]. In general, experimental conditions follow those previously used.

As in previous work, the GA employs binary tournament selection without replacements, and new samples replace the population's worst. The crossover is a $BLX - 0.5$ blend operator [13] and mutation is an exponentially distributed creep operator with low probability of causing significant disruption. In all cases, an empirically determined optimal configuration is used for each algorithm. The optimal configurations for experiment one are $N = 40, \mu = 40, P_m = 0.1$ for the island model; $N = 35, P_m = 0.2$ for the partition model, with partitions adapted every 50 generations as per the rules specified in [6]; $E = 100$, $D^0_{\text{threshold}} = 5$ for the cluster model; where N is the size of sub-population, μ is migration rate, P_{m} is the per-parameter mutation probability, E is the number of generations between cluster analysis, $D^0_{\text{threshold}}$ is the initial cluster diameter used by the clustering method, and $\sigma_{\text{sthreshold}}$ is the threshold that determines whether two clusters overlap. The configuration for experiment two is $N = 50, P_{\text{m}} = 0.1, \mu = 40$ for island model; $N = 50, P_{\text{m}} = 0.1$ for partition model, with partitions adapted every 50 generations as per the rules specified in [6]; $E = 100, D^0_{\text{threshold}} = 100$ for the cluster model. All experiments are executed for 500 trials and average data reported.

4.1 Ability to Cope with Deception

In [5], the performance of the partition model was examined on a simple problem with a variable degree of deception. The basic concept was to take an easily

optimised problem, where the global optimum (denoted o_1) is located in an area of generally high fitness, and add a new global optimum (denoted o_2) of relatively small area of attraction in an area of generally poor fitness. Algorithms with "greedy" sampling behaviour will tend to overlook this new peak and focus their attention on the more easily obtain solution offered by o_1. By varying the area of attraction of o_2, the performance under varying degrees of deception can be investigated. The partition model was previously shown to cope significantly better with this than the island model, finding the deceptive peak with higher probability.

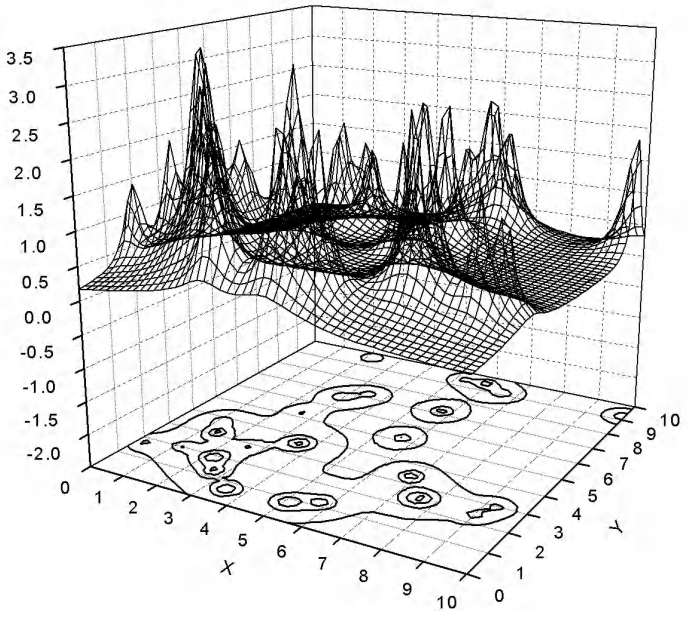


Fig. 1. Simple multi-modal test problem

The unmodified problem (i.e. without o_2), shown in Figure 1, comprises 25 peaks of random height, distribution and area of attraction. The best solution, o_1, is approximately 3.66. The new peak, o_2, was added at $(x = 8, y = 6)$, with an approximate height of 5.15. Since the algorithms used are elitist, convergence to o_2 is guaranteed if a sample of fitness greater than o_1 is reached. Therefore, the area of the peak exposed at this height is considered to be indicative of the difficulty of locating it. This exposure is denoted α, and expressed as a percentage of the total area of the search space.

Figure 2 and Figure 3 compare the average solution quality and success rate respectively for each model for a range of α. The solution quality is the fitness of the best individual at each generation. The success rate is the number of times the final solution of a trial is within some small distance of the global optimum

(0.2 is used here). With small α, none of the techniques is able to locate the global optimum because it is simply too small. With large α, all three techniques can readily locate the global optimum. At these extremes the solution quality is similar for all models. Between these extremes, however, the partition model, as previous found, offers a significant improvement in performance. The largest difference is evident for $\alpha = 0.2$, with, approximately, an 8% improvement in solution quality and a 21% improvement in success rate.

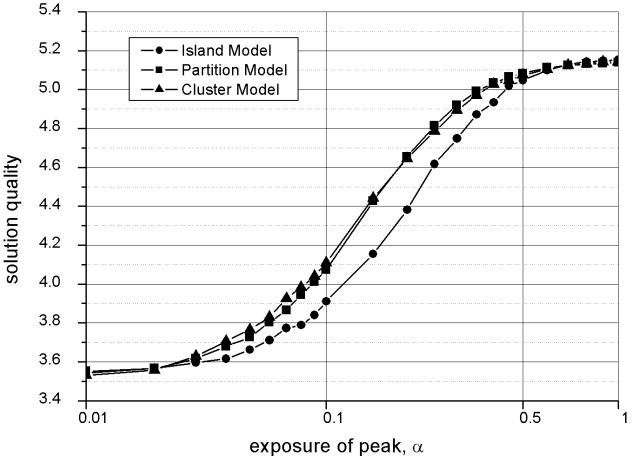


Fig. 2. Comparison of solution quality versus α

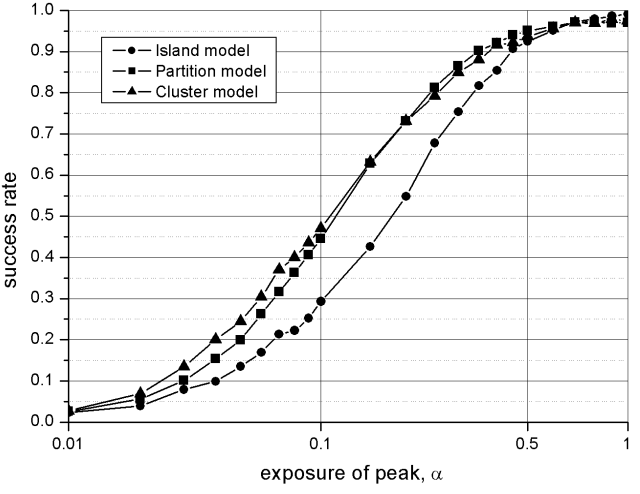


Fig. 3. Comparison of success rate versus α

This result is comparable to that previously reported. Importantly, the performance of the cluster model under these conditions is practically identical to that of the partition model.

Figure 4 and Figure 5 illustrate the clustering in action, with a plot of the sub-population state before and after clustering is applied (different symbols denote the location of samples from each sub-population). Areas of overlap are clearly removed by the clustering process.

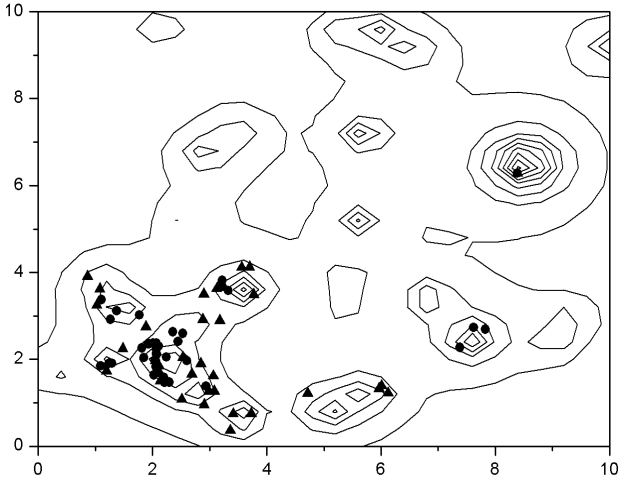


Fig. 4. State of the sub-populations prior to clustering

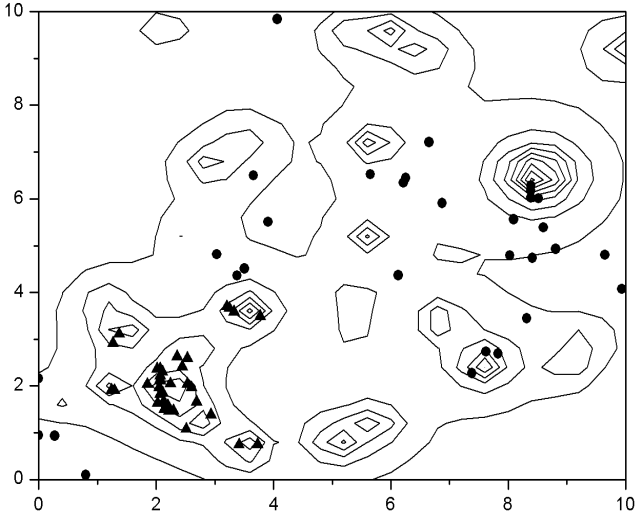


Fig. 5. State of the sub-populations after clustering

4.2 A More Complex Problem

The performance of the three models is compared on the larger, more complex problem shown in Figure 6, as previously used in [7].

This comprises 1000 randomly distributed optima having very small area of attraction.

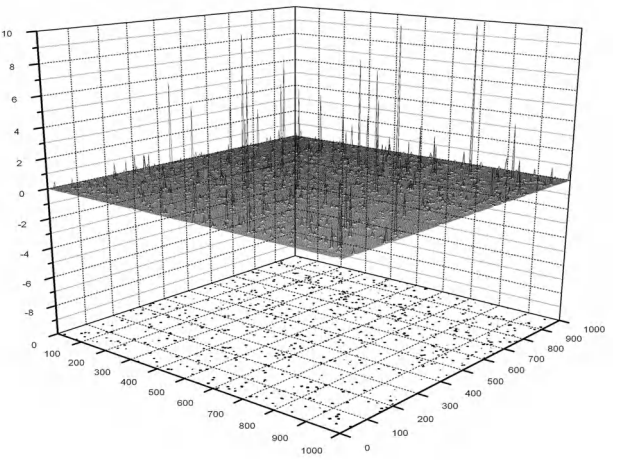


Fig. 6. A harder test problem

Ten sub-populations are used for each model. The solution quality is compared in Figure 7. From this, it is clear that the partition model offers a substantial increase in both convergence speed and final solution quality over the island model. The cluster model, whilst not quite reaching the general performance level of the partition model, is nonetheless a significant improvement over the island model, and manages to achieve the highest final solution quality. The small loss of convergence speed is considered a reasonable price for the improved scalability that the cluster model offers over the partition model.

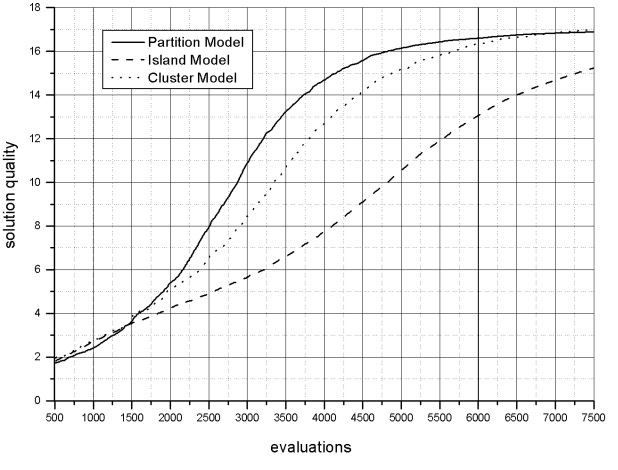


Fig. 7. Comparison of solution quality

5 Conclusions

This paper has presented a new model for a coarse-grained PGA referred to here as the cluster model. The cluster model effectively removes the overlap in sampling between sub-populations which is shown to offer superior performance in terms of rapid convergence and final solution quality compared with the island model on two relatively small-scale problems. It is shown to have generally comparable performance with the partition model from which it was developed, but removes the scalability problems that prevent the wide application of the partition model. Further work is being conducted on the clustering and cluster redistribution algorithms to maximise robustness, computational efficiency and effectiveness. Empirical study on a wider range of test problems is also required to establish the domain of applicability of the cluster model.

References

[1] D. E. Goldberg, *Genetic algorithms in search, optimisation and machine learning*, Addison Wesley, ISBN 0201157675, 1989.

[2] E. Alba, J. M. Troya, *A survey of parallel distributed genetic algorithms*, Complexity, 4(4), Wiley, 1999.

[3] E Cantú-Paz, *Efficient and accurate parallel genetic algorithms*, Kluwer, ISBN 0792372212, 2000.

[4] V. S. Gordan, D. Whitley, *Serial and parallel genetic algorithms as function optimisers*, Proc. 5th Int. Conf. on Genetic Algorithms, Morgan Kaufmann, 1993.

[5] J. Vincent, G. King, *Performance Implications of Domain Decomposition in the Parallelisation of Genetic Search*. Proc. Genetic and Evolutionary Computation Conference, Late Breaking Papers, pp443-449, July 2001.

[6] J Vincent, G King, B Dupée, *Dynamic balancing of workload in a parallel genetic algorithm based on domain decomposition*, Proc. IASTED Int. Conf. on Parallel and Distributed Computing Systems, ACTA Press, 2001.

[7] J Vincent, G King, *Improving the parallel genetic algorithm with a view to online optimisation*, Proc. IASTED Int. Conf. on Intelligent Systems and Control, ACTA Press, 2001.

[8] J. Vincent, R. Mintram, *On the Scalability of a Parallel Genetic Algorithm Based on Domain Decomposition*. Proc. IASTED Int. Conf. on Artificial Intelligence and Applications, ACTA Press, 2003

[9] D. E. Goldberg, H. Kargupta, J. Horn, E Cantú-Paz, *Critical deme size for serial and parallel genetic algorithms*. Technical Report IlliGAL 95002, University of Illinois, 1995.

[10] D. E. Goldberg, J. Richardson, *Genetic Algorithms with Sharing for Multimodal Function Optimisation*. Proc. 2nd Inter. Conf. Genetic Algorithms, pp41-49, 1987.

[11] K. Deb, D. E. Goldberg, *An Investigation of Niche and Species Formation in Genetic Function Optimisation*, Proceedings of The Fifth International Conference on Genetic Algorithms and Their Applications. Morgan Kaufmann, pp42-50, 1989.

[12] B. Everrit. *Cluster Analysis*, Edward Arnold, ISBN 0470220430, 1993.

[13] L. J. Eshelman, J. D. Schaffer, *Real-coded genetic algorithms and interval schemata*, Foundations of Genetic Algorithms 2, Morgan Kaufmann, San Mateo, CA, ISBN 1558602631, 1993.

A Study of Diversity in Multipopulation Genetic Programming

Marco Tomassini[1], Leonardo Vanneschi[1],
Francisco Fernández[2], and Germán Galeano[2]

[1] Computer Science Institute, University of Lausanne, Lausanne, Switzerland
[2] Computer Science Institute, University of Extremadura, Spain

Abstract. In this work we study how using multiple communicating populations instead of a single panmictic one may help in maintaining diversity during GP runs. After defining suitable genotypic and phenotypic diversity measures, we apply them to three standard test problems. The experimental results indicate that using multiple populations helps in maintaining phenotypic diversity. We hypothesize that this could be one of the reasons for the better performance observed for distributed GP with respect to panmictic GP. Finally, we trace a sort of history of the optimum individual for a set of distributed GP runs, trying to understand the dynamics that help in maintaining diversity in distributed GP.

1 Introduction

The diversity of genetic material plays an important role in evolutionary algorithms (EAs). In fact, it has been shown that premature loss of diversity may lead to search stagnation on restricted regions of the search space instead of convergence towards better solutions. In tree-based genetic programming (GP) the problem is complicated by the fact that genotype-to-phenotype mapping is not as straightforward as it is in other EAs. At any point in time during evolution, the population can be examined from the point of view of its genotypic pool – the structure of the trees – or from the point of view of the capability of the individuals to solve the problem at hand – their fitness –. Thus, both genotypic and phenotypic diversity play a role in GP and both are not necessarily correlated in a straightforward manner. In particular, the phenomenon of "bloat", consisting in the tendency of code to grow in size over generations, is well-known and it often gives rise to large non-functional tree portions [11]. Over the years, researchers have suggested various techniques for maintaining diversity in artificially evolving populations. Among those that have found application to GP, one may mention fitness sharing [6] which has been extended to GP in [7], and multi-objective optimization [5], where fitness, size, and diversity are the objectives to be satisfied. In the past few years, systematic experimental investigation of the behavior of semi-isolated populations in GP have been done. Among the others, we have empirically observed that distributing the individuals among several communicating islands allows not only to save computation

P. Liardet et al. (Eds.): EA 2003, LNCS 2936, pp. 243–255, 2004.

time, due to the fact that the system runs on multiple machines, but also to find better solutions' quality. This is true both on a set of common GP benchmarks and on some real-life problems [8,4] and it agrees with results obtained by other researchers (see for instance [1,13]). These results, and the analogous ones for EAs, have often been attributed to better diversity maintenance due to the periodic migration of groups of good individuals among the subpopulations. For example, McPhee and Hopper [12] suggested that demes might be a possible way for maintaining diversity in GP (but they did not explore the issue). We also believe that this might be the case and in this paper we present a study on the evolution of diversity in multi-island GP. Such an investigation has, to our knowledge, never been performed before. A preliminary, but rather sketchy analysis of the issue has been presented by us in [14]. One advantage of multiple populations as means for maintaining diversity is that, in contrast to the clever methods mentioned above, diversity is maintained "for free", so to speak, without any particular algorithmic device beyond the simple communication among islands.

The paper is organized as follows. In section 2 we present the diversity measures employed. Section 3 shortly describes the test problems used and the GP environment. Sections 4 and 5 discuss the experimental results obtained, and section 6 provides the conclusions and discusses future work.

2 Diversity Measures

A rather complete survey of diversity measures in panmictic GP have been presented in [2,3]. The diversity measures that we use in this paper are based on the concepts of *entropy* and *variance*. Both these concepts are used to measure the phenotypic (i.e. based on fitness) and genotypic (i.e. based on the syntactical structure of individuals) diversity of populations.

The entropy of a population P is defined as follows:

$$H(P) = \sum_{j=1}^{N} F_j \log(F_j)$$

If we are considering phenotypic entropy, we define F_j as the fraction n_j/N of individuals in P having a certain fitness j, where N is the total number of fitness values in P. To define genotypic entropy, we have decided to use two different techniques. The first one consists in partitioning individuals in such a way that only identical individuals belong to the same group. The algorithm used to perform this partitioning is efficient as it is based on previously stored triplets of trees attributes, composed by the total number of nodes, the number of leaves and the number of internal nodes. Only in case of equal triplets, trees are visited to establish if they are indeed identical (for the sake of brevity, this algorithm will be called triplet algorithm in the following). In this case, we have considered F_j as the fraction of trees in the population P having a certain genotype j, where N is the total number of genotypes in P. The second technique consists

in defining a distance measure, able to quantify the genotypic diversity between two trees. In this case, F_j is the fraction of individuals having a given distance j from a fixed tree (called *origin*), where N is the total number of distance values from the origin appearing in P.

The variance of a population P is defined as follows:

$$V(P) = \frac{1}{n-1}\sum_{i=1}^{n}(f_i - \bar{f})^2.$$

If we are considering phenotypic variance, $\bar{f}$ is the average fitness of the individuals in P, f_i is the fitness of the i^{th} individual in P and n is the total number of individuals in P. To define genotypic variance, we only use the notion of tree distance. In this case, $\bar{f}$ is the average of all the individual distances from the origin tree, f_i is the distance of the i^{th} individual in P from the origin tree and n is the total number of individuals in P.

A few tree distances have been proposed in literature. Here we use Ekárt's and Németh's definition [7]. According to this measure, given the sets $\mathcal{F}$ and $\mathcal{T}$ of functions and terminals, a coding function c must be defined such that $c : \{\mathcal{T} \cup \mathcal{F}\} \to \mathbb{N}$. Then, the distance of two trees T_1 and T_2 with roots R_1 and R_2 is defined as follows:

$$dist(T_1, T_2) = d(R_1, R_2) + k\sum_{i=1}^{m} dist(child_i(R_1), child_i(R_2))$$

where: $d(R_1, R_2) = (|c(R_1) - c(R_2)|)^z$, $child_i(Y)$ is the i^{th} of the m possible children of a generic node Y, if $i \leq m$, or the empty tree otherwise, and c evaluated on the root of an empty tree is 0. Constant k is used to give different weights to nodes belonging to different levels and z is a constant usually chosen in such a way that $z \in \mathbb{N}$. In the following experiments, the empty tree will be chosen as the origin.

3 Test Problems and GP Parameters

We decided to address a set of problems that have been classically used for testing GP: the even parity problem, the symbolic regression problem and the artificial ant on the Santa Fe trail problem, since there is a fair amount of accumulated knowledge on those in the GP community ([10,11]). The following is a brief description of the problems, details can be found in [10].

Even Parity 4 Problem. The boolean Even Parity k function of k boolean arguments returns *true* if an even number of its boolean arguments evaluates to true, otherwise it returns *false*. If $k = 4$, then 16 fitness cases must be checked to evaluate the fitness of an individual. The fitness is computed as 16 minus the number of hits over the 16 cases. Thus a perfect individual has fitness 0, while the worst individual has fitness 16. The set of functions is the following: $\mathcal{F} = \{NAND, NOR\}$. The terminal set in this problem is composed of 4 different boolean variables $\mathcal{T} = \{a, b, c, d\}$.

Artificial Ant Problem on the Santa Fe Trail. In this problem, an artificial ant is placed on a 32×32 toroidal grid. Some of the cells from the grid contain food pellets. The goal is to find a navigation strategy for the ant that maximizes its food intake. We use the same set of functions and terminals as in [10]. As fitness function, we use the total number of food pellets lying on the trail (89) minus the amount of food eaten by the ant during his path. This turns the problem into a minimization one, like the previous one.

Symbolic Regression Problem. The problem aims to find a program which matches a given equation. We employ the classic polynomial equation $f(x) = x^4 + x^3 + x^2 + x$, and the input set is composed of the values 0 to 999 (1000 fitness cases). For this problem, the set of functions used for GP individuals is the following: $\mathcal{F} = \{*, //, +, -\}$, where // is like / but returns 0 instead of *error* when the divisor is equal to 0. We define the fitness as the sum of the square errors at each test point. Again, lower fitness means a better solution.

GP Parameters. In all the experiments we used the following set of GP parameters: generational GP, crossover rate: 95%, mutation rate: 0.1%, tournament selection of size: 10, ramped half and half initialization, maximum depth of individuals for the creation phase: 6, maximum depth of individuals for crossover: 17, elitism (i.e. survival of the best individual into the newly generated population for panmictic populations. The same was done for each subpopulation in the distributed case). All the experiments should definitely be checked using a different selection scheme, since a higher or lower selection pressure could affect phenotypic diversity. This work will be done in the future. Furthermore, to avoid complicating the issue, we refrained from using advanced techniques such as ADFs, tailored function sets and so on. The distributed GP algorithm is explained in detail in [8]. Basically, each population evolves independently with the same parameters as panmictic GP, except for the migration of the best p individuals every k generations from a given island to a randomly chosen one different from itself, where they replace the worst p individuals. In all the experiments $p = 10\%$ of the population size, and $k = 10$ as these values have been shown adequate in [8]. Sending and receiving blocks of individuals is done synchronously.

4 Experimental Results

In this section we describe the results of our simulations. All the curves represent average values over 100 independent GP runs. Note that, for reasons of space, we report here the results of the simulations for one population size and a fixed number of islands for each problem. Although this might seem arbitrary, it is based on the results we obtained in [8], where we studied many more cases.

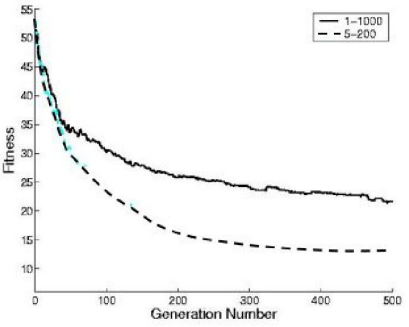


Fig. 1. Artificial Ant Problem. 1000 total individuals. Average of the best fitness vs. generations over a set of 100 independent runs.

4.1 Artificial Ant

Figure 1 shows the average of the best fitness value at each generation over 100 independent runs. Population size is 1000 and we use five subpopulations in the distributed case. This figure clearly shows that splitting the individuals on a set of communicating subpopulations leads to a gain in the best solution quality, if compared to the case of one panmictic population.

In [8], we show that this result is statistically significant and also that qualitatively analogous results can be obtained with other population sizes and other subpopulation numbers.

Figure 2(a) depicts the behavior of genotypic entropy as calculated by using structural tree distances, for the same runs as in figure 1. The gray curve represents the overall entropy of one panmictic population, while the black curve shows the aggregated entropy of all islands, i.e. the entropy of all the individuals in the islands considered as a single population. Figure 2(b) shows the evolution of genotypic entropy when using the triplet representation of unique trees. We observe that the behavior is qualitatively very similar to that of figure 2(a), thus confirming that the two entropy measures are consistent. Experiments have shown that the same thing holds for all the other cases considered in this paper. Because of that, we only show the entropy based on structural distance from now on. Figure 2(c) shows the genotypic variance at each generation. Looking at these figures, we can observe that genotypic diversity, after an initial arrangement (increase in the case of entropy, decrease in the case of variance), tends to remain constant over time. This is in agreement with the findings of Gustafson and coworkers [2,3]. The jigged behavior of the multipopulation curves when groups of individuals are sent and received is not surprising: it is due to the sudden change in diversity when new individuals enter a subpopulation. If we don't consider these oscillations, the genotypic diversity of the panmictic population and the one of the aggregated subpopulations can be considered very similar.

The behavior of genotypic diversity in individual islands can be seen in figure 3, where only two populations are reported to avoid cluttering the drawing.

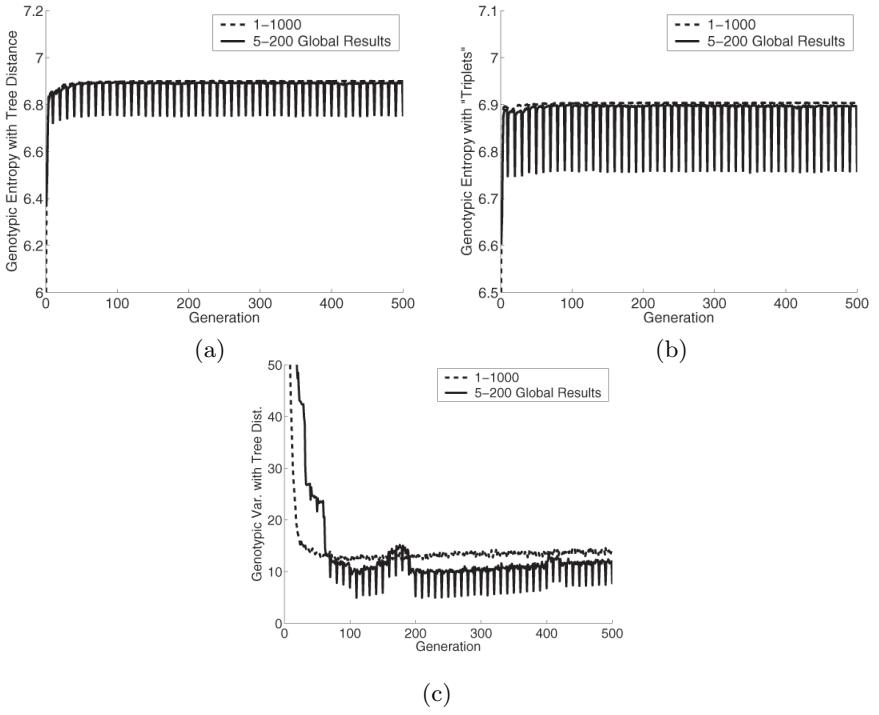


Fig. 2. Artificial Ant Problem. 1000 total individuals. (a): Genotypic entropy using structural distance. (b): Genotypic entropy using the triplet algorithm. (c): Genotypic variance using structural distance. Gray curves: panmictic population. Black curves: entropy of the aggregated subpopulations.

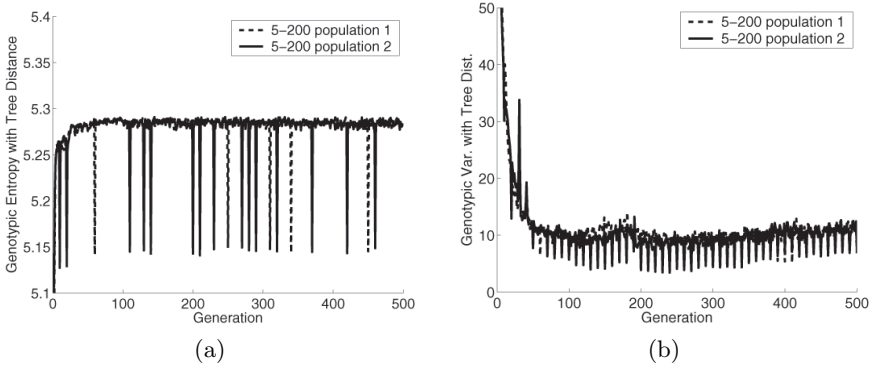


Fig. 3. Artificial Ant Problem. Genotypic entropy (a) and variance (b) using structural distances in two subpopulations of 200 individuals each. Note the change of scale of the y-axis with respect to the curves of figure 2.

Figure 4 shows graphs of the phenotypic entropy (a) and variance (b) for the panmictic population and the multipopulation case. It is apparent here that,

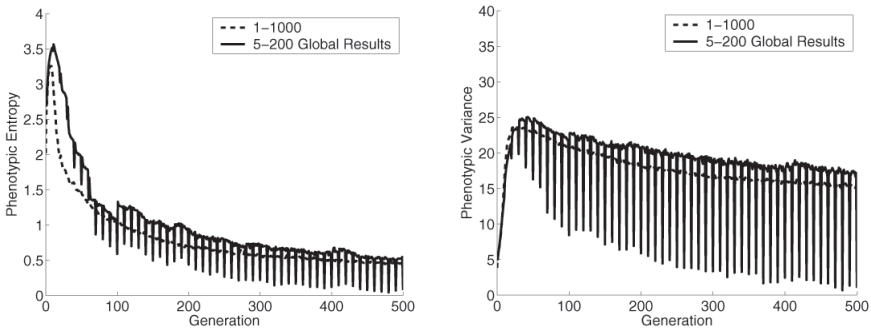


Fig. 4. Artificial Ant Problem. 1000 total individuals. Phenotypic entropy (a) and variance (b). Gray curves: panmictic population. Black curves: entropy of the aggregated subpopulations.

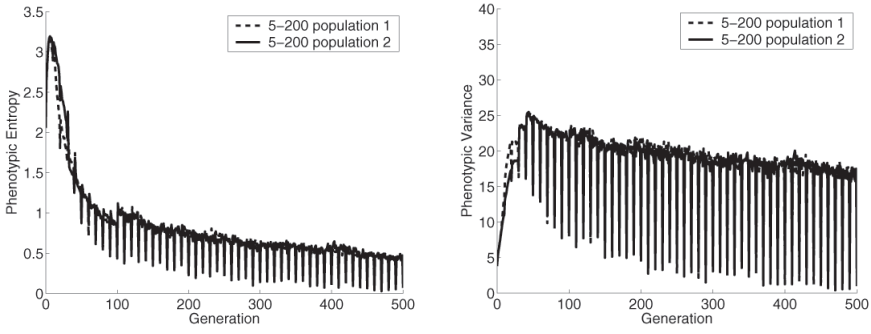


Fig. 5. Artificial Ant Problem. Phenotypic entropy (a) and variance (b) in two subpopulations of 200 individuals each.

contrary to genotypic diversity, phenotypic diversity tends to decrease steadily as time goes by, which is again in agreement with the results of [2,3]. This behavior has often been observed in GP runs, see for instance [11]. The interesting remark is that, even if in the multipopulation case the average phenotypic diversity tends to oscillate as groups of individuals are sent and received, it globally remains higher than in the panmictic case.

Figure 5 presents phenotypic entropy (a) and variance (b) for two islands. This figure and figure 3 show that the diversity behavior in the different islands is more or less the same during the evolution (for this problem).

4.2 Symbolic Regression

Figure 6 shows the average of the best fitness value at each generation over 100 independent runs. Population size is 250 and we use five subpopulations in the distributed case. This figure clearly shows that the multipopulation system allows to find solutions of better quality, if compared to the case of one panmictic

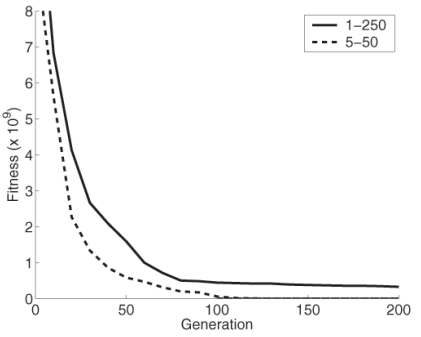


Fig. 6. Symbolic Regression Problem. 250 total individuals. Average of the best fitness vs. generations over a set of 100 independent runs. Results for the first 25 generations have been omitted, because the values of fitness are too high.

population. As it was the case for figure 1, in [8] we show that this result is statistically significant and also that qualitatively analogous results can be obtained with other population sizes and other subpopulation numbers.

Genotypic entropy and variance with structural tree distance are shown in figure 7.

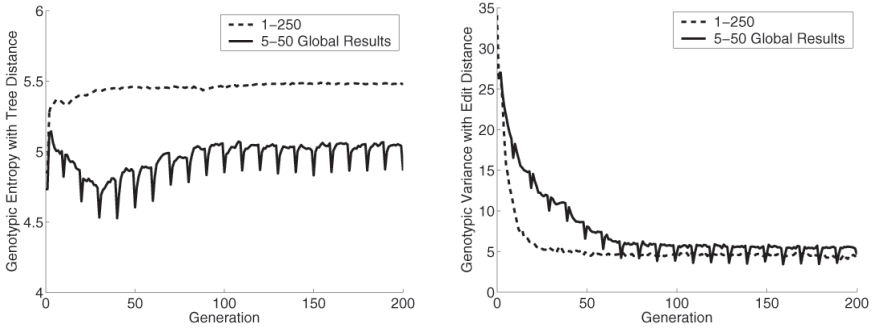


Fig. 7. Symbolic Regression problem. 250 total individuals. Genotypic entropy (a) and variance (b) calculated using structural distance. Gray curves: panmictic population. Black curves: entropy of the aggregated subpopulations.

Again, we see that, after an initial period of arrangement, genotypic diversity remains approximately constant during the evolution. Moreover, while the genotypic variance have more or less the same values for the multipopulation system and the panmictic one, the genotypic entropy is higher for the single panmictic population. A comparison between the results of figures 6 and 7(a) show that the smaller value of genotypic entropy in the multi-islands system does not affect performance.

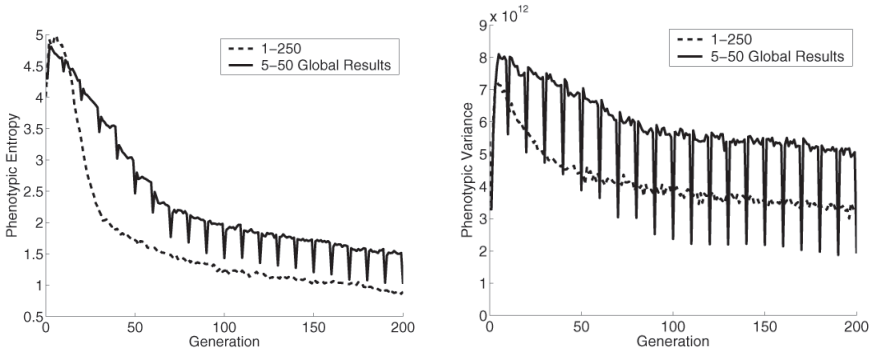


Fig. 8. Symbolic Regression problem. 250 total individuals. Phenotypic entropy (a) and variance (b). Gray curves: panmictic population. Black curves: entropy of the aggregated subpopulations.

Figure 8 shows the phenotypic diversity (as measured by entropy and variance) for the multipopulation system and the panmictic one. It appears that phenotypic diversity, in average, has higher values for the multi-island case. This is a qualitative confirmation that distribution helps in maintaining phenotypic diversity. A comparison of the results of figures 6, 7 and 8 shows that there is little correlation between genotypic and phenotypic diversity (which is in agreement with [2], and is probably due to bloat, neutral networks in genotypic space and non-functional code [11]). Moreover, no apparent correlation seems to exist between the capacity of GP to find good quality solutions and the genotypic diversity. On the other hand, the capacity of GP to find good quality solutions seems to have a correlation with phenotypic diversity, at least in these experiments.

Results of genotypic and phenotypic diversity for the single subpopulations of the multi-island system (not shown for lack of space) confirm that the amount of diversity in the different islands is, in average, more or less the same.

4.3 Even Parity 4

Figure 9 shows the average of the best fitness value at each generation over 100 independent runs. Population size is 500 and we use five subpopulations in the distributed case. Once again, the multipopulation system allows, in average, to find solutions of better quality, if compared to the case of one panmictic population. As it was the case for Figure 1 and 6, in [8] we show that this result is statistically significant and also that qualitatively analogous results can be obtained with other population sizes and other subpopulation numbers.

In Figure 10, we observe that a pattern of genotypic diversity similar to the one found for the artificial ant problem (Figure 2) emerges again: genotypic diversity has a variation in the first part of the evolution (increase for the entropy

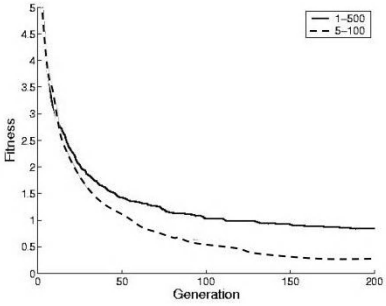


Fig. 9. Even Parity 4 Problem. 500 total individuals. Average of the best fitness vs. generations over a set of 100 independent runs.

and decrease for the variance) and then levels-off and stays in average practically constant. For the islands, there are oscillations at migration times but otherwise the behavior is quite similar to the panmictic system.

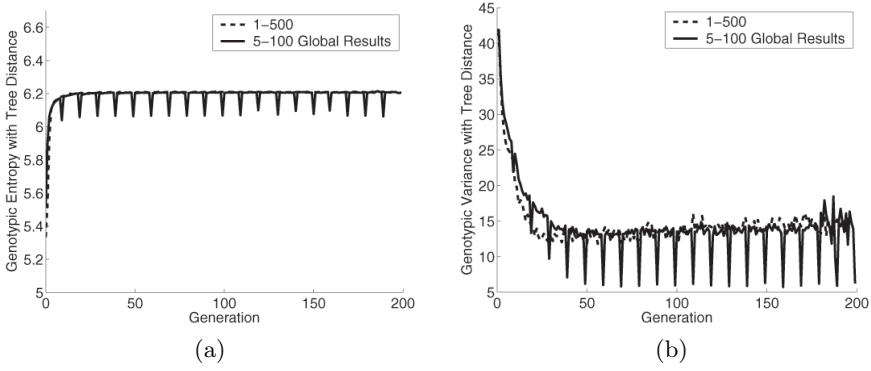


Fig. 10. Even Parity 4 problem. 500 total individuals. Genotypic entropy (a) and variance (b) calculated using the structural distance. Gray curves: panmictic population. Black curves: entropy of the aggregated subpopulations.

Also for this problem, phenotypic diversity (entropy and variance are shown in figure 11) always decreases on average during evolution, but it remains higher for the multipopulation system. The behavior for single islands is similar but is not shown here for lack of space.

5 Solutions History

Although the previous sections have shown convincingly that phenotypic diversity is better preserved in the multipopulation case, it would be interesting to study how solutions originate and propagate in the distributed system. To do so,

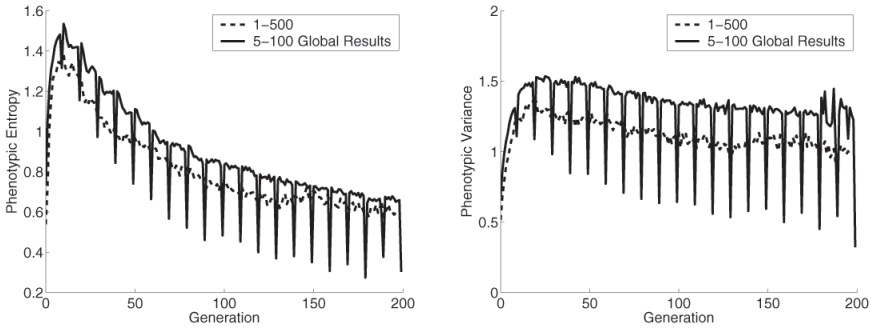


Fig. 11. Even Parity 4 problem. 500 total individuals. Phenotypic entropy (a) and variance (b). Gray curves: panmictic population. Black curves: entropy of the aggregated subpopulations.

	Pop 1	Pop 2	Pop 3	Pop 4	Pop 5
Test 1	13	12	15	14	46
Test 2	18	10	45	13	14
Test 3	46	12	16	12	14
Test 4	16	41	17	13	13
Test 5	19	21	19	20	21
Test 6	16	17	30	19	18
Test 7	16	17	30	19	18
Test 8	12	11	11	16	52
Average	20	18.62	21.5	15.62	24.5

Fig. 12. The numbers on the row corresponding to a given test represent the average amount of time spent by a solution or a part thereof in the corresponding population. The last line is the average of the eight previous tests. Results refer to the Ant problem with a population size of 1000.

we have performed several runs in which a label have been associated to individuals, containing a tag for each subpopulation. At generation zero, all these tags are set to zero, except the tag of the current subpopulation, that is set to one. Each time that an individual is formed by crossover, its label is built by taking the maximum of its parents' tags for each subpopulation. Then the current population's tag is incremented. This process allows to trace a coarse-grained history of the movement of genetic material. Subpopulations are useful if the final solutions show a good amount of mixing in their history; in other words, if

they have been formed by repeated migration and hybridation with other blocks coming from separate populations. Figure 12 is a synthesis of a few runs of the Artificial Ant problem. The row numbers for a given run represent the population's tag rates in the final solution's label, and thus represent the amount of time the final solution, or parts thereof, have spent in a given population (each column representing a population). Although the averages are not statistically significant (too few runs), they do indicate that all the islands participate in the formation of a solution.

6 Conclusions and Future Work

In this work we have studied how using loosely coupled populations instead of a single panmictic one may help in maintaining diversity during GP runs. By defining genotypic and phenotypic diversity indices and by monitoring their variation over a large number of runs on three standard test problems, we have empirically shown that diversity evolves differently in the distributed case. In fact, while genotypic diversity is not much affected by splitting a single population into multiple ones, phenotypic diversity, which is linked to fitness, remains higher in the multipopulation case for all problems studied here. Thus, the low correlation between evolution of genotypes and problem-solving behavior is confirmed in the distributed setting [2,3]. Given that better convergence properties have been empirically established for distributed GP (see, among others, [1], [13] and [8]), it is tempting to attribute this observation to the diversity behavior. Although our measures suggest that this could indeed be the case, maybe through an implicit control of the bloat phenomenon (as suggested in [9]), a direct effect cannot be established, only a plausible indication. We have also studied how solutions arise in the distributed case, and we have empirically shown that all the subpopulations contribute in the building of the right genetic material, which again tends to indicate the usefulness of having smaller communicating populations rather than a big panmictic one. In conclusion, using multiple loosely coupled populations is a natural and easy way for maintaining diversity and, to some extent, avoiding premature convergence in GP. Of course, more complicated methods for promoting diversity (e.g. [5,7]) may be used in the subpopulations in conjunction with the natural distribution of the genetic material offered by the latter. Future work includes the study of the relationships between different diversity measures, the influence of asynchronous communication policies, and of the communication topology on diversity behavior.

References

1. D. Andre and J. R. Koza. Parallel genetic programming: A scalable implementation using the transputer network architecture. In P. Angeline and K. Kinnear, editors, *Advances in Genetic Programming 2*, pages 317–337, Cambridge, MA, 1996. The MIT Press.

2. E. Burke, S. Gustafson, and G. Kendall. A survey and analysis of diversity measures in genetic programming. In W. B. Langdon et al., editor, *GECCO 2002: Proceedings of the Genetic and Evolutionary Computation Conference*, pages 716–723. Morgan Kaufmann Publishers, San Francisco, CA, 2002.
3. E. Burke, S. Gustafson, G. Kendall, and N. Krasnogor. Advanced population diversity measures in genetic programming. In J. J. Merelo et al., editor, *Parallel Problem Solving from Nature - PPSN VII*, volume 2439 of *Lecture Notes in Computer Science*, pages 341–350. Springer-Verlag, Heidelberg, 2002.
4. B. Chopard, O. Pictet, and M. Tomassini. Parallel and distributed evolutionary computation for financial applications. *Parallel Algorithms and Applications*, 15:15–36, 2000.
5. E. D. de Jong, R. A. Watson, and J. B. Pollack. Reducing bloat and promoting diversity using multi-objective methods. In L. Spector et al., editor, *Proceedings of the genetic and evolutionary computation conference GECCO'01*, pages 11–18. Morgan Kaufmann, San Francisco, CA, 2001.
6. K. Deb and D.E. Goldberg. An investigation of niche and species formation in genetic function optimization. In J. D. Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algorithms*, pages 42–50. Morgan Kaufmann, 1989.
7. A. Ekárt and S. Z. Németh. Maintaining the diversity of genetic programs. In J. A. Foster et al., editor, *Genetic Programming, Proceedings of the 5th European Conference, EuroGP 2002*, volume 2278 of *LNCS*, pages 162–171. Springer-Verlag, 2002.
8. F. Fernández, M. Tomassini, and L. Vanneschi. An empirical study of multipopulation genetic programming. *Genetic Programming and Evolvable Machines*, 4(1):21–52, 2003.
9. G. Galeano, F. Fernández, M. Tomassini, and L. Vanneschi. Studying the influence of synchronous and asynchronous parallel GP on programs length evolution. In *Congress on Evolutionary Computation (CEC'02)*, pages 1727–1732. IEEE Press, Piscataway, NJ, 2002.
10. J. R. Koza. *Genetic Programming*. The MIT Press, Cambridge, Massachusetts, 1992.
11. W. B. Langdon and R. Poli. *Foundations of Genetic Programming*. Springer, Berlin, 2002.
12. N.F. McPhee and N.J. Hopper. Analysis of genetic diversity through population history. In W. Banzhaf et al., editor, *Proceedings of the Genetic and Evolutionary Computation Conference*, volume 2, pages 1112–1120. Morgan Kaufmann, Orlando, Florida, USA, 1999.
13. K. Stoffel and L. Spector. High-performance, parallel, stack-based genetic programming. In J. R. Koza et al., editor, *Genetic Programming 1996: Proceedings of the First Annual Conference*, pages 224–229, Stanford University, MIT Press, CA, USA, 1996.
14. M. Tomassini, L. Vanneschi, F. Fernández, and G. Galeano. Experimental investigation of three distributed genetic programming models. In J. J. Merelo et al., editor, *Parallel Problem Solving from Nature - PPSN VII*, volume 2439 of *Lecture Notes in Computer Science*, pages 641–650. Springer-Verlag, Heidelberg, 2002.

Self-Improvement to Control Code Growth in Genetic Programming

Bart Wyns[1], Stefan Sette[2], and Luc Boullart[1]

[1] Ghent University, Department of Electrical Energy, Systems and Automation, Technologiepark 913, B-9052 Zwijnaarde, Belgium,
Bart.Wyns@UGent.be
[2] Ghent University, Department of Textiles, Technologiepark 914, B-9052 Zwijnaarde, Belgium

Abstract. An important problem with genetic programming systems is that in the course of evolution the size of individuals is continuously growing without a corresponding increase in fitness. This paper reports the application of a self-improvement operator in combination with a characteristic based selection strategy to a classical genetic programming system in order to reduce the effects of code growth. Two examples, a symbolic regression problem and an 11-bit multiplexer problem are used to test and validate the performance of this newly designed operator. Instead of simply editing out non-functional code this method tries to select subtrees with better fitness. Results show that for both test cases code growth is substantially reduced obtaining a reduction factor of 3–10 (depending on the problem) while the same level of fitness is attained.

1 Introduction

According to Koza [7] one of the central challenges of computer science is "... to get a computer to do what needs to be done, without telling it how to do it...". Genetic programming (GP) addresses this challenge by providing a method for automatically creating a working computer program from a high-level statement of the problem. Genetic programming achieves this goal of automatic programming (also called program induction) by genetically breeding a population of computer programs using the principles of Darwinian natural selection and biologically inspired operations [10]. The operations include reproduction, crossover (sexual recombination), mutation, and architecture-altering operations mimicking gene duplication and gene deletion in nature.

1.1 The Origin of Code Growth

An important problem with genetic programming systems is that in the course of evolution the size of individuals (and inherently also the depth) is continuously growing. Code growth can be defined as the tendency of programs generated by GP to grow much larger than is functionally necessary. The growth is out of proportion to improvements in performance and consists almost exclusively

P. Liardet et al. (Eds.): EA 2003, LNCS 2936, pp. 256–266, 2004.

of code, which does not contribute to performance. When evaluating the genetic programs over the fitness cases, much of the time is spent on irrelevant non-functional code fragments. In that way they reduce speed, consume more memory than necessary and make the programs harder to read by humans. Code growth has been a thoroughly analyzed topic during the past few years. Equally important as the analysis of the causes and sources of such a phenomenon is the search for strategies to control or moderate the effect of code growth.

Koza [7] proposed the use of a maximum permitted size (both in depth and in node count) for the evolved individuals as a parameter of GP systems. Hereby, genetic programs are allowed to grow until a predefined size is reached. More complex operators have also been introduced such as the editing operation for making the output more readable and producing simplified output, both improving the overall performance of GP. Langdon and Crawford [8] introduce two special crossover operators, so-called size fair crossovers. These operators create an offspring by replacing a randomly selected subtree of one parent with a carefully selected similar-size subtree of the other parent. Ekart [3] used a special mutation operator inspired by two forms of mutation in biology (frameshift mutation - addition or deletion of one or more base pairs and large DNA sequence rearrangements), for the simplification of the genetic programs. Code simplification has also been used by Hooper and Flann [5]. Iba [6] defines a fitness function based on a Minimum Description Length principle. The structure of the tree representing the genetic program is reflected by its fitness value. Soule [11] suggested two methods for reducing code growth: (1) the straightforward editing out of irrelevant and redundant code and (2) the use of a fitness function that penalizes longer programs.

Regardless many attempts to tackle the problem of code growth other studies have shown that seemingly non functional code (introns) shield highly fit building blocks of programs from the destructive effects of crossover [1]. Nordin [9] demonstrate through experiments that introns allow a population to keep the highly-fit building blocks and in the meantime make possible the protection of individuals against destructive crossover. Some results report premature convergence when editing out large portions of non-functional code [4].

Summarizing, the underlying cause of bloat and how to react on it is still open to debate. What is universally agreed upon, however, is that bloat occurs and often has detrimental effects on the improvement in fitness in genetic programming runs. In addition, it clearly slows down genetic programming runs by consuming CPU cycles and large amounts of memory and can hamper effective breeding. But then again solving the problem of code growth entirely doesn't seem to be the best solution since highly fit building blocks are left unprotected.

This contribution will try to control code bloat using a newly developed self-improvement (SI) operator, which tries to satisfy all of the aforementioned demands. The remaining of this paper is organized as follows. The next section will explain the basic idea and principle of the self-improvement operator. Section 3 will briefly discuss the experimental setup used for testing. Section 4 and 5 will give an overview of the results. Finally section 6 will summarize the conclusions.

2 A Self-Improvement Operator

2.1 Definition of Characteristics

In many biological systems when mating season is yet to come, individual organisms want to attract their partners of the opposite sex by using some (sometimes bizarre) rituals. For example: male birds try to chant a beautiful song in order to get as much attention as possible from a group of females. Butterflies show off their beautiful colored wings to impress the females. Even humans use complex techniques when choosing a mate. They are guided by different aspects (both inner and outer characteristics) in their quest for the "best fitting" partner.

It seems clear that animals don't use strictly one fitness value to measure the other sexes strength but rather a complete set of criteria, a set of different characteristics or objectives. In this way many biological systems use multi-objective methods to calculate some kind of weighted average of the other sexes negative and positive characteristics. Ballard and Rosca [2] report results on experiments with more adaptive representations (building block generation) that decreases the time needed for complex systems to evolve. Guided by this idea and findings an individuals representation is extended with such a set of characteristics representing its qualities. The goal is to convey more fitness information about an individual and about its capacities without explicitly incorporating a second penalizing objective like node size or tree depth. A characteristic can be defined as follows:

Definition 1. *A characteristic of an individual Y is a subtree of Y with an internal (non-terminal) node as its root.*

This definition also states that the tree of the individual Y can also be a characteristic on its own. Fitness of such a characteristic is calculated in the same way as it is obtained for the parent. To avoid overhead in memory storage only the best characteristic is kept. The rest is discarded.

2.2 Specific Selection Method

Like any other operator self-improvement selects its candidate individuals based on a certain selection strategy. Since for each individual both a global fitness measure as well as a local set of fitness characteristics is kept, the proposed selection method will use these properties to determine whether or not an individual will be selected for further processing. This method mainly consists of two stages, which operate as follows:

In the first stage individuals will be added to a global selection list S (there is only one such list for the entire population). In order to be on such a list an individual must comply with some demands:

- First of all, an individual must have at least one characteristic. This demand removes trees solely consisting of a terminal node.

– Secondly, for each individual on the list S the best characteristic must have a fitness value that is strictly higher than the fitness of the entire (mother) individual. In this way a fitter individual will replace a less performing individual. Even more, the depth and inherently also node count of the characteristic is less than the original individual. If an individual doesn't fulfill this second demand it will be thrown out of the global selection list S. Note that the root node of the parent individual is always excluded from the selection list if this demand is fulfilled. In practice this means that only individuals having internal nodes on depths larger than zero, will be added to S while others will be rejected.

In the second stage the list S is sorted according to the fitness characteristics. So the individual who has the best characteristic comes first in the list and will be, when the operator is called, the first individual to be selected.

2.3 Implementation of the Operator

When the self-improvement operator is chosen with some predefined probability, it will take the top individual from the global selection list S. By looking at the position of the fittest characteristic (i.e. the ID of the internal node) it will select the corresponding subtree. This subtree will be upgraded to a new individual by removing the other branches of the original (mother) tree. This results in a new individual with an increased fitness but a lower node count and / or depth. A graphical example is given in Fig. 1.

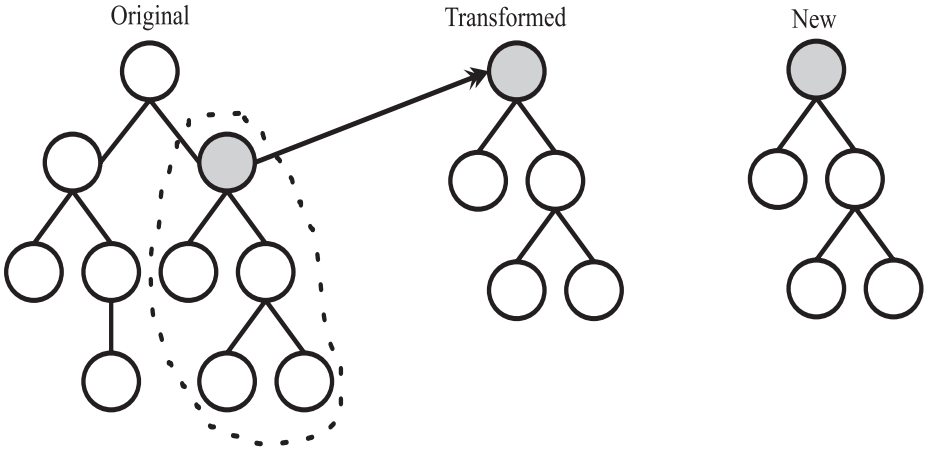


Fig. 1. Transformation of a characteristic into a complete individual.

3 Experimental Setup

This section will describe two experiments that were used to test and validate the effect of this new operator. First traditional GP will be used to show that code growth is occurring in these two examples. Node count will be carefully monitored regarding to changes in fitness. In the next section GP enhanced with the self-improvement operator will be used to tackle the code growth problem.

3.1 GP Engine

The same basic GP was used for all of the test problems. The GP is generational and is run for 100 generations. The population size is 500. The ramped half-and-half technique is used to generate the initial populations. This means that the initially generated programs are evenly distributed (ramped) between depths of 3, 4, 5, 6 and 7. For each of these maximum depths half of the programs are generated as full trees and the other half are grow trees. Other parameters for the GP are listed in Table 1.

No additional depth limit is placed on the programs (maximum depth = infinity). The GP system continues running even when an optimal solution (100% correct) has been discovered. All results are averaged over thirty independent runs.

Table 1. Breeding rates for each operator

	Rate(%)	
Operator	no SI	SI
Crossover	96%	51%
Reproduction	3%	1%
Mutation	1%	0.5%
Self-improvement	0%	47%

3.2 Symbolic Regression

The symbolic regression problem is to evolve a function $g(x)$ that matches sample point taken from a function $f(x)$. The target function chosen is e^{x^2} with x ranging from -1 to +1. Candidate solutions are tested against 40 uniformly distributed points ranging over the same interval. Fitness is defined as follows:

$$\frac{1}{1 - \sum_{i=1}^{n} |P_i - A_i|}$$

with P_i the predicted value, A_i the actual (real) output and n the number of fitness cases. More details are given in Table 2.

3.3 11-Bit Multiplexer Problem

As a second example a Boolean 11-bit multiplexer function (Fig. 2) is chosen. Koza previously stated that Boolean functions provide a useful test bed for machine learning methods for several reasons. The most important ones with respect to the proposed SI operator are:

- Boolean problems have an easily quantifiable search space.
- The number of fitness cases is finite; so it is possible to test 100% of the possible fitness cases for a given problem.

Fitness is defined as $\frac{Number\ of\ hits}{Number\ of\ fitnesscases}$. More details are given in Table 2.

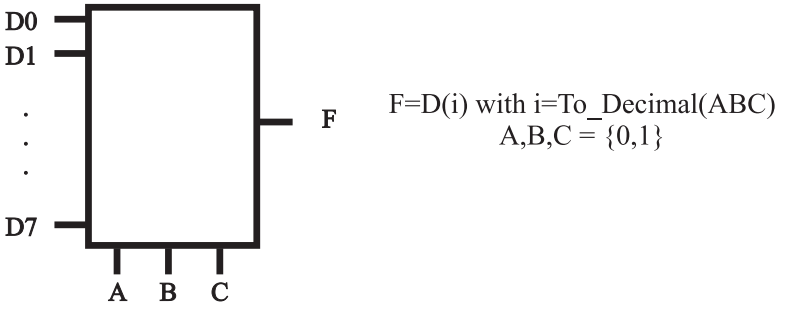


Fig. 2. Example of a 11-bit boolean multiplexer. D_i denotes input data line i. A,B and C are address inputs. F equals D_i with i the number formed by the binary digits ABC.

Table 2. Different parameter settings for the two test cases in the GP system

	Symbolic regression	11-bit multiplexer
Functions	$+, -, *, /$	and, or, not, IFTE
Terminals	constants, x	$A0..A2$: address lines, $D0..D7$: data lines
Fitness cases	40	2048

4 Results

In what follows the GP system with SI was tested and compared with the classical GP system on two aspects. First of all (and the most important one) the effect has been studied on code growth. Secondly the reflection on fitness was analyzed.

4.1 Self-Improvement versus Code Growth

As previously stated in the definition of a characteristic and based on the working principle of the self-improvement operator (in particular the specific selection method), it is expected that large individuals (i.e. individuals with node size larger than the average node count) will be chosen more often than smaller individuals. Thereby reducing population members in size. A more detailed analysis will be given later on. Figure 3 shows a crisp and clear difference considering node count when applying the aforementioned self-improvement operator to both test cases (notice the difference in scale). For both examples a respective reduction (reduction factor is depending on the problem) can be obtained.

To analyze this reduction more thoroughly it is useful to take a look at the average node size of actually selected (and transformed) individuals and the average node count of the total population (Fig. 4). For both examples this number is higher than the average node size for the entire population. This means that the self-improvement operator successfully retracts the larger individuals from the population. This can be explained by the fact that smaller individuals contain less internal nodes and thereby less candidate characteristics (with better fitness), which are required by the selection method. On the contrary it is obvious that larger individuals contain more internal nodes and consequently more candidate characteristics (with better fitness).

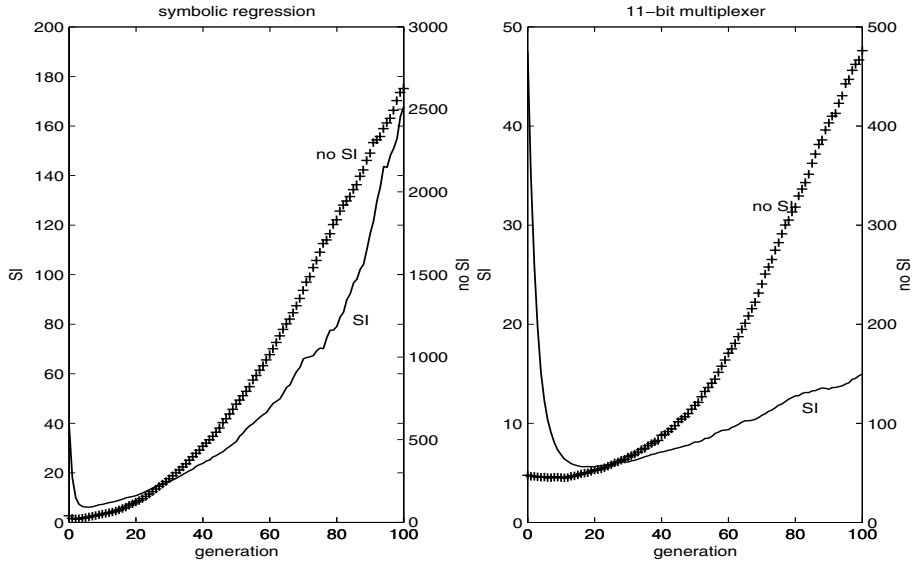


Fig. 3. Average node size with and without the use of the SI operator for both test problems. Left axis denotes mean node size when using the SI operator (black curve). Right axis denotes mean node size without the use of the SI operator ('+' signs). Please note the difference in scale.

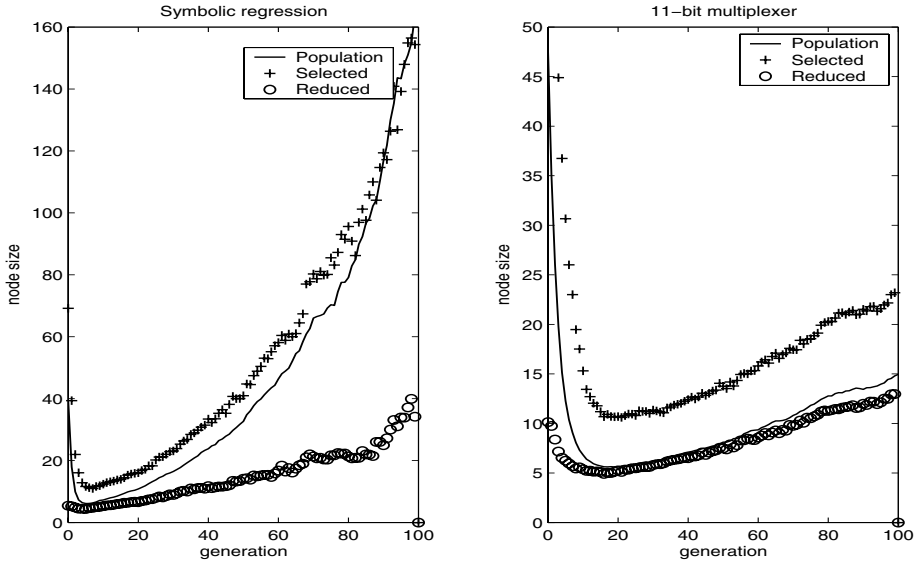


Fig. 4. Average node size of population (black curve), average node size of individuals that were actually selected by the SI operator ('+' signs) and mean node size of selected individuals after reduction by the SI operator (circles) for both test problems.

4.2 Self-Improvement versus Fitness

Remember that the specific selection method of the SI operator selects individuals based upon their best fitness characteristic. By definition, the fitness of this characteristic is larger than the fitness of the parent that will be replaced. So in GP systems that use the SI operator, selected individuals will have larger fitness compared to their parents. Comparing the results for both test problems and for both versions of GP (with and without SI) shows that there isn't a loss of performance (Fig. 5).

Summarizing, results show that a self-improvement operator in combination with a specific selection strategy based upon both fitness and the definition of a characteristic, successfully downsizes the mean node size of a population of individuals while attaining higher fitness when compared to GP systems that don't use this kind of operator. Instead of getting rid of non-functional code by using some special clean-up routines, which try to reduce the tree of an individual by eliminating portions of this tree this operator certifies that the resulting population member will have:

- higher fitness than its parent by using a specific "best-characteristic" selection strategy,
- a reduced, pruned tree with smaller node size than its parent. As has been explained in the introduction, simplification algorithms slice all excessive code

from the tree and thereby include the possibility of building block destruction by genotype altering operations like crossover and subtree mutation. Self-improvement allows individuals to prune non-functional branches but does not force it. So it is still possible that a new individual has a few non-functional portions of code hidden in its genotype. In this way highly fit code fragments can still be protected against crossover.

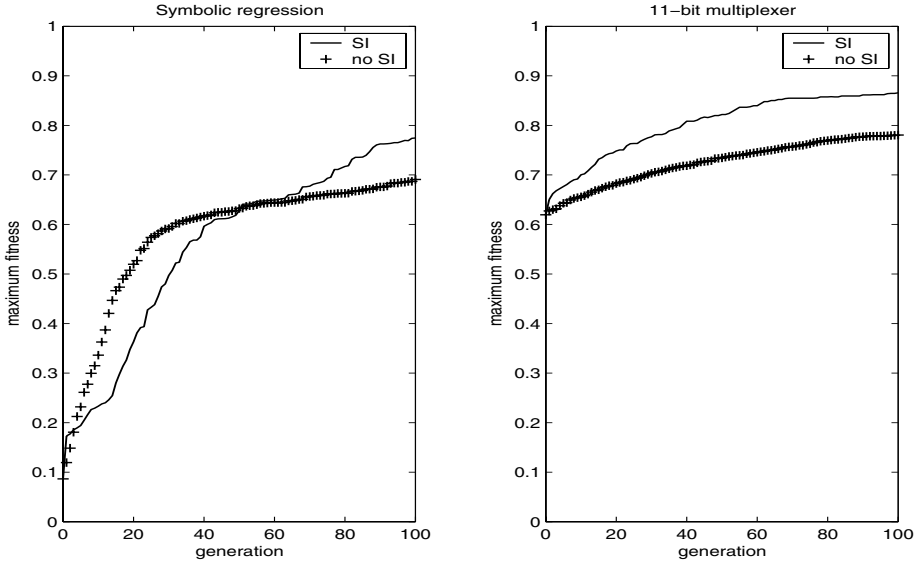


Fig. 5. Maximum fitness of the population (black curve) when using the SI operator versus maximum fitness of individuals in a GP system without the SI operator ('+' signs) for both test problems.

5 Discussion

When comparing the performance of the GP systems with and without the self-improvement operator according to old fashioned CPU time analysis there are some positive and negative aspects:

– A lot of overhead is introduced by computing all characteristics of each individual (Fig. 6). When given a full grown individual having for each function two arguments and a depth of k, $2^k - 1$ possible characteristics need to be evaluated starting from depth 0 = root). Problems which are hard to solve, such as the symbolic regression problem (e^{x^2}) produce quite large trees in order to approximate the function at its best. Figure 6 (symbolic regression problem) shows that more (evaluation) time is spent on these larger trees.

However, the SI operator reduces the node size and implicitly also the depth of an individual. This means that k in the long run stays relatively small thereby reducing the overhead generated by the calculation of the characteristics (Fig. 6, 11-bit multiplexer problem.).

– Especially memory requirements are much lower in the GP system using the SI operator. This is easy to understand since larger individuals require more storage space. Keeping a limited set of characteristics only consumes a limited amount of memory cells.

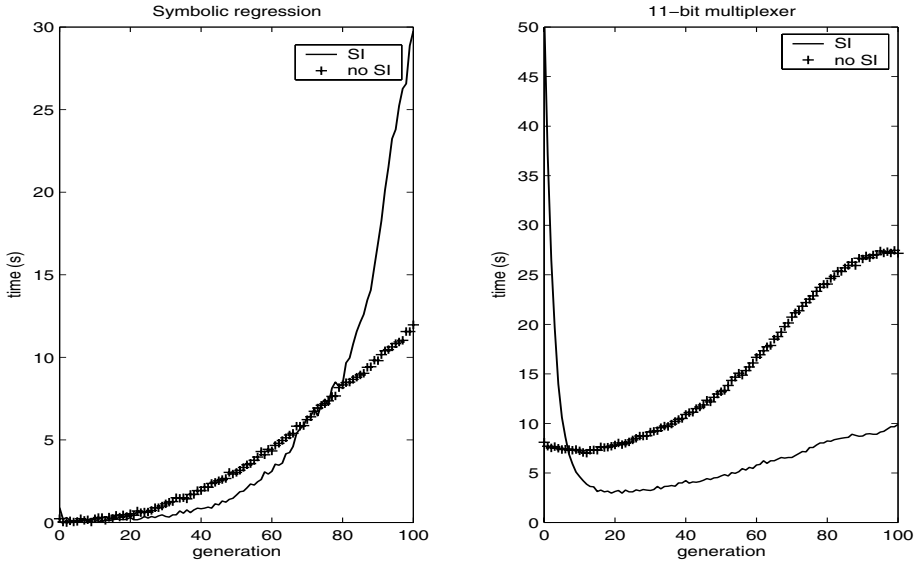


Fig. 6. Execution time in seconds for both test problems.

6 Conclusions and Future Work

By using the concept of characteristics and based on a specific selection method which sorts individuals according to best characteristics the self-improvement operator manages to control code growth effectively. By the way the algorithm is constructed no loss of fitness occurs and individuals are transformed in smaller more competent ones. In this paper two test problems (11-bit multiplexer and symbolic regression) show good results when using the SI operator to reduce tree size. Preliminary results on other problems (i.e. the artificial ant and a lawnmower control problem) show that the SI operator can be applied with good results. In that kind of problems subtrees walk the ant trail just like the complete individual, taking into account time constraints.

Instead of using a fixed probability rate for the self-improvement operator that will affect the population even when there is no code growth occurring, future work will be oriented towards the development of an adaptive control strategy that reduces the malicious effects of code bloat depending of the average size/fitness ratio of the population. In this way the population is allowed to grow and individuals are allowed to get larger as long as it corresponds with a proportional increase in fitness. Hereby allowing introns to protect individuals against destructive crossovers and keep fit building blocks. Once this growth isn't proportional with fitness improvement the SI operator can immediately respond and reduce code bloat.

References

1. Angeline, P.J.: Genetic programming and emergent intelligence. In: Kinnear, K.E. (ed.): Advances in genetic programming. MIT Press, (1994) 75–97
2. Ballard, D., Rosca, J.: Learning by adapting representations in genetic programming. In The IEEE Conference on Evolutionary Computation (1994) 407–412.
3. Ekart, A.: Controlling code growth in genetic programming by mutation. In: Langdon, W.B., Poli, R., Nording, P., Fogarty, T. (eds.): Late breaking papers of EuroGP (1999) 3–12.
4. Haynes, T.: Collective adaptation: The exchange of coding segments. Evolutionary Computation, **6(4)** (1998) 311–338.
5. Hooper, D.C., Flann, N.S.: Improving the accuracy and robustness of genetic programming through expression simplification. In: Koza, J.R., Goldberg, D.E., Fogel, D.B., Riolo, R.L., (eds.): Genetic Programming 1996: Proceedings of the first annual conference (1996) 428.
6. Iba, H., De Garis, H., Sato, T.:Genetic programming using a minimum description length principle. In: Kinnear, K.E. (ed.): Advances in Genetic Programming. MIT Press (1994) 265–284.
7. Koza, J. R.: Genetic Programming: On the programming of computers by Means of natural selection. MIT Press (1992).
8. Langdon, W. B.: Size fair and homologous tree genetic programming crossovers. In: Banzhaf, W., Daida, J., Eiben, A.E., Garzon, M.H., Honavar, V., Jakiela, M., Smith, R.E. (eds.): GECCO-99: Proceedings of the genetic and evolutionary computation conference (1999).
9. Nordin, P., Francone, F.D. and Banzhaf, W.: Explicitly defined introns and destructive crossover in genetic programming. In: Rosca, J.P. (ed.): Proceedings of the Workshop on genetic programming: from theory to real-world applications (1995) 6–22.
10. Sette, S., Boullart, L.: Genetic Programming: principles and applications. Engineering Applications of Artificial Intelligence, **14** (2001) 727–736.
11. Soule, T., Foster, J.A. and Dickinson, J.: Code growth in genetic programming. In: Koza, J.R., Goldberg, D.E., Fogel, D.B., Riolo, R.L. (eds.): Genetic Programming 1996: Proceedings of the first annual conference (1996) 215–223.

Exploring Overfitting in Genetic Programming

Grégory Paris, Denis Robilliard, and Cyril Fonlupt

Université du Littoral-Côte d'Opale
LIL
BP 719
62228 Calais Cedex, France
paris@lil.univ-littoral.fr

Abstract. The problem of overfitting (focusing closely on examples at the loss of generalization power) is encountered in all supervised machine learning schemes. This study is dedicated to explore some aspects of overfitting in the particular case of genetic programming. After recalling the causes usually invoked to explain overfitting such as hypothesis complexity or noisy learning examples, we test and compare the resistance to overfitting on three variants of genetic programming algorithms (basic GP, sizefair crossover GP and GP with boosting) on two benchmarks, a symbolic regression and a classification problem. We propose guidelines based on these results to help reduce overfitting with genetic programming.

1 Introduction

This study is about exploring overfitting in the case of supervised learning done by a genetic programming (GP) algorithm. It is well-known that overfitting learning samples is a common problem to every supervised learning methods. Nonetheless the importance of overfitting and the possible solutions to reduce it may be specific to the algorithm used, advocating for such a study. If overfitting has been well explored for some machine learning schemes, this is no yet the case for genetic programming, and this is the goal of this work to somewhat reduce this gap. The expected results are the ability to direct attention towards implementations that have good chances to reduce the effect of overfitting, such as specific choice of parameters range or GP algorithms improvements. The study have been conducted along the following guidelines: we have chosen two standard benchmarks, a symbolic regression and a classification problem; we have measured the learning error for several values of classic parameters such as program trees depth or size of population; three different GP algorithms have been tested, basic GP as it was defined by Koza in his seminal work [1], GP with sizefair crossover as proposed by B. Langdon [2], and basic GP run as the core learning algorithm in a boosting framework; noise has been added to the learning samples because it is reputed to trigger overfitting. An analysis of the learning performance under these settings is done, which allows us to extract and propose some useful guidelines in order to reduce overfitting effects with genetic programming algorithms.

P. Liardet et al. (Eds.): EA 2003, LNCS 2936, pp. 267–277, 2004.

2 Definition and Causes of Overfitting

2.1 Definition

There is a fundamental expectation lying at the core of supervised learning: the hope that what has been learned on a set of samples will be useful in dealing with non previously seen cases. Disregarding the algorithm used, this generalization property depends on the representativeness of the set of samples. It may thus happen that the computational effort spent on obtaining a more precise fit of the samples results in an increased error on other data. This situation when the final hypothesis is "specialised" on the learning samples is called overfitting. A more formal definition is given in [3]:

Definition 1. *(Mitchell 1997) Given a hypothesis space H, a hypothesis $h \in H$ is said to overfit the training data if there exists some alternative hypothesis $h' \in H$ such that h has smaller error than h' over the training examples, but h' has a smaller error than h over the entire distribution of instances.*

2.2 Origins of Overfitting

Among the possible causes of overfitting, the "complexity" or even the sheer size of the hypothesis generated by the machine learning algorithm are often invoked as an important factor. For example in decision trees, the growth of trees allows them to specialize on "difficult" learning cases which may not be representative enough of the problem. The number of different hypotheses examined by the algorithm has also been proposed as a strong cause [4], [5]. Indeed, having many hypotheses intuitively increase the risk of randomly finding a solution that learn the training set "by heart", with limited generalization ability. These two explanations can perhaps be unified by saying that apparition of overfitting notably occurs when too much computing effort is spent.

Another independent cause is the presence of noise in the learning cases. One can then expect that too good results on the noisy data is only achievable at the cost of precision on the entire distribution.

3 Experimental Setting

As explained in the introduction, this study is made on two benchmarks:

- a symbolic regression: $f(x) = x^3 e^{-x} \cos x \sin x (sin^2 x - 1)$
- a classification problem: the 11-bit multiplexer

The learning samples have been perturbed with random noise, as shown in the parameter table (Fig. 1), in order to help in triggering overfitting, and we perform some experiments on three different implementations of GP: standard GP as it was defined by koza, GP with sizefair crossover, and standard GP in a boosting framework (see [6], [7], [8] for example). Some variants may be

Table 1. Fixed parameters table.

	regression	multiplexer
Terminals	x, random constants $\in [-1, +1]$	a_0, a_1, a_2 output selection; $d_0, d_1, d_2, d_3, d_4, d_5, d_6, d_7$ input values
Operators	+, -, *, /, sin, cos, exp, log	and, or, not, if
Learning samples	50 pairs (x, y) drawn randomly in $x \in [-1, +1]$ with $y = f(x)$	500 randomly selected configuration among the possible 2048 $(= 2^{11})$
Noise	$y = y \pm 10\%$	output value is inverted in 5% of the learning cases
Crossover proba	0.85	0.9
Copy proba	0.1	0.1
Mutation proba	0.05	0
Selection method	fitness tournament of size 5	
Creation method	half and half	

supposed to improve resistance to overfitting, or on the contrary to focus too much on noisy data as may be suspected for boosting.

As we suppose that apparition of overfitting mainly occurs when too much computing effort is spent (not excluding other possible causes), we monitor the evolution of the error across the number of generations, and we also propose to explore the effects of increasing the effort by way of increasing either the size of populations or the maximum depth of program trees, as shown in Tab. 2.

Table 2. Moving parameters table.

Moving parameter	max generation	pop size	max depth
population size	500	1000 up to 10000 (steps of 1000)	15
maximum depth	5000	500	5 up to 15

The apparition of overfitting has to be detected by measuring the evolution of E_L the error on the learning cases set and E_R the real error. On the multiplexer problem, we can compute the real error using every 2048 possible inputs, while on the regression problem we must approximate the real error by computing it on an independent test set(5000 instances). In the following we will denote as E_R either the real error (multiplexer case) or the approximation of mean square error (regression case). We notably use as an overfitting indicator the difference between the smallest error and the last generation error on the test set during a run. More precisely let n be the last generation number of the GP run, $E_R(n)$ the final real error and $E_L(n)$ the final learning error, we define $min < n$ the minimum real error generation number such that the learning error

has improved from generation m to the last: $E_R(min) = Inf\{E_R(i), 1 \leq i < n | E_L(i) > E_L(n)\}$. Then if the difference $\Delta = E_R(n) - E_R(min)$ is positive we can diagnose overfitting in the sense of Def. 1. We also use what could be called overfitting "magnitude" or amount: $\Delta / E_R(n)$, denoted as a percentage.

Nonetheless, measurements of the sensibility to overfitting should not mask the overall performance of the algorithms in terms of precision and computing time spent. Indeed let us suppose we are doing a classification task, then it is better to somewhat overfit a good hypothesis rather than being wrong all the time (a degenerated case but one that would satisfy a "no overfitting" property under Def. 1). In order to improve readability and take into account the above remark we run every experiment (under a given set of parameters for a given algorithm) 31 times, and we give several detailed results and up to two overfitting indicators:

Final error $E_R(n)$: given as average, median, min and max of the 31 runs.
Minimum error $E_R(min)$: given as average, median and min of the 31 runs.
Indicator 1: the chance that stopping GP at any time after generation *min* yields an overfitting amount less than 10%. Thus, the greater indicator 1 the better, as it can be seen as a reliability measure.
Indicator 2: the number among the 31 runs where overfitting amount at last generation exceeds a threshold of 10% (the lesser the better).

4 Results

4.1 Standard GP

Here we examine how standard GP deals with overfitting. Figure 1 shows final error range and average minimum error when increasing maximum tree depth on the regression benchmark. These plots illustrate that large tree depths, from 11 and up, are subject to overfitting on this problem as is shown by large ranges of final error. This is confirmed by looking at the detailed results in Table 3, especially indicator 1 and 2. Moreover there is no great difference in minimum final error, and it is not in favour of large depths, thus small trees on this example are both quicker to compute and more reliable.

Figure 2 plots the same data as Fig. 1 but this time when increasing population size on the regression benchmark. This shows that from 4000 individuals and up, the minimum final error (bottom of error range) is *always* above the average minimum error (plotted as a cross), indicating the presence of overfitting. Again, minimum final error is rather in favour of small computing effort, here small populations. This may seem counter-intuitive since small populations quicker runs are shown to be more precise than big runs. In fact they seem to be more reliable: bigger population runs *do find* better solutions (see minimum error columns in Tab. 5) but odds are great that they loose them in overfitting the training data before they stop, as shown by indicator 1.

On the multiplexer problem, learning reveals itself to be hard, with less than 70% of good answers: it seems that noise is simply too big for the standard GP to

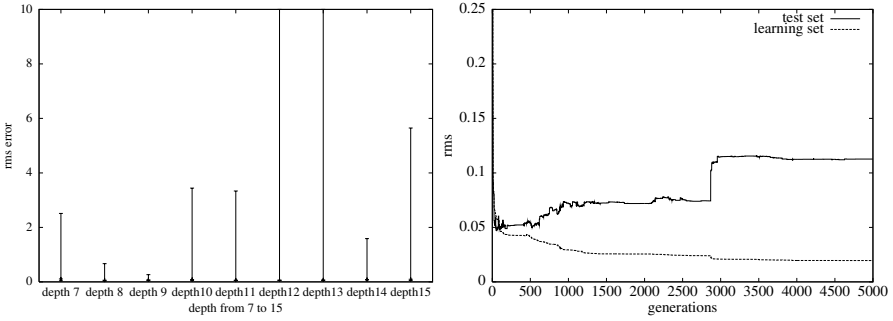


Fig. 1. On the left, final error range and average minimum error for variable maximum depth on the regression problem; on the right, evolution of error in a run (maximum depth = 11).

Table 3. Regression results for standard GP, with varying maximum tree depth.

	Final error				Minimum error			indic.1		indic.2
depth	average	median	min	max	average	median	min	≤10%	time	over-fitting
7	0,3786	0,1604	0,0406	2,5117	0,1087	0,1129	0,0401	18%	460	24/31
8	0,1918	0,1177	**0,0301**	0,6694	**0,0616**	0,0658	0,0280	21%	853	26/31
9	**0,1231**	0,1091	0,0458	**0,2662**	0,0761	0,0679	0,0420	27%	1918	22/31
10	0,4395	**0,0784**	0,0507	3,4427	0,0867	0,0514	0,0311	38%	1142	20/31
11	0,6135	0,1413	0,0524	3,3363	0,0618	0,0524	0,0336	10%	1621	25/31
12	6527604	0,3422	0,0497	65275575	0,0700	**0,0455**	0,0295	17%	1629	27/31
13	1255086	0,1348	0,0451	12550859	0,0747	0,0514	**0,0248**	11%	3940	26/31
14	0,5661	0,1508	0,0719	1,5880	0,0855	0,0576	0,0428	1%	6265	30/31
15	1,0409	0,1345	0,0518	5,6471	0,0972	0,0914	0,0424	18%	4153	28/31

Table 4. Multiplexer results for standard GP, with varying maximum tree depth.

	Final error				Minimum error			indic.1		indic.2
depth	average	median	min	max	average	median	min	≤ 10%	time	overfitting
5	0,3742	0,3789	0,3413	0,4165	0,3709	0,3789	0,3413	100%	203	0/31
6	0,3849	0,3857	0,3599	0,4067	0,3737	0,3711	0,3579	100%	223	0/31
7	0,3768	0,3809	0,3462	0,4058	0,3701	0,3682	0,3462	100%	291	0/31
8	0,3685	0,3711	0,3364	0,3931	0,3655	0,3696	0,3364	100%	288	0/31
9	0,3836	0,3894	0,3413	0,4204	0,3739	0,3838	0,3413	100%	275	0/31
10	0,3743	0,3828	0,3374	0,4048	0,3683	0,3828	0,3345	100%	398	0/31
11	0,3668	0,377	**0,2915**	0,4106	0,3601	0,3726	**0,2915**	100%	422	0/31
12	0,3668	0,3696	0,3335	0,4146	0,3591	0,3589	0,3335	100%	375	0/31
13	0,3595	0,3594	0,3364	**0,3862**	0,3533	0,3486	0,03364	97%	416	1/31
14	0,3652	0,3652	0,3423	0,4067	0,362	0,3647	0,3345	100%	491	0/31
15	**0,352**	**0,3554**	0,3091	0,3989	**0,3461**	**0,3447**	0,3091	100%	488	0/31

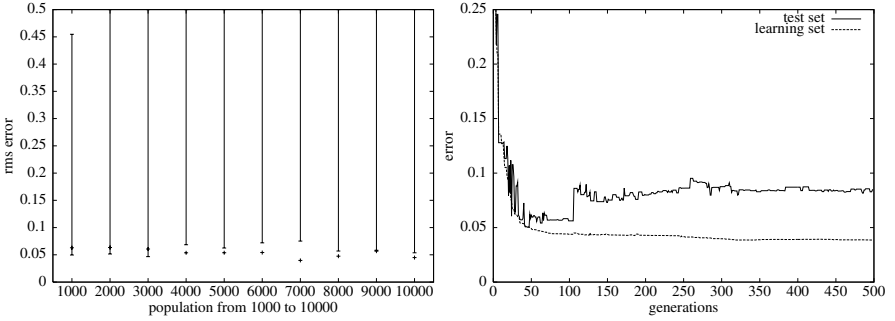


Fig. 2. On the left, final error range and average minimum error for variable population size on the regression problem; on the right, evolution of error in a run (population size = 11).

Table 5. Regression results for standard GP, with varying maximum population size.

	Final error				Minimum error			indic.1		indic.2
pop size	average	median	min	max	average	median	min	≤10%	time	over-fitting
1000	**0,1458**	**0,0727**	0,0497	**0,4547**	0,0630	0,0522	0,0332	19%	374	25/31
2000	34,071	0,1078	0,0516	339,17	0,0635	0,0522	0,0388	12%	975	26/31
3000	0,2667	0,0997	**0,0467**	1,0568	0,0607	0,0513	0,0342	8%	1514	28/31
4000	7,3961	0,7748	0,0685	38,16	0,0538	0,0432	0,0269	2%	1923	30/31
5000	6,0704	0,1451	0,0625	27,589	0,0539	0,04667	0,0308	11%	2457	29/31
6000	10,372	0,1246	0,0720	94,727	0,0542	0,0466	0,0290	2%	3090	31/31
7000	65,717	0,5166	0,0752	476,85	**0,0398**	**0,0368**	**0,0240**	3%	4699	31/31
8000	4,9093	0,2815	0,0570	43,335	0,0475	0,0453	0,0334	2%	4187	31/31
9000	0,3632	0,2425	0,0584	1,5730	0,0568	0,0518	0,0337	3%	4799	31/31
10000	8195010	0,3991	0,0537	81950101	0,0450	0,0434	0,0256	2%	6678	31/31

learn. There is only evidence of few overfitting effect and there is no significant difference between varying population size or maximum tree depths. Varying depths results are given in Tab. 4 and Tab. 6 : better results are associated to largest computing effort, with almost no overfitting but overall bad precision.

4.2 Sizefair Crossover

The sizefair crossover [2] is a crossover operator intended to slow down the growth of trees during a GP run. It does so by exchanging subtrees whose size is quite similar. Although this may be specially dedicated to counter bloat, we thought it could also be interesting against overfitting by avoiding the construction of overly big and complex solutions [9]. Tab. 7 results on the regression problem show that the overfitting phenomenon already arises at rather small depth compared to standard GP, and even if it does not increase with the maximum depth allowed, the best hypotheses seem associated to small depths, as in standard GP.

Table 6. Multiplexer results for standard GP, with varying maximum population size.

	Final error				Minimum error			indic. 1		indic. 2
pop size	average	median	min	max	average	median	min	≤10%	time	overfitting
1000	0,3282	0,3413	0,25	0,3633	0,3199	0,3125	0,25	90%	264	3/31
2000	0,3139	0,3125	0,207	0,3730	0,3093	0,3125	0,2070	89%	636	1/31
3000	0,3013	0,3086	0,25	0,3862	0,2935	0,2969	0,25	91%	1128	3/31
4000	0,2833	0,2915	0,1875	0,3403	0,2761	0,2813	0,1875	95%	1419	2/31
5000	0,2932	0,2939	0,2485	0,3530	0,2902	0,2939	0,2485	94%	2054	0/31
6000	0,2977	0,3013	0,2241	0,3413	0,2936	0,2959	0,2241	98%	2068	1/31
7000	0.2668	0,2656	0,1953	**0,3364**	0,2591	0,2578	0,1899	96%	3124	3/31
8000	0,264	**0,25**	0,1875	0,3511	0,261	**0,25**	0,1875	92%	4009	2/31
9000	0,2609	0,2656	0,1875	0,3589	0,2559	0,2539	0,2539	98%	3927	2/31
10000	**0,2587**	0,2578	**0,1748**	0,3433	**0,2526**	**0,25**	**0,1748**	93%	4937	2/31

Table 7. Regression results with sizefair crossover for varying maximum depth.

	Final error				Minimum error			indic.1		indic. 2
depth	average	median	min	max	average	median	min	≤10%	time	overfitting
7	2197429	0,1710	0,0370	20707613	0,0834	0,0718	0,0309	38%	1248	23/31
8	**0,1518**	0,0992	0,0495	**0,4253**	0,0752	0,0577	0,0341	44%	1113	18/31
9	0,3578	0,1076	0,0398	2,1091	0,0794	0,0662	0,0383	68%	2218	8/31
10	2,664	0,1480	**0,0368**	24,4401	0,0878	0,0745	0,0324	12%	4153	26/31
11	72,13	0,0903	0,0519	720,4	0,0746	0,0557	0,0388	25%	4269	24/31
12	1384440	**0,0751**	0,0458	13844403	0,0882	0,0628	**0,0274**	52%	6206	17/31
13	2578690	0,1583	0,0819	25786555	0,0932	0,0705	0,0463	12%	7132	26/31
14	16,14	0,1221	0,0542	129,1728	**0,0648**	**0,0523**	0,0381	10%	6166	30/31
15	0,1960	0,1303	0,0554	0,8068	0,0721	0,0652	0,0385	11%	6622	29/31

On the multiplexer problem (Tab. 8), sizefair crossover produces small overfitting, for better results than classical GP. Best results are obtained for small depth (looking at minimal final error). We noticed that sizefair crossover produce trees really more complex than classical crossover which allows to explore better the search space but which has a big cost in execution time.

4.3 Boosting

Boosting is a scheme that can be wrapped around any supervised learning method. Its principle lies in building several hypotheses in several rounds, under a distribution associated to the learning cases that evolves at every round so that hard to learn cases deserve more and more attention from the algorithm. Each resulting hypothesis gets a confidence value that is used to combine them together to get one final result. It is proved that this method can theoretically improve indefinitely the error on the training set when it is wrapped around a so-called weak learner algorithm (see [10]). As for overfitting, the focus on hard training cases may imply a structural tendency to overfit, although it is balanced

Table 8. Multiplexer results with sizefair crossover for varying maximum depth.

	Final error				Minimum error			indic. 1		indic. 2
depth	average	median	min	max	average	median	min	≤10%	time	overfitting
5	0,3636	0,3711	0,2813	0,4072	0,3483	0,3438	0,2813	86%	2880	5/31
6	0,3432	0,3594	**0,1504**	0,4023	0,3246	0,3281	**0,1504**	81%	5123	6/31
7	0,3462	0,3516	0,1875	0,417	0,318	0,3223	0,1875	75%	7527	11/31
8	0,3533	0,3652	0,2734	0,4063	0,3409	0,3477	0,2734	89%	10632	3/31
9	0,3552	0,3535	0,3008	0,4063	0,3418	0,3438	0,3008	92%	11601	2/31
10	0,3471	0,3516	0,25	0,4218	0,3341	0,3428	0,25	93%	15533	4/31
11	0,3532	0,3633	0,2188	0,4218	0,3376	0,3438	0,2188	84%	16011	5/31
12	0,3598	0,3516	0,2969	0,5	0,333	0,338	0,25	92%	17516	3/31
13	0,3451	0,3535	0,2539	0,3945	0,3378	0,3418	0,2539	99%	26192	3/31
14	**0,322**	**0,3247**	0,2578	**0,3828**	**0,3155**	**0,3203**	0,25	100%	25412	2/31
15	0,3549	0,3589	0,2891	0,3906	0,3406	0,342	0,2891	96%	26767	3/31

by using a committee of hypotheses [11]. Applying boosting to genetic programming has already given good results [8,6], despite the fact that GP is not known as a weak learner. For these experiments, the parameters are: population size 500, maximum generation 200, maximum depth 15, number of rounds 100, and the results are shown in Tab. 9.

Table 9. Boosting GP results.

	Final error				Minimum error			indic. 1		indic. 2
	average	median	min	max	average	median	min	≤10%	time	overfitting
Regression	0,0246	0,0246	0,0227	0,0311	0,0263	0,0258	0,0218	78%	6948	0/31
Multiplexer	0,1442	0,1450	0,1338	0,1509	0,1251	0,125	0,1138	45%	4458	31/31

An interesting phenomenon appears on the regression problem: the real error (approximated on the noise-free test set as for the other experiments) is smaller than that on the noisy learning set. Thus it seems boosting has somehow succeeded in filtering the noise, see Fig. 3. No overfitting appears and the real error is much smaller than for the other methods.

For the multiplexer problem, this gives an empirical confirmation that the error on the training set can be reduced more and more, almost until zero here. Notice that the real error rises at the end of the experience, exhibiting overfitting (Fig. 3). Nonetheless the overall precision is more than twice as good as standard GP.

5 Conclusions

From these results, we can tell that "standard GP" practitioner should not blindly rely on big populations for increased precision: this could bring the op-

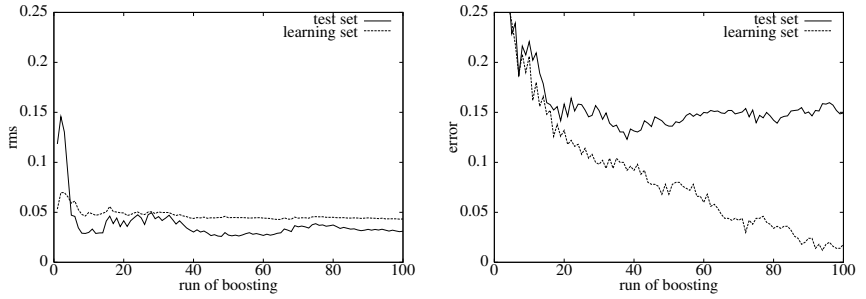


Fig. 3. Real and learn error for the boosting GP, left on the regression problem, right on the multiplexer problem.

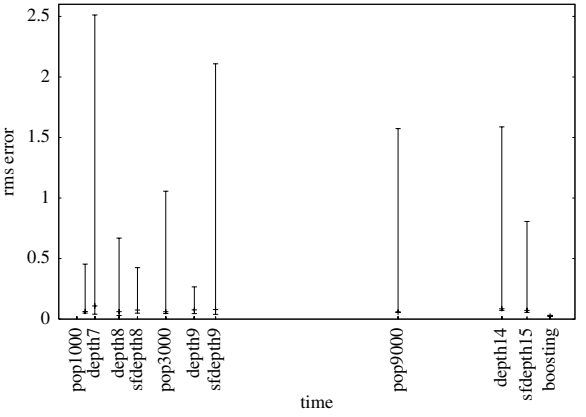


Fig. 4. Final error range and average minimum error on regression problem for a selection of best parameters and methods, plotted against computing time.

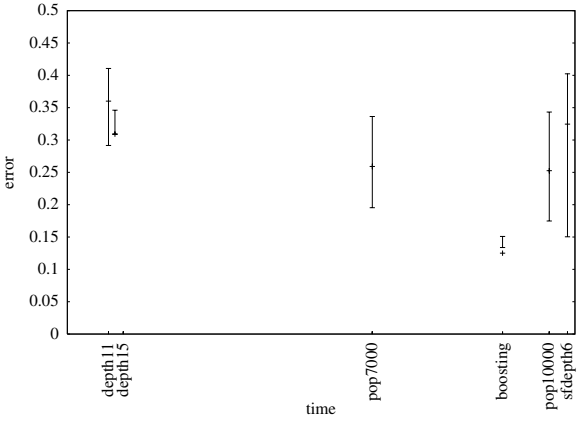


Fig. 5. Final error range and average minimum error on multiplexer problem for a selection of best parameters and methods, plotted against computing time.

posite results. This is in accordance with previous works that warned against exploring too many simple solutions. Like for the number of generations [12], computational effort on size of population does not seem to improve results. The effect of maximum depth is less strongly characterized, but results suggest there may be an optimal maximum depth above which good hypotheses are unlikely to be retrieved due to overfitting. Notice that the number of explored solutions is *constant* in the varying maximum depth experiments, discarding the previous explanation (too many solutions explored). Thus, at least for our regression problem, the "simplicity means efficiency" property seems to hold, although it is strongly questioned elsewhere (see [13]). When used in the spirit of preserving simplicity, sizefair crossover did not bring interesting results in term of overall precision, although overfitting seems to be controlled to some extent. More investigation should be done here to begin to understand the phenomenons at work. At last, GP with boosting proves remarkably good on both problems. It was able to filter part of the noise in the regression case, and even if it showed overfitting with the multiplexer, its accuracy remained much better than standard GP. The plot on Fig. 4 shows the compared performance, on the regression problem, of a selection of the best association of parameters and methods. It suggests to keep to small populations of small trees if there is not much time to invest in the search, or to bridge the gap and straightly use boosting GP. Observation of similar plot (Fig. 5) on multiplexer problem confirm that boosting was the only method (among the tested methods) to learn efficiently on this problem and suggests to use this technique for classification problem.

References

1. Koza, J.: Genetic Programming: On the Programming of Computers by Means of Natural Selection. The MIT Press (1992).
2. Langdon, W.B.: Size fair and homologous tree crossover for tree genetic programming. In: Genetic Programing and Evolvable Machines. Volume 1., Boston, Kluwer Academic Publishers (2000) 95–119.
3. Mitchell, T.M.: Machine Learning. Mc Graw-Hill (1997)
4. Ng, A.Y.: Preventing overfitting of cross-validation data. In: Proc. 14th International Conference on Machine Learning, Morgan Kaufmann (1997) 245–253
5. Cavaretta, M.J., Chellapilla, K.: Data mining using genetic programming: The implications of parsimony on generalization error. In Angeline, P.J., Michalewicz, Z., Schoenauer, M., Yao, X., Zalzala, A., eds.: Proceedings of the Congress of Evolutionary Computation. Volume 2., Mayflower Hotel, Washington D.C., USA, IEEE Press (1999) 1330–1337.
6. Paris, G., Robilliard, D., Fonlupt, C.: Applying boosting techniques to genetic programming. In: [14] (2001) 267–278.
7. Paris, G., Robilliard, D., Fonlupt, C.: Genetic programming with boosting for ambiguities in regression problems. In: [15] (2003) 183–193.
8. Iba, H.: Bagging, boosting, and bloating in genetic programming. In: [**?**] (1999) 1053–1060.
9. Langdon, W.B.: Combining decision trees and neural networks for drug discovery. In: [17] (2002) 60–70.

10. Y. Freund, R.S.: A short introduction to boosting. Journal of Japanese Society for Artificial Intelligence (1999) 771–780.
11. Y. Freund, Y.M., Schapire, R.: Why averaging classifiers can protect against overfitting. In: Proceeding of the Eighth international WOrkshop on Artificial Intelligence and Statistics (2001).
12. Luke, S.: When short runs beat long runs. In: Proceedings of GECCO-2001, Morgan Kaufmann (2001) 74–80.
13. Domingos, P.: The role of Occam's razor in knowledge discovery. Data Mining and Knowledge Discovery **3** (1999) 409–425.
14. Collet, P., Fonlupt, C., Hao, J.-K., Lutton, E. and SChoenauer, M., editors. *Artificial Evolution, 5th International Conference, Evolution Artificielle, EA 2001*, Le Creusot, France, 2001, LNCS 2310, Springer.
15. Ryan, C., Soule, T., Keijzer, M., Tsang, E., Poli, R. and Costa, E., editors. *6th European Conference, EuroGP 2001*, LNCS 2610, Springer.
16. Banzhaf, W., Daida, J., Eiben, A., Garzon, M., Honavar, V., Jakiela, M. and Smith, R., editors. *Proceedings of the Genetic and Evolutionary Computation Conference*, Orlando, Florida, USA, July 1999, Morgan-Kaufmann.
17. Foster, J., Lutton, E., Miller, J., Ryan, C. and Tettamanzi,A., editors. *5th European Conference, EuroGP 2001*, LNCS 2278, Springer.

An Agent Model for First Price and Second Price Private Value Auctions

A.J. Bagnall and I. Toft

School of Information Systems
University of East Anglia
Norwich, NR47TJ
England
ajb@sys.uea.ac.uk

Abstract. The aim of this research is to develop an adaptive agent based model of auction scenarios commonly used in auction theory to help understand how competitors in auctions reach equilibria strategies through the process of learning from experience. This paper describes the private value model of auctions commonly used in auction theory and experimentation and the initial reinforcement learning architecture of the adaptive agent competing in auctions against opponents following a known optimal strategy. Three sets of experiments are conducted: the first establishes the learning scheme can learn optimal behaviour in ideal conditions; the second shows that the simplest approach to dealing with situations of uncertainty does not lead to optimal behaviour; the third demonstrates that using the information assumed common to all in private value model allows the agent to learn the optimal strategy.

1 Introduction

The Internet has meant that auctions have become a crucial means of conducting business, in both the consumer to consumer market (through such companies as ebay) and the business to business arena through companies such as FreightTraders Ltd [2] and FreeMarkets Inc [1]. Research indicates that the B2B market will escalate from $43 Billion in 1998 to $1 Trillion in 2003 [12], and much of this business will be conducted through auctions, which are rapidly replacing request for quotes as the preferred means of business procurement. This growth in online auctions has created an increasing interest in the study of the behaviour of bidders and the effect of auction structure on the choice of bidding strategy [3, 5].

Broadly speaking, some of the key questions in studying auctions are: what are the optimal strategies for a given auction structure; how do agents reach an optimal strategy through experience in the market; how does the restriction of information affect the ability to learn optimal behaviour; and what market structures lead to the best outcome in terms of efficient allocation of resources and maximizing the revenue for the seller?

P. Liardet et al. (Eds.): EA 2003, LNCS 2936, pp. 281–292, 2004.

These issues can be addressed theoretically through auction theory, experimentally through simulated auction environments with human volunteers competing in a controlled environment or by computer simulation with autonomous agents either following fixed strategies [9] or learning through interaction [6, 15,13]. All approaches have a role to play in understanding auctions. Auction theory can tell us about the theoretical behaviour of agents in auctions and under certain certain constrained models can derive the optimal behaviour [11]. However, real world auctions rarely satisfy these constraints, and understanding of actual behaviour often requires observation and analysis of patterns of actual bidding. Experimental studies can help increase our understanding of how competitors behave in auctions through simulations of different auction structures with imaginary goods [8]. However, experimental studies suffer from drawbacks, chief among these being the difficulty in collecting an adequate amount of data and the problem with finding subjects with suitable experience in similar scenarios. Computer simulations offer an additional information source for addressing the issues that do not necessarily suffer the same problems.

This paper examines behaviour of autonomous adaptive agents (AAA) in simulations of common auction scenarios with a single seller. Recently, there has been much interest in studying the behaviour of agents in double auctions (auctions with multiple sellers and buyers) and in models of more constrained markets [4]. For example, Cliff [5] and Hu [10] have examined how alternative agent structures perform and the effect of market structure on performance. The aim of this research is to investigate how AAA behave in 'one-way' auction environments. The potential benefits of this project are: an economic test bed to examine effect on behaviour of alternative auction structures; an alternative mechanism to verify theoretical behaviour in auctions and compare with real world experimental studies; a new arena in which to study the evolution of multi-agent systems.

Although agent simulations offer a means of overcoming some of the problems with human experimentation, in order to be of use in the analysis of behaviour it is important to demonstrate that the agents can learn to perform well (in terms of meeting their objectives) and that they can learn the complexity of strategy exhibited by human competitors. Our approach to developing AAA for auctions is to start with a constrained environment where a single adaptive agent competes against agents that adopt the symmetric equilibrium strategy. This has the benefit that we can deduce the optimal strategy for the adaptive agent and thus assess performance and study the effects of alternative learning mechanisms on the quality of the strategy learnt.

Section 2 describes the auction environment. We adopt the private values model commonly used in auction theory and experimental studies [8]. The alternative agent learning mechanisms employed are described in Section 3. The learning mechanism used is similar to that employed by Cliff's ZIP agents [6], but with certain differences necessitated by the nature of the problem. The results for three alternative learning schemes are presented in Section 4. The first set of results in Section 4.1 demonstrate that the learning mechanism employed is

able to learn the optimal strategy when provided with perfect information about what it should have bid after the event. This scenario is not realistic, since some information is usually withheld from the agent. However, it is desirable that any learning agent should be able to learn optimality in the complete information case, and the tests described in Section 4.1 give us a means of testing whether the agent meets the minimum level of performance we require. The second and third set of results contrast two approaches to learning when faced with uncertainty and find that the agent needs to utilise the information it is assumed to be party to under the PVM in order to reach the optimal strategy. Finally, the conclusions and future directions are discussed in Section 5.

We use the terms agents and bidders interchangeably. We use N to denote the number of agents including the adaptive agent and n to denote the number of non-adaptive agents (i.e. $n = N - 1$). We subscript the variables associated with the agents with i and those relating to a particular auction from a sequence of auctions with j, although we drop the second subscript unless explicitly required.

2 The Simulated Auction Model

2.1 Private Value Model

We use the private value model (PVM) proposed by Vicary [14] in the experimentation described in this paper. The PVM can be described in the following way. In an auction of N interested bidders, each bidder i has a valuation x_i of the single object. Each x_i is an observation of an independent, identically distributed random variable X_i with range $[0, \omega]$ (ω is the universal maximum price) and distribution function F.

The benefits of this model are that for certain auction mechanisms and assumptions (described in Section 2.2) there is provably optimal behaviour. Thus we have a clear way of gauging the performance of adaptive agents and assessing under what conditions learning is most efficient. This is a necessary condition to studying more interesting scenarios where the assumptions about the competitors behaviour do not hold true.

2.2 Auction Mechanisms

The four most used and studied auction forms are:

1. The open ascending price or *English* auction, where bidders submit increasing bids until no bidders wish to submit a higher bid;
2. The open descending price or *Dutch* auction, where the price moves down from a high starting point until a bidder bids, at which point the auction terminates.
3. The first-price sealed-bid auction (FPSB), where each bidder submits a single bid, the highest bidder gets the object and pays the amount he bid;
4. A second-price sealed-bid auction (SPSB), where each bidder submits a single bid, the highest bidder gets the object and pays the second highest bid.

Under the PVM, a Dutch auction is strategically equivalent to FPSB auction and an English auction is equivalent to a SPSB auction (strategic equivalence implies that for every strategy in one auction there is an equivalent strategy in the other auction that would have identical outcomes). Hence we restrict our attention to FPSB and SPSB auctions. The FPSB and SPSB simplify the bidding mechanism, since each agent can submit at most one bid for any auction. We denote the bids of the agents b_i. Each agent forms its bid with a bid function

$$\beta_i : [0, \omega] \rightarrow \Re_+, \qquad \beta_i(x_i) = b_i.$$

The set of all bids for a particular auction is denoted $B = \{b_1, b_2, \ldots, b_N\}$. For both FPSB and SPSB, the winning agent, w, is the highest bidder,

$$w = \arg\max_{i \in N}(b_i \in B).$$

The price paid by the winning agent, p, is dependent on auction structure. In a FPSB, the price paid is the highest bid,

$$p = \max_{i \in N}(b_i \in B).$$

In a SPSB, the price paid is the second highest bid,

$$p = \max_{i \in N, i \neq m}(b_i \in (B - b_m)).$$

where b_m is the largest bid, i.e. $b_m = \max_{i \in N}(b_i \in B)$.

2.3 Optimal Strategies

An agent's profit (or reward) is

$$r_i(x_i) = \begin{array}{ll} x_i - p & \text{if } i = w \\ 0 & \text{otherwise.} \end{array} \tag{1}$$

All agents are risk neutral, i.e. they are attempting to maximize their profit. The optimal strategy is the bid function that will maximize the profit of the agent. A symmetric equilibrium is a Nash equilibrium in which all bidders follow the same strategy. Hence a bid function, B^*, is a symmetric equilibrium if any one agent can do no better (in terms of maximizing expected reward) than follow B^* if all other agents use B^*.

First Price Sealed Bid. The symmetric equilibrium strategy in a FPSB for agent i is the expected value of the largest of the other agents' private values, given that the largest of these values is less than the private value of agent i, i.e. the agent should bid as high as all the other agents' values (not bids) without exceeding its own value. More formally, let $Z_1, Z_2, \ldots Z_n$ be the order statistics of the agent values $X_1, X_2, \ldots X_n$. The symmetric equilibrium strategy for agent N is

$$\beta(x_N) = E(Z_n | Z_n < x_N) \tag{2}$$

The optimal strategy is dependent on the form and assumed commonality of the value distribution function F and the independence of the bidders' values. When F is a uniform distribution on $[0, 1]$ the symmetric equilibrium strategy is

$$\beta(x_i) = \tfrac{N-1}{N} x_i$$

Second Price Sealed Bid. The symmetric equilibrium strategies in a SPSB auction are given by

$$\beta(x_N) = x_N. \tag{3}$$

The optimal strategy for a SPSB auction is independent of the form of the distribution function F and does not require that all bidders have the same value function. Proofs and a more complete description of auction formats are given in [11].

2.4 Auction Simulation Structure

The agents compete in a series of k auctions indexed by $j = 1, \ldots, k$. For any auction j, each bidder is assigned a value $x_{i,j}$ by sampling F. Prior to bidding, each bidder is aware of:

1. The realisation $x_{i,j}$ of X_i;
2. The distribution function common to all bidders, F;
3. The universal maximum value, ω;
4. The number of competitive bidders, N.

Once the auction is complete, the bidder is informed of whether they were the winner and the price the winner must pay. No other information relating to the other agents' bids is made available. As discussed in Section 3, this restriction of information affects an agent's ability to learn a good strategy. The agent is also unaware of the number of auctions in an experiment.

We use a uniform distribution on $[0, 1]$ for F and fix N for the duration of an experiment. All experiments involve a single adaptive agent competing against n non-adaptive agents following the optimal strategy relevant to the auction structure (Equations 2 and 3).

3 The Adaptive Agent Structure

With no loss of generality we assume the adaptive agent is bidder N. An adaptive agent attempts to maximize its expected reward, $E(r_N)$ where r_N is as defined in Equation 1, by finding an optimal strategy or bid function B^*. Under the model described in Section 2 the optimal strategy is as given in Equation 2 for FPSB and Equation 3 for SPSB auctions. The key issue is how the agent uses the information made available to it after the auction is complete in order to alter its behaviour to that closer to the optimal. Hence the agent needs to be able to assess or estimate how good a particular bid was and how that information

should be employed to alter behaviour. We restrict the class of bid functions to linear functions of the form

$$B_N = x(1 - \mu_N),$$

where μ_N represents the percentage margin below its value that the agent bids at. This form allows us to simply relate the behaviour of the adaptive agent to the non-adaptive agents and measure how close the agent is to adopting the optimal strategy. The optimal strategies (and hence the strategies followed by the non-adaptive agents) are recreated by setting $\mu = 0$ for SPSB and $\mu = 1/N$ for FPSB. Thus the problem of learning a strategy is now the problem of learning the best margin μ based on the information available.

After an auction has been concluded the agent attempts to calculate the optimal bid it could have made, o_j, and hence the margin it should have used to bid in the auction, $d_{N,j}$, in order to maximize profit. It uses the estimate of the desired margin for auction j to adjust its margin using the Widrow-Hoff update rule

$$\mu_{N,j+1} = \mu_{N,j} + \sigma_{N,j}, \tag{4}$$

$$\sigma_{N,j} = \beta(d_{N,j} - \mu_{N,j}). \tag{5}$$

We drop the auction subscript j for the remainder of this Section for clarity. $b_1, ..., b_n$ are the bids of the n other agents and $y_1, ..., y_n$ are the bids sorted into ascending order, i.e. the observed values of the n bid order statistics $Y_1, ..., Y_n$. The optimum bid, o, is defined as the bid which would have maximised the agents reward r_N, given all the information about the other bids.

In auctions where the highest bid wins, the optimal bid is always equal to a small amount above the bid of the highest other bidder, as long as that bid is below the winning bid. If we ignore the small increment, then when $y_n < x_N$ the optimal bid is the largest of the other bids, $o = y_n$. Given the optimal bid, the optimal margin is

$$d_N = 1 - \frac{o}{x_N}. \tag{6}$$

There are two crucial issues to address in designing a learning mechanism for auction environments. The first issue is what the agent should do when the price paid is higher than the agent's value for that auction. In this scenario it is unclear what the optimal bid should have been, because the maximum attainable reward of zero is achieved by any bid less than the agent's value. Our initial approach is simply to set the optimal bid to the previous actual bid, thus not altering the bid margin. Although reasonable in the context of a single auction, this approach ignores the potential of using these bids to learn about the strategy of the other agents. However, as we demonstrate in Section 4.1, this approach is sufficient to learn optimal behaviour if the agent can calculate o accurately in all cases when $p \leq x_N$. The second issue is what to do when the optimal bid cannot be calculated exactly. Two approaches to this problem are presented in Section 3.1.

3.1 Estimating the Optimal Bid

Whether the agent can calculate the optimal bid exactly is dependant on the information available to the agent, which itself is dependant on the auction structure. In order to calculate o in a FPSB and SPSB auction, the agent needs to know the largest bid of the other agents. One of the key differences between FPSB and SPSB is the circumstances under which the agent does not know with certainty what the optimal bid should have been. The two cases where the agent faces this situation of incomplete information are

1. **Case 1**: in a FPSB when the agent wins ($p = b_N$, $w = N$) the agent is not told any of the other agents' bids. The adaptive agent only knows that the highest bid of the other agents, and hence the optimal bid, is in the interval $[0, p)$.
2. **Case 2**: In a SPSB when the agent loses ($p \leq b_N$, $w \neq N$), the agent is only informed of the second highest bid of the other agents, y_{n-1}, is less than or equal to p. The agent does know that the highest bid is on the interval $[p, \omega)$, and that its optimal bid is in the interval $[p, x_N]$.

In all other scenarios when $p < x_N$ the agent is able to determine exactly what the best bid would have been, and adjust its behaviour accordingly. In both cases of incomplete information, the agent is interested in estimating y_n, the highest bid (not counting the agents own) and hence the optimal bid. There are two approaches we investigate for estimating the optimal bid in the two cases of incomplete information.

1. *Naive estimation*: The agent simply guesses a value on the interval, thus in Case 1 the agent estimates y with a random value on $[0, p)$ for case 1 and $[p, \omega)$ in Case 2.
2. *Estimate based on assumptions*: Based on the PVM assumptions about the behaviour of the other agents, the agent can estimate the expected value of the relevant order statistics. This method is described in section 3.2

3.2 Estimating from Order Statistics

If we assume that each $b_1, ..., b_n$ is an independent identically distributed ($i.i.d$) random variable, then we can define the problem facing the agent as calculating or estimating the expected value of the n^{th} order statistic conditional on the information provided by the auction. Under the PVM each bidders' value is an observation of $i.i.d.$ random variable, hence the assumption is equivalent to assuming that each agent is following an identical bidding strategy (i.e. the model is symmetric) and that this strategy remains constant from auction to auction (i.e. the agents are non-adaptive). These assumptions are common in auction theory.

Let f_b be the common density function of the bids and F_b the distribution function. The distribution of the n^{th} order statistic is

$$g_n(y_n) = n \cdot f_b(y_n) \cdot [F_b(y_n)]^{n-1}$$

Case 1: FPSB, $p = b_i$, $w = N$. In a FPSB auction where the agent wins (case 1), the optimal bid is the highest of the other bids, i.e. y_n. The distribution of the n^{th} order statistic conditional on $y_n < p$ is

$$g_n(y_n|y_n < p) = \frac{f_b(y_n)[F_b(y_n)]^{n-1}}{\int_{t=-\infty}^{p} f_b(t)[F_b(t)]^{n-1}dt}$$

and the agent wishes to calculate $E(y_n|y_n < p)$. See [7] for more background on conditional distributions of order statistics. Suppose we know f_b is uniform on $[0, \omega]$, then

$$g_n(y_n|y_n < p) = \frac{n[y_n]^{n-1}}{p^n}$$

and

$$E(y_n|y_n < p) = p\frac{n}{n+1}.$$

Hence, if the agent assumes the distribution of y_n is uniform, then the estimate of the optimal bid is

$$o = b_i\frac{n}{n+1}.$$

Case 2: SPSB, $w \neq N$. In a SPSB auction where the agent loses, we have two possible situations. Firstly when $p = b_N$, i.e. the adaptive agent's bid was the second largest. There are two pieces of information available, firstly that $y_n > p$ and secondly that $y_{n-1} < p$.Secondly when $p > b_N$, i.e. the adaptive agent's bid was not the second largest. The agent can now infer that $y_{n-1} = p$. Both scenarios require the distribution of the n^{th} order statistic conditional on the $Y_{(n-1)}$. From [7], the conditional distribution is

$$g_n(y_n|y_{n-1}) = \frac{f_b(y_n)}{[1 - F_b(y_{n-1})]}.$$

If we assume f_b is uniform on $[0, 1]$,

$$g_n(y_n|y_{n-1}) = \frac{1}{[1 - y_{n-1}]}$$

and

$$E(y_n|y_{n-1}) = \frac{1 + y_{n-1}}{2}.$$

If the agent assumes the distribution of y_n is uniform on $[0, 1]$ we can generalise the two scenarios so that the estimate of the optimal bid is

$$o = \frac{x_i + p}{2}.$$

4 Results

All experiments involve a single adaptive agent competing against 15 non-adaptive agents following the optimal strategy, over 10000 auctions. A learning rate of $\beta = 0.1$ was used.

4.1 Agents Learning with Sufficient Information to Estimate the Optimal Bid

Before continuing to examining performance in environments where the agent is confronted with *a posteriori* incomplete information, it is important to verify that the learning mechanism employed can at least converge to optimality when the agent is allowed to know the best it could have done. These "cheating" agents provide us with reassurance that the learning mechanism proposed in Equations 4, 5 and 6 will tend toward the optimal strategy with complete *a posteriori* information.

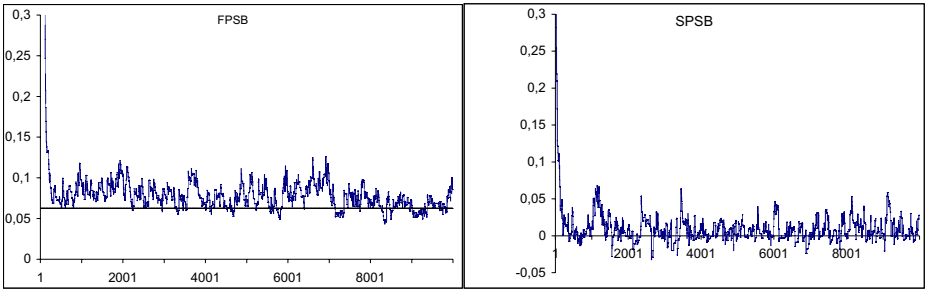


Fig. 1. "Cheating" adaptive agent bid margin for FPSB and SPSB auctions. The straight line for FPSB is the optimal bidding margin.

Figure 1 shows the bid margin $\mu_{N,j}$, of a "cheating" adaptive agent over a single experiment of 10000 auctions for FPSB and SPSB respectively. The optimal margin for FPSB is $1/N$ and for SPSB the optimal margin is 0. Table 1 shows the total profit made by the adaptive agent and the 15 non-adaptive agents over the last 5000 auctions in a run of 10000.

Table 1. Total profit over the last 5000 auctions for the cheating agent against 15 non-adaptive optimal agents in FPSB and SPSB auctions

FPSB					SPSB				
Adaptive Agent	Non-Adaptive Agents				Adaptive Agent	Non-Adaptive Agents			
Profit	Min	Max	Mean	Std Dev	Profit	Min	Max	Mean	Std Dev
17.93	16.70	20.73	18.59	0.98	19.44	16.59	21.81	18.51	1.39

Figure 1 indicates that there may be some minor deviation from optimality with FPSB auctions. However, this small deviation is not significant enough to effect the profits of the adaptive agent which, as can be seen in Table 1. For both FPSB and SPSB the adaptive agent profit is within the range of the profit of the other agents, and is not significantly less than the mean. The cheating agent is quickly learning the optimal strategy for SPSB.

Hence the learning mechanism describe in Section 3 converges to the symmetric equilibrium strategy when provided with complete information. The next

two sets of experiments are designed to see how the adaptive agent performs in the more realistic scenario of incomplete information, which requires that the optimal bid be estimated rather than known.

4.2 Agents Using Naive Estimation

Figure 1 and Table 1 show that the agent can learn an optimal strategy for FPSB and SPSB when given the optimal bid in the case when $p \leq x_N$. However, in both auction types this information is not always available. Section 3.1 describes two approaches adopted faced with this uncertainty. The most obvious strategy when faced with uncertainty as to the optimal bid, the naive approach, is simply to guess a value on the interval in which the optimal bid must lie. Figure 2 shows that adopting this approach does not lead to the optimal strategy. This point is reinforced by Table 2 which clearly show the adaptive agent is making significantly less profit than the adaptive agents and is doing markedly worse in FPSB than SPSB.

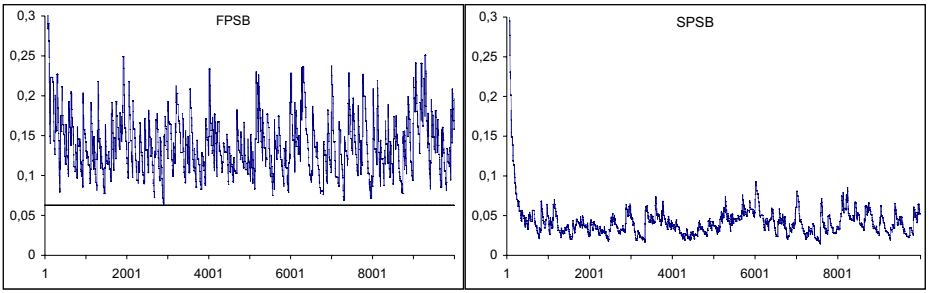


Fig. 2. Naive adaptive agent bid margin FPSB and SPSB auctions

Table 2. Total profit over the last 5000 auctions for the naive agent against 15 non-adaptive optimal agents in FPSB and SPSB auctions

FPSB					SPSB				
Adaptive Agent	Non-Adaptive Agents				Adaptive Agent	Non-Adaptive Agents			
Profit	Min	Max	Mean	Std Dev	Profit	Min	Max	Mean	Std Dev
10.77	17.20	20.95	19.20	1.19	16.74	16.96	20.62	19.06	1.11

Failure to reach the optimum strategy can be explained by the uniform sampling method of estimating the optimal bid. In a FPSB, when the agent wins the estimate is made on the range $[0, p]$, and will on average be $\frac{p}{2}$. However, the expected value of the second highest bid is $\frac{N-1}{N}p$, hence the agent is repeatedly increasing it's bid margin by too much, hence losing auctions it could have won with a profit. In SPSB the estimate is uniform on the range $[p, x_N]$. The improved performance in SPSB shown by the greater profit is due to the fact that the range $[p, x_N]$ will be smaller than the range $[0, p]$, hence the bias introduced by the random estimation will be less.

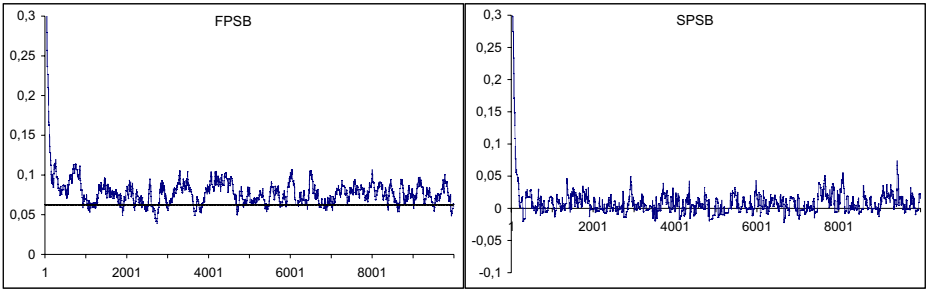


Fig. 3. Estimating adaptive agent bid margin for FPSB and SPSB auctions

Table 3. Total profit over the last 5000 auctions for the estimating agent against 15 non-adaptive optimal agents in FPSB and SPSB auctions

FPSB					SPSB				
Adaptive Agent	Non-Adaptive Agents				Adaptive Agent	Non-Adaptive Agents			
Profit	Min	Max	Mean	Std Dev	Profit	Min	Max	Mean	Std Dev
17.86	16.51	20.66	18.62	1.12	19.13	15.81	22.81	19.02	2.06

4.3 Agents Using Estimation Based on Assumptions

Figure 3 and Table 3 show the results for the estimating agent. The results indicate that the adaptive agent is doing as well as the non-adaptive agents in terms of profit. The agent margin converges towards the optimal, with wider variation for FPSB than SPSB. This demonstrates that the learning mechanism is able to utilise the information assumed available under the PVM to learn an optimal strategy.

5 Conclusions and Future Directions

This paper has introduced an agent model of common auction scenarios, FPSB and SPSB, using the private values model and examined how an autonomous adaptive agent using a basic learning mechanism can learn to behave optimally in terms of evolving towards a symmetric equilibrium solution. We think it it is a necessary prerequisite that any learning mechanism should be at least able to learn an optimal strategy given the necessary information to do so. We have demonstrated that the learning mechanism described in Section 3 is sufficient for the agent to make as much profit as the non-adaptive agents in both FPSB and SPSB auctions. In order to cope with the uncertainty inherent in accurate auction simulations, the agent using this learning mechanism must estimate in certain situations dependent on auction structure. We have presented two techniques for this estimation, a naive approach which was found to be insufficient and an estimation technique built on the assumptions of the PVM which quickly converged to optimality. This work will be extended through the following stages.

5.1 Future Work

We will continue to experiment with the PVM in order to find the most robust learning mechanism for scenarios with a known optimal strategy. This process will include: a thorough analysis will be conducted of the apparent slight deviation from optimality suggested by Figure 1; an extension of the learning mechanism that uses the information available in situations where the price paid is above the agents value; an assessment of how the learning mechanisms perform with alternative value distributions; an assessment of how the learning mechanisms perform when the number of adaptive agents can vary during an experiment; relaxation of the PVM assumptions. Having developed learning mechanisms best able to cope in situations where optimality can be measured, we will then examine how these mechanisms perform when the assumptions are no longer met and begin an investigation into behaviour of multi-adaptive agent models of auction scenarios.

References

1. FreeMarkets Inc. `http://www.freemarkets.com/`.
2. FreightTraders Ltd. `http://www.freight-traders.com/`.
3. A.Chavez, A. Moukas, and P. Maes. Challenger: A multi-agent system for distributed resource allocation. In W. Lewis Johnson and Barbara Hayes-Roth, editors, *Proceedings of the First International Conference on Autonomous Agents (Agents'97)*, pages 323–331, New York, 1997. ACM Press.
4. A. J. Bagnall and G. D. Smith. An adaptive agent model for generator company bidding in the uk power pool. In *Lecture Notes in Computer Science 1829, Artificial Evolution*, pages 191–203, 1999.
5. D. Cliff. Evolution of market mechanism through a continuous space of auction-types. In *Proceedings of the 2002 Congress on Evolutionary Computation (CEC2002)*, pages 206–209, 2002.
6. D. Cliff and J. Bruten. Zero is not enough: On the lower limit of agent intelligence for continuous double auction markets, 1997.
7. H. A. David. *Order Statistics*. John Wiley and Sons, 1970.
8. D. Friedman and J. Rust, editors. *The Double Auction Market*. Perseus Publishing, 1991.
9. D. Gode and S. Sunder. Allocative efficiency of markets with zero intelligence traders. *Journal of Political Economy*, 101:119–137, February 1993.
10. Junling Hu and Michael P. Wellman. Multiagent reinforcement learning: theoretical framework and an algorithm. In *Proc. 15th International Conf. on Machine Learning*, pages 242–250. Morgan Kaufmann, San Francisco, CA, 1998.
11. V. Krishna. *Auction Theory*. Harcourt Publishers Ltd, 2002.
12. M. Sawhney and S. Kaplan. The-b-to-b-boom, let's get vertical. *Business 2.0*, September 1999.
13. G. Tesauro and R. Das. High-performance bidding agents for the continuous double auction. In *Proceedings of the Third acm Conference on Electronic Commerce*, pages 206–209, 2001.
14. W. Vickery. Counterspeculation, auctions, and competitive sealed tenders. *Journal of Finance*, 16:8–37, 1961.
15. Michael P. Wellman and Junling Hu. Conjectural equilibrium in multiagent learning. *Machine Learning*, 33(2-3):179–200, 1998.

A Clustering Based Niching EA for Multimodal Search Spaces

Felix Streichert[1], Gunnar Stein[2], Holger Ulmer[1], and Andreas Zell[1]

[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/welcome.html
[2] Inst. 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

Abstract. We propose a new niching method for Evolutionary Algorithms which is able to identify and track global and local optima in a multimodal search space. To prevent the loss of diversity we replace the global selection pressure within a single population by local selection of a multi-population strategy. The sub-populations representing species specialized on niches are dynamically identified using standard clustering algorithms on a primordial population. With this multi-population strategy we are able to preserve diversity within the population and to identify global/local optima directly without further post-processing.

1 Introduction

In this paper we describe the Clustering Based Niching method (CBN) for Evolutionary Algorithms (EA) to identify multiple global and local optima in a multimodal search space. The basic idea was to transfer the biological concept of non-interbreeding species living in separated ecological niches into EA to preserve diversity in the EA population. One of our goals was to make the CBN 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 CBN on the EA mechanism and the fitness function is as small as possible. Also, we aimed for the CBN to have as few critical parameters as possible to allow black box optimization.

In biology species have in common that they don't interbreed anymore. Taking into account that species in different ecological niches don't compete for the same resources, but evolve independently of each other. Using this property should be the most natural way to create and maintain diversity in an EA population.

Our method to artificially create species in a primordial EA population is to search for groups or clusters of EA individuals in the search space, which will naturally occur due to the general convergence behavior of EA by the means of clustering algorithms. These clusters can then be separated into isolated sub-populations. Such a sub-population would represent a single species. Therefore,

P. Liardet et al. (Eds.): EA 2003, LNCS 2936, pp. 293–305, 2004.

individuals of different sub-populations do not compete with each other and are not be allowed to interbreed. The individuals of a single sub-population on the other hand do compete and breed like in any traditional EA and each sub-population behaves like an EA converging to a global/local optimum. With additional mechanisms to split sub-populations if necessary, dynamic specialization can occur. Merging species that become too similar will allow only one sub-population per niche.
In Sec. 2 we give an overview over some niching EAs and multi-start algorithms for multimodal search spaces. The algorithm for the Cluster Based Niching method is given in Sec. 3, and in Sec. 4 results are presented comparing the CBN with an Evolution Strategy (ES) to Multi-Start Hill-Climbing and an Evolution Strategy with Fitness Sharing on several two-dimensional benchmark functions and one n-dimensional benchmark function.

2 Related Work

2.1 Niching Evolutionary Algorithms

Niching Evolutionary Algorithms (NEAs) try to identify as many optima as possible, by preserving diversity within the EA population and by using this diversity as resource for exploratory crossover. Most NEAs are based on the idea to preserve the diversity within the population by altering the EA operators to prevent premature convergence to one optimum, like Fitness Sharing [1], Crowding [2] and Deterministic Crowding (DC) [3] and Tagging [4, Sec. 6.2.3]. Other NEAs use a multi-population approach like the Multinational (MN) GA [5,6] and the Forking GA [7], which divide a primordial global EA population into sub-populations with reduced interaction to preserve diversity above the level of sub-populations.
The most common and best studied niching method, based on prevention of premature convergence, is Fitness Sharing by Goldberg and Richardson [1]. Sharing adds a penalty to the fitness $\Phi(x_i)$ of all individuals x_i in the population P of size μ that are too similar to another individual in respect of a given metric $\|x_i, x_j\|$ between two individuals and the sharing distance σ_{share}:

$$\Phi'(x_i) = \frac{\Phi(x_i)}{\sum_{j=0}^{\mu} sh(x_i, x_j)} \tag{1}$$

$$sh(x_i, x_j) = \begin{cases} 1 - \left(\frac{\|x_i, x_j\|}{\sigma_{share}}\right) & : \text{ if } \|x_i, x_j\| \leq \sigma_{share} \\ 0 & : \text{ else} \end{cases} \tag{2}$$

This penalty is intended to lead to an even distribution of the EA population around promising areas in the search space and to prevent the overall convergence of the whole population on one optimum. Unfortunately, there are several drawbacks:

- There is a fixed σ_{share} for all individuals. For multimodal optimization all optima in the search space must be nearly equidistant.

- To set σ_{share} *a priori* knowledge about the distribution of the optima and their fitness is required, see for example [8].
- The population size is also dependent on *a priori* knowledge of the number of optima, an example is given in [9].
- Using no or a bad scaling function on the fitness Φ can prevent the individuals from finding the actual peak of the niche [10]. Alternatively, an additional hill-climbing post-processing algorithm is needed to improve the results of sharing, see [11].

Yin and Germay [12] introduced an Adaptive Clustering Algorithm (ACA) to avoid the *a priori* estimation of σ_{share}. Instead of the original sharing function of equations (1) and (2) they used equations (3) and (4).

$$\Phi'(x_i) = \frac{\Phi(x_i)}{S(x_i)} \tag{3}$$

$$S(x_i) = n_{c_j} - n_{c_j} \cdot \left(\frac{\|x_i, c_j\|}{2 \cdot d_{max}} \right)^{\alpha} \text{, with } x_i \in C_j \tag{4}$$

where α is a constant, $\|x_i, c_j\|$ is the distance between the individual x_i and the centroid c_j of the cluster C_j which x_i belongs to, and n_{c_j} is the number of individuals associated with cluster C_j. The ACA is able to get rid of σ_{share} by applying the sharing function only within identified clusters, but introduces two additional variables d_{min} and d_{max}, which define the minimum and the maximum radii of a cluster, which need to be set to reasonable values. Although Gan and Warwick [13] extended the ACA by retaining the niches found by the ACA, they did not implement a multi-population approach.

One multi-population approach is the Forking GA [7], which monitors a global GA population for the occurrence of dominating schemes in the population. If such a scheme occurs, a sub-population is created which includes all individuals that meet the scheme and this sub-population continues the search in a subspace of the original space reduced by the fixed elements of the scheme. The global population continues the search in the original space but the already identified schemes are forbidden.

The Multinational GA groups the individuals together into sub-populations called nations [5,6]. Each nation consists of individuals which are not separated by valleys of lower fitness in the search space. This is tested with the 'hill-valley' function by evaluating several interpolating individuals between each individual of the GA population. A valley would obviously separate different local optima and therefore different nations. The Multinational GA is distinguished from other NEAs by the fact that it is the only NEA known to us which actually identifies the optima on the basis of the nations residing on them, without any further post-processing.

2.2 Multi-start Algorithms

Although the Multi-Start Hill-Climber (MS-HC) represents a rather primitive technique it can become useful in simple and low dimensional multimodal search

spaces. The MS-HC performs several local HC searches in parallel, each one initialized randomly. A single HC will most likely converge to different global/local optima depending on the initialization.
There is a high probability that several HC will converge toward the same local optimum. To prevent this, Törn introduced the LC (multiple local searches with clustering) algorithm [14]. After a number of parallel local search HC steps, he applies a clustering algorithm and continues the local search with only one sample from each cluster. Hanagandi and Nikolaou applied the principles of the LC algorithm on GA [15]. Here also the best individuals of each cluster survive, but the rest is created by recombination and mutation of the surviving elite. The Clearing Procedure by Petrowski uses the same approach [16].
Beasley et. al. proposed a Multi-Start GA, the Sequential Niching algorithm, for the multimodal optimization problem [17]. The optimization result of each GA run is stored as identified optimum and a penalty function is added permanently to the fitness function to prevent the next run of the GA from finding the very same optimum again. The shape and the radii of the penalty function are crucial for this algorithm since it can add several artificial deceiving local optima to the search space. However, the Sequential Niching yields the advantage that the basic EA mechanisms remain unaltered.

3 Clustering Based Niching EA

To introduce the biological concept of species into NEA we combined two methods: First a multi-population strategy with localized selection and limited interbreeding between sub-populations ($\widehat{=}$species) and second the use of clustering algorithms to identify species in an undifferentiated population. Both methods have been successfully applied independently, but to our knowledge they have not been merged into a single NEA approach so far.
The multi-population strategy has the advantage that it can preserve diversity through localized competition, for example the selection from localized clusters as performed by Hanagandi and Nikolaou, and localized interbreeding, compare mating restriction [4, Sec. 6.2.4]. However, each sub-population still behaves like a standard EA optimizer. This allows us to apply the full range of possible EA extensions or specialized EA operators on the level of sub-populations, like CMA [18] or MVA [19] mutation or even other NEA techniques like Fitness Sharing within each sub-population.
The goal of clustering algorithms is to group data units into cluster in such a way that units within a cluster are most similar while the clusters are relatively distinct from each other. In our application the EA individuals x_i are the data units and the resulting clusters will represent the species/niches. Using clustering algorithms to identify species allows us to move all relevant parameters for the niching behavior down to the clustering algorithm. There are numerous clustering algorithms available for multiple data types and applications with varying parameters requirements. We just need to choose the algorithm that yields the best ratio between clustering behavior and number of necessary parameters.

3.1 The Multi-population Strategy

We implemented a multi-population strategy that starts with a randomized single primordial undifferentiated population D_0 and allows differentiated sub-populations(species) $D_{i\geq 1}$ to be generated dynamically. D_0 plays the special role of containing all individuals that do not belong to any identified species. While the undifferentiated population D_0 explores the general search space, species $D_{i\geq 1}$ exploit already identified niches.
After initialization of D_0 the CBN-EA generational cycle is entered until an EA termination criterion is met, like maximum number of generations reached or numbers of fitness evaluations, see Fig. 1.

```
D0 = createInitialPop();
repeat{
  // species evolution phase
   for all i do simulateEAGeneration(Di);
  // species differentiation phase
   if (numOfClusters(D0) ≥ 1) split(D0);
  for all (i>0) do {
    if (numOfClusters(Di) > 1) split(Di);
    D0.add(Di.getLoners());
  }
  // species convergence phase
  TLP = createEmptyPop();
  for all (i>0) do TLP.addCentroidOf(Di);
  if (numOfCluster(TLP) ≥ 1) mergeDemes();
} until (maxGenerationReached());
```

Fig. 1. Pseudocode of CBN

First the **species evolution phase** is performed by simulating evaluation, selection and reproduction independently for each population D_i.
Then the **species differentiation phase** is entered:D_0 is tested whether differentiation occurs (`numOf Clusters(`D_0`) ≥ 1`). If clusters can be identified by the clustering algorithm new sub-populations are created from the members of the clusters. Then the clustering algorithm is called on all sub-populations $D_{i\geq 1}$ to test if a species continues to differentiate further into new species which are to be separated. On the other hand, if the clustering algorithm finds individuals in $D_{i\geq 1}$ that do not belong to any species (``D_i`.getLoners()`), those individuals are moved to D_0 as straying loners.
Finally, the **species convergence phase** is performed: All species $D_{i\geq 1}$ add a representative (e.g. a centroid) to a temporary population of representatives (`TLP`). Clusters found in `TLP` indicate species that converge to the same niche which are to be merged to join their effort.
Depending on the convergence speed of the EA method, multiple EA generational steps can be performed before applying the species differentiation and convergence mechanisms. We found that early differentiation enables CBN to identify more optima, as the standard EA convergence behavior would progressively remove individuals located on unincisive local optima.
Merging of species that converge on the same niche will have the effect that niches with a bigger basin of attraction will be populated by species with more members. But this mechanism also guarantees, that if the CBN is converged there exists at most one species per niche. This enables the CBN to identify global/local optima directly, by returning just the best individual of each species. It is necessary to note, that currently there is no crossover between species and that there is no competition between species. We want to introduce these mechanisms to the CBN in the near future.

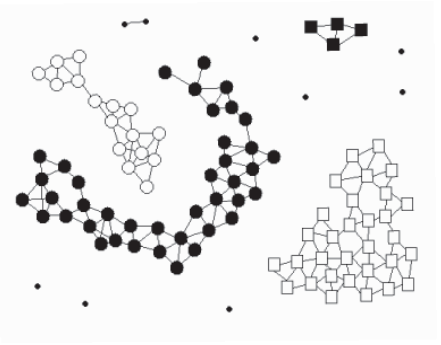


Fig. 2. Density-Based Cluster-Analysis

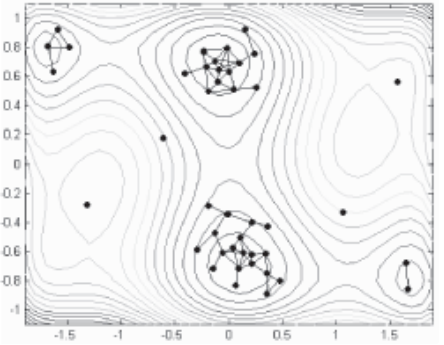


Fig. 3. Clustered population on M2

3.2 The Clustering Algorithm

Ester, Kriegel and Xiaowei to identify species [20,21]. This clustering algorithm identifies clusters by connecting individuals if the distance $\|x_i, x_j\|$ between them is lower than a given threshold value σ_{dist}. All interconnected groups of individuals, whose group size exceeds a minimum value *MinPts* are identified as cluster. Fig. 2 gives an example what kind of clusters can be identified by the 'density-based' clustering algorithm using $MinPts = 4$.
The 'density-based' clustering algorithm offers several advantages:

- it allows clusters of varying size and shape,
- it can identify clusters of *a priori* unknown number,
- the algorithm allows for loners which do not belong to any species, indicated as small dots in Fig. 2,
- it requires only two parameters that are easy to interpret.

Since *MinPts* gives the minimum size of a cluster it also gives the minimum size of a sub-population. And the minimum size of a sub-population can be chosen *a priori* not depending on the problem but on the EA method used. Small values for *MinPts* will be sufficient for mutation oriented EA methods like ES but larger values will be necessary for crossover based EA methods like GAs.
The parameter σ_{dist} is not as easy to select. It gives a lower bound, below which two optima can not be distinguished, because the clustering algorithm is not able to separate the clusters.
Fig. 3 gives an example for a clustered ES population with $MinPts = 2$ using an euclidean distance metric. Note that although an individual is located on the local optimum in the lower right corner, no species can be established. To increase the stability of identified species the mutation step size must be of the same order of magnitude as σ_{dist}. Otherwise individuals leave and enter species randomly simply because of mutation. Therefore, we currently limit the ES mutation step size to σ_{dist}, if an individual belongs to a species. This constraint can be removed, if the individual returns to D_0 as a loner.

4 Results

We compared the CBN on five multimodal benchmark functions, given in the Appendix, to Fitness Sharing (FS) and a Multi-Start Hill-Climber (MS-HS). Since we used real-valued benchmark functions we decided to apply Evolution Strategies (ES). We used a $(60 + 120)$-ES with best selection scheme simulated for $T = 100$ generations. For the CBN we used $MinPts = 2$ as minimum cluster size. To increase the performance of the FS-ES and to avoid the problem of finding a suitable scaling function, we used a hill-climbing post-processing step start at $T = 70$ but only for the FS-ES. The MS-HC was simulated by 120 independent (1+1)+ES trials with a fixed mutation rate σ_{mut}.
The performance of the algorithms is measured by the number of optima each algorithm found, averaged over 25 runs. An optimum o_j was considered 'found' if "$\exists\ x_i \in P_{t=T}\ ,\ \|x_i, o_j\| \leq \epsilon = 0.005$," where $P_{t=T}$ is the complete population at the end of each run and x_i an individual in $P_{t=T}$.
We applied all three algorithms on a normalized search space and varied a 'resolution' parameter to find the best parameter settings for each algorithm. In case of the MS-HC 'resolution' represents the fixed mutation rate σ_{mut}, for the FS-ES the parameter gives the critical value of σ_{share} and for the CBN the 'resolution' gives the σ_{dist} of the 'density-based' clustering algorithm.

4.1 Two Dimensional Benchmark Functions

On all four benchmark problems the MS-HC achieved the best results due to the limited search space and the high number of multi-starts and the available number of fitness calls. Only in case of M2 and M3 the MS-HS failed to identify all local optima if σ_{mut} becomes too big, since the high mutation rates enable a single HC to escape local optima, see Fig. 6 and Fig. 7.

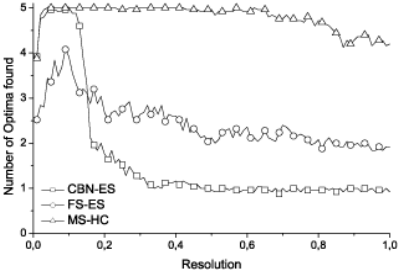


Fig. 4. M0, avg. number of optima found

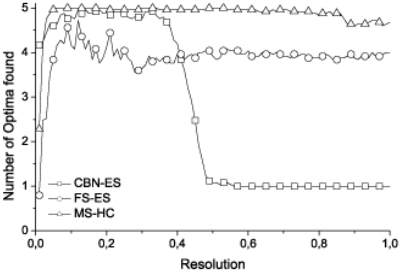


Fig. 5. M1, avg. number of optima found

With the additional HC post-processing step the FS-ES became less prone to bad values for σ_{share}, but performed not as well as the MS-HC. But although extremely high values for σ_{share} rendered the FS ineffective, they caused an evenly distributed start for the HC post-processing step, compare Fig. 7.

The CBN-ES performed better than FS-ES on M0 and M1. But the importance of σ_{dist} becomes evident. As discussed before, σ_{dist} gives the minimal distance between two separable clusters, see 3.2. In case of the M0 benchmark function the optima are about 1/5 units apart, compare Fig. 10, and about 1/2 units in case of M1, compare Fig. 11. These are exactly the σ_{dist} values where the performance of the CBN-ES drops to the behavior of a standard ES without niching, thus converging to a single optima. The same effect can be observed for M2 and M3, compare Fig. 12 and Fig. 13. But here the optima are not as evenly spaced and the CBN-ES fails step-by-step as σ_{dist} is increasing.

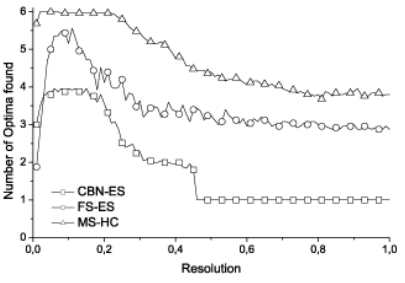


Fig. 6. M2, avg. number of optima found

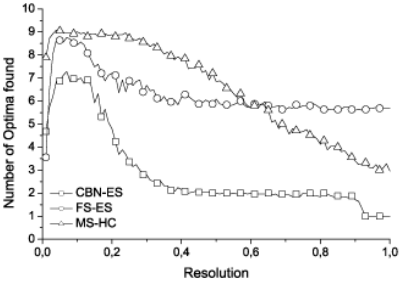


Fig. 7. M3, avg. number of optima found

On M2 and M3 CBN performed not as well as the FS-ES with the additional post-processing step. Regarding M2 CBN failed to identify the two most unincisive local optima, compare Fig. 3. This is caused by the greedy best selection strategy used. Most individuals of D_0 converged to the more attractive optima before species can be identified. On both problems the exploring character of D_0 seemed to fail.

4.2 The *n*-Dimensional Benchmark Function

To examine how the algorithms react to increasing dimensionality we compared them on the n-dimensional M5 benchmark function with 5^n optima, see Fig. 8 and 9. Although the MS-HC has the advantage that it could track one optima per deployed individual, it finds considerably less optima. Also the performance of the HS-HC suffers if the problem dimension is increased.
The FS-ES on the other hand fails completely with the increasing problem dimension, although the evenly spaced optima with equal fitness should provide perfect conditions for Fitness Sharing.
Regarding the CBN the number of optima identified with $\mu = 60$ never exceeds fifteen even for $n < 4$. This suggests that CBN is limited by the population size rather than the problem dimension. Although $MinPts = 2$ allows a minimum species size of two, merging of species causes much bigger sub-populations.

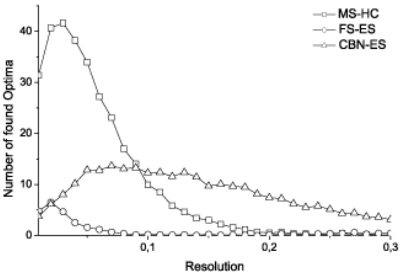


Fig. 8. M5 $n = 4$, avg. number of optima

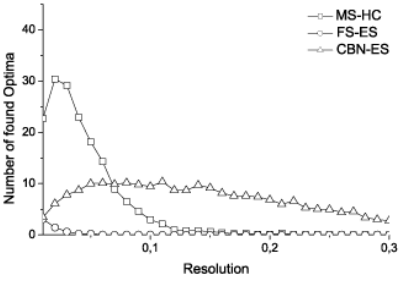


Fig. 9. M5 $n = 5$, avg. number of optima

These results indicate, that CBN scales better with increasing search space dimension than MS-HC and FS-ES but that CBN needs more individuals to retain a niche than *MinPts* would suggest.

5 Conclusions and Future Research

The CBN-EA we proposed, is a new niching EA method based on the formation of species. CBN joins a multi-population strategy with clustering analysis for species detection. This approach is virtually independent of the EA method it is applied to, because it does not put any restriction on the EA method used. Furthermore, CBN does not alter the search space and therefore does not disrupt the normal convergence behavior of the EA. Additionally CBN is able to actually identify optima using the concept of species without further post-processing.
Since we used benchmark functions that do not benefit from EA crossover as an exploratory operator, except for the M4 function, it is no surprise that the simple Multi-Start-(1+1)ES strategy performs best on all benchmark functions. And although Fitness Sharing was allowed to make extensive use of the HC post-processing step, it never squares the performance of the Multi-Start-(1+1)ES. CBN on the other hand is actually able to equal the Multi-Start-(1+1)ES on M0 and M1, if suitable values for σ_{dist} are used. The failure of CBN on M2 and M3 indicates, that currently the exploratory elements of CBN are too weak.
Regarding the n-dimensional M4 benchmark function Fitness Sharing fails even in spite of the available exploratory crossover. Most likely because of the increasing problem dimension. CBN on the other hand scales rather well with the increase of problem dimension, but it is limited by the fixed population size and the fact that currently no exploratory interbreeding between species is implemented.
Therefore our future work will concentrate on introducing exploratory interbreeding between species to the CBN-EA, to take the full advantage of the positive properties of niching EAs. And we will focus on problems where exploratory crossover yields a greater benefit, otherwise Multi-Start Local-Search strategies will most likely prevail over an niching EA.

To free resources for interspecies crossover within the CBN-EA, we will introduce mechanisms that use redundant individuals from extremely large species or already converged species for reinitialization. If the reinitialized individuals are descendants from parents from different competing species, we are also able to introduce a additional selection pressure on species.

Acknowledgements. This research has been funded by the German Federal Ministry of Research and Education (BMBF) under contract No. 03C0309E.

References

1. Goldberg, D., Richardson, J.: Genetic algorithms with sharing for multimodal function optimization. In Grefenstette, ed.: Proceedings of the 2nd International Conference on Genetic Algorithms. (1987) 41–49
2. De Jong, K.: An analysis of the behavior of a class of genetic algorithms. (Doctoral dissertation, University of Michigan) Dissertation Abstracts International, 36(10), 5140B. (University Microfilms No. 76-9381) (1975)
3. Mahfoud, S.W.: Crowding and preselection revisited. In Männer, R., Manderick, B., eds.: Parallel Problem Solving from Nature II, Amsterdam, North-Holland (1992) 27–36
4. Baeck, T., Fogel, D., Michalewicz, Z., eds.: Handbook of Evolutionary Computation. Oxford University Press, New York, and Institute of Physics Publishing, Bristol (1997)
5. Ursem, R.K.: Multinational evolutionary algorithms. In Angeline, P.J., Michalewicz, Z., Schoenauer, M., Yao, X., Zalzala, A., eds.: Proceedings of the Congress on Evolutionary Computation. Volume 3., Mayflower Hotel, Washington D.C., USA, IEEE Press (1999) 1633–1640
6. Ursem, R.K.: Multinational GAs: Multimodal optimization techniques in dynamic environments. In Whitley, D., Goldberg, D.E., Cantu-Paz, E., Spector, L., Parmee, I., Beyer, H.G., eds.: Proceedings of the Genetic and Evolutionary Computation Conference, Las Vegas, Nevada, USA, Morgan Kaufmann (2000) 19–26
7. Tsutsui, S., Fujimoto, Y.: Forking genetic algorithm with blocking and shrinking modes (FGA). In Forrest, S., ed.: Proceedings of the 5th International Conference on Genetic Algorithms, San Mateo, CA, Morgan Kaufman (1993) 206–215
8. Mahfoud, S.W.: Genetic drift in sharing methods. In: Proceedings of the 1st IEEE Conference on Evolutionary Computation. Volume 1., Piscataway, NJ, IEEE Service Center (1994) 67–72
9. Mahfoud, S.W.: Population sizing for sharing methods. Technical Report IlliGAL Report No 94005* (1994)
10. Darwen, P., Yao, X.: How good is fitness sharing with a scaling function. Technical Report CS 8/95, University College, The University of New South Wales, Canberra (1995)
11. Mahfoud, S.W.: A comparison of parallel and sequential niching methods. In Eshelman, L., ed.: Proceedings of the Sixth International Conference on Genetic Algorithms, San Francisco, CA, Morgan Kaufmann (1995) 136–143
12. Yin, X., Germany, N.: A fast genetic algorithm with sharing using cluster analysis methods in multimodal function optimization. In: Proceedings of the International Conference on Artificial Neural Nets and Genetic Algorithms, Innsbruck, Austria (1993) 450–457

13. Gan, J., Warwick, K.: A genetic algorithm with dynamic niche clustering for multimodal function optimisation. (Internal Report No. 98-001) Cybernetics Dept, Reading University (1998)
14. Törn, A.: Cluster analysis using seed points and density-determined hyperspheres as an aid to global optimization. IEEE Transactions on Systems, Man and Cybernetics **7** (1977) 610–616
15. Hanagandi, V., Nikolaou, M.: A hybrid approach to global optimization using a clustering algorithm in a genetic search framework. Computers and Chemical Engineering **22** (1998) 1913–1925
16. Pétrowski, A.: A clearing procedure as a niching method for genetic algorithms. In: ICEC96, IEEE Press (1996) 798–803
17. Beasley, D., Bull, D., Martin, R.: A sequential niche technique for multimodal function optimization. Evolutionary Computation **1** (1993) 101–125
18. Hansen, N., Ostermeier, A.: Adapting arbitrary normal mutation distributions in evolution strategies: The covariance matrix adaption. In: Proceedings of the 1996 IEEE International Conference on Evolutionary Computation. (1996) 312–317
19. Poland, J., Zell, A.: Main vector adaptation: A cma variant with linear time and space complexity. In: Proceedings of the Genetic and Evolutionary Computation Conference, Morgan Kaufman (2001) 1050–1055
20. Ester, M., Kriegel, H.P., Sander, J., Xiaowei, X.: A density-based algorithm for discovering clusters in large spatial databases with noise. In Simoudis, E., Han, J., Fayyad, U., eds.: 2nd International Conference on Knowledge Discovery and Data Mining, Portland, Oregon, AAAI Press (1996) 226–231
21. Sander, J., Ester, M., Kriegel, H.P., Xiaowei, X.: Density-based clustering in spatial databases, the algorithm gdbscan and its applications. Data Mining and Knowledge Discovery **2** (1998) 169–194

Appendix: Benchmark Functions

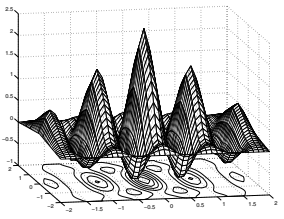


Fig. 10. Benchmark function M0

M0: Five hills and four valleys (5 peaks), as suggested in [5]:

$$\begin{aligned} M0(x,y) = \sin(2.2\pi x + 0.5\pi) \\ \cdot \frac{2-\text{abs}(y)}{2} \cdot \frac{3-\text{abs}(x)}{2} \\ + \sin(0.5\pi y^2 + 0.5\pi) \\ \cdot \frac{2-\text{abs}(y)}{2} \cdot \frac{3-\text{abs}(x)}{2} \end{aligned} \tag{5}$$

where $-2 \leq x, y \leq 2$

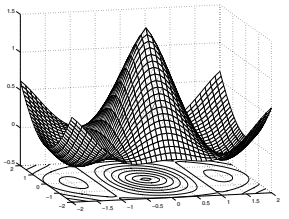


Fig. 11. Benchmark function M1

M1: One center peak and four neighbors (5 peaks), as suggested in [5]:

$$M1(x,y) = 3\sin((0.5x\pi + 0.5\pi) \cdot \frac{2 - \sqrt{x^2+y^2}}{4}) \quad (6)$$

where $-2 \leq x, y \leq 2$

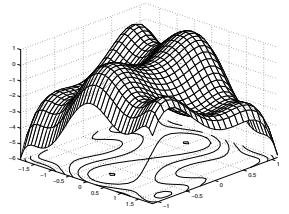


Fig. 12. Benchmark function M2

M2: Six hump camel back (6 peaks), as suggested in [5]:

$$M2(x,y) = -((4 - 2.1x^2 + \frac{x^4}{3})x^2 + xy + (-4 + 4y^2)y^2) \quad (7)$$

where $-1.9 \leq x \leq 1.9$ and $-1.1 \leq y \leq 1.1$

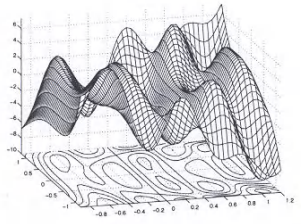


Fig. 13. Benchmark function M3

M3: Waves (10 peaks), as suggested in [5]:

$$M3(x,y) = -(y^2 - 4.5y^2)xy - 4.7\cos(3x - y^2(2+x)) \cdot \sin(2.5\pi x) + (0.3x)^2 \quad (8)$$

where $-0.9 \leq x \leq 1.2$ and $-1.2 \leq y \leq 1.2$

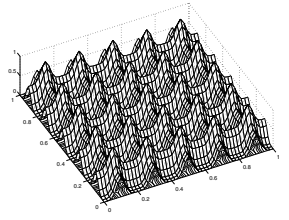


Fig. 14. Benchmark function M5

M5: n-dimensional sine (5^n peaks):

$$M5(\bar{x}) = 1 - \frac{1}{n}\sum_{i=1}^{n}(1 - \sin^6(5\pi x_i)) \quad (9)$$

where $0 \leq x_i \leq 1$, $n = 2$

Evolving a Cooperative Transport Behavior for Two Simple Robots

Roderich Groß and Marco Dorigo

IRIDIA, Université Libre de Bruxelles
Avenue Franklin D. Roosevelt 50
CP 194/6, 1050 Bruxelles, Belgium
{rgross,mdorigo}@ulb.ac.be

Abstract. This paper addresses the problem of cooperative transport of an object by a group of two simple autonomous mobile robots called *s-bots*. S-bots are able to establish physical connections between each other and with an object called the prey. The environment consists of a flat ground, the prey, and a light-emitting beacon. The task is to move the prey as far as possible in an arbitrary direction by pulling and/or pushing it. The object cannot be moved without coordination. There is no explicit communication among s-bots; moreover, the s-bots cannot sense each other. All experiments are carried out using a 3D physics simulator.
The s-bots are controlled by recurrent neural networks that are created by an evolutionary algorithm. Evolved solutions attained a satisfactory level of performance and some of them exhibit remarkably low fluctuations under different conditions. Many solutions found can be applied to larger group sizes, making it possible to move bigger objects.

1 Introduction

The field of distributed robotics has received growing attention by researchers within the last 15 years. Multi-robot systems have been studied in various topic areas and in different application domains (Parker, 2000). Several works considered the cooperative transport of objects by a group of mobile robots. Some of these have been inspired by social insects such as ants.

Almost half a century ago, Sudd (1960) studied the transport of prey by single ants and by groups of ants of the species *Pheidole crassinada*. Although he observed that single ants would mostly behave similar to those engaged in group transport, he reported that group transport "showed co-operative features" (Sudd, 1960).

Deneubourg et al. (1990) proposed the use of self-organized approaches for the collection and transport of objects by robots in unpredictable environments. Each robot unit could be simple and inefficient by itself, but a group of robots would have a complex and efficient behavior. Cooperation could be achieved without any direct communication among robots, as in some biological systems that rely on interactions via the environment or that exhibit particular individual

P. Liardet et al. (Eds.): EA 2003, LNCS 2936, pp. 305–316, 2004.

behaviors (Grassé, 1959; Deneubourg and Goss, 1989). In a transport task, for instance, coordination could occur by inter-individual competition and/or via the object to be moved.

In a remarkable series of works, Kube and Zhang (1993a, 1993b) and Kube et al. (1993) studied a decentralized approach to let a group of simple robots push an object that was too heavy to be moved by a single robot. The approach did not make use of explicit communication among the robots. Taking inspiration from ant colonies, they extended the model adding a stagnation recovery mechanism (Kube and Zhang, 1995). Later, Kube and Bonabeau (2000) provided a first formalized model of cooperative transport in ants. They used a 2D simulator and a real robotic system for validation. Kube and Zhang, as well as Sudd, reported a relative lack of efficiency of the behaviors they observed.

The aim of the experiment described in the following of this paper is to study to what extent a group of two simple s-bots equipped with very limited cognitive and computational abilities is able to exhibit coordination of individual activities and to exploit environmental signals in order to transport an object as far as possible within a fixed time period. To do so, we synthesize individual control policies via an evolutionary algorithm with the objective of obtaining highly efficient group behaviors.

2 Experimental Framework

2.1 The Environment and the s-Bot

The experiment is carried out using a 3D physics simulator.[1] The simulated environment consists of a flat ground of infinite size, a light-emitting beacon, and a prey. The prey is modeled as a cylinder of mass 500 g, 10 cm in height, and 12 cm in radius. The friction coefficient between the prey and the ground is set to 0.25. To calculate frictional forces, an approximation based on the Coulomb friction law is used. Gravity is set to $981\,\mathrm{cm\,s^{-2}}$.

The s-bot model is shown in Figure 1. It is an approximation of a real s-bot, currently under construction within the "Swarm-bots" project (Mondada et al., 2002; Dorigo et al., 2003; Trianni et al., 2003; see also `www.swarm-bots.org`). The body is composed of a cylindrical torso, a *gripper* element that is fixed on the torso's edge heading forward, a long cylindrical pillar placed on the top of the torso to support a visual camera system, and four wheels: two active wheels on the left and right, and two passive wheels, one in the front, and one in the back. The s-bot has a height of 16.85 cm and a mass of 660 g.

The control system reads the sensors status and sets actuator activations every 0.1 seconds. In the following, the actuators and sensors of the s-bots are detailed.

[1] The core simulator system has been developed in cooperation with IDSIA (Dalle Molle Institute of Artificial Intelligence Studies), Switzerland. It is based on libraries of the commercial physics engine Vortex^{TM} from CMLabs.

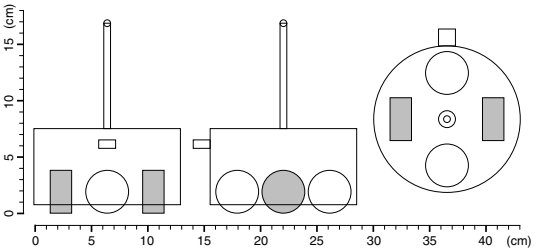


Fig. 1. Front, side and top view of the s-bot: the cylindrical torso is equipped with a small gripper element heading forward, and a camera placed on the pillar. Two passive, spherical wheels (white) are connected to the torso's back and front part. Two motorized, cylindrical wheels (gray) are connected to the torso on the left and right side.

Actuators. The s-bot is equipped with two active wheels and a gripper element. The active wheels are connected to the torso using a motorized hinge joint. The torque that can act to accelerate an active wheel is limited to 200,000 dyne cm.[2] An active wheel can be controlled by setting a desired angular speed $v \in [-15\,\text{rad/s}, 15\,\text{rad/s}]$. To model real actuators' noise in the simulation, some wheels are biased to turn faster than others, the wheel's speed is noisy, and activations less than certain random thresholds will not result in any movement of the wheel (for details see Groß, 2003).

The gripper element is a controllable, sticky box heading forward. If the element is in *gripping* mode, a connection will be established as soon as the gripper element is in contact with a grippable object. This is realized by dynamically creating a ball-and-socket joint connecting the s-bot's torso and the object. The joint is positioned on the gripper element. Once the gripper is set to the *open* mode, this joint will be removed to release the object.

The connection formed by the gripper element will break if a too strong force is transmitted via the corresponding joint. This is a characteristic of the real s-bot and of gripper elements in general. The limit for the force acting on the gripper element is 1,000,000 dyne (i.e., 10 Newton). This limit still permits s-bots engaged in the transport of a heavy prey to form structures such as small pulling chains.

Sensors. The s-bot is equipped with an omnidirectional camera and a gripper status sensor. These sensors have their counterparts on the real s-bot. A summary of the information provided by each sensor is presented in Table 1.

The camera is mounted on a pillar support that is fixed at the center of the torso's top. The camera sensor is situated 16.85 cm above the ground. In principle, the camera of the real s-bot is able to sense objects in all horizontal directions having an infinite range. However, the quality of the perceived signal

[2] 100,000 dyne correspond to 1 Newton.

decreases when the object's distance is increased. Therefore, the sensing range is restricted depending on the type of feature that is supposed to be extracted.

The simulated camera provides data based not on recorded images but on information available in the simulator, e.g., the distance to another object. The implicit assumption here is that in case of the real s-bot, such information can be obtained from the camera using feature extraction algorithms.

The camera sensor can provide information about a cylindrical prey in the surrounding, if the horizontal distance between the cylinder's border and the camera is at most 50 cm. In particular, the camera sensor is able to detect the horizontal distance between the s-bot and the closest point of the prey's border, and the horizontal angle to the corresponding point with respect to the s-bot's direction of forward motion.

Moreover, the camera sensor is able to detect the horizontal direction of the highest intensity of light perceived. Light is emitted by a single beacon that is placed in the environment.

Noise affecting the sensors is modeled in simulation in various ways. In case of the light or the prey perceived, an angular deviation r_α is added to the horizontal angle α_{rad} (radian measure) that indicates the target direction to the light or the prey. r_α is generated using the normal distribution $N(0, 0.01)$. Also the measured distance (in cm) to a perceived prey is affected by noise. This is modeled by adding a random variable following the normal distribution $N(0, 1)$.

Table 1. List of information provided by the sensors of an s-bot.

Sensor device	Information provided
gripper status sensor	status of being connected to another object through the gripper element
camera sensor	horizontal angle and distance to a cylindrical prey (horizontal sensing range: 50 cm)
	horizontal angle of highest intensity of light perceived (a constant light signal is emitted by a beacon)

2.2 The Neural Network Controller

The group of s-bots is controlled by a simple recurrent neural network that is synthesized by an evolutionary algorithm. All the s-bots of a group transporting a prey are initially equipped with an identical neural network. Due to recurrent connections, these networks have memory and they are not restricted to purely reactive behaviors. The ability to compare current and previous states of the environment perceived might be beneficial to recognize the presence and the actions of teammates.

We used an Elman network (Elman, 1990) with five input neurons, five (fully inter-connected) hidden neurons and three output neurons. The activations of

the neurons of the input layer correspond to the current s-bot's sensor reading values. The activations of the output neurons are used to control the motorized wheels and the gripper element.

2.3 The Evolutionary Algorithm

The evolutionary algorithm used is a self-adaptive version of a $(\mu+\lambda)$ evolutionary strategy (Rechenberg, 1965; Schwefel, 1975). An individual is composed of real-valued object parameters $X = (x_1, x_2, \ldots, x_N)$ specifying the weights of the Elman Network used to control the s-bots, and real-valued strategy parameters $S = (s_1, s_2, \ldots, s_N)$ specifying the mutation strength used for each component of the object parameter vector.

Before the evolution is started, a *random walk* is performed in order to generate a wide variety of somewhat acceptable solutions to start from. The total number of fitness evaluations during the random walk is equal to the number of evaluations of an evolution lasting ten generations.

In each generation all individuals are assigned a fitness value. The best μ individuals are selected to create λ offspring. With a probability of 0.8 an offspring is created by mutation, otherwise two parent individuals are selected and recombined. In the latter case, the mutation operator is applied to the created offspring. The λ offspring and the μ parent individuals are copied into the population of the next generation. Note that the parent individuals that are replicated from the previous generation get re-evaluated. In fact, the fitness values are affected by various random components in the fitness estimation procedure. Re-evaluating the parents' fitness values inhibits a potential systematic over-estimation in time due to previous fitness estimations.

The object parameter x_i is mutated by adding a random variable from the normal distribution $N(0, s_i^2)$. Beforehand, the mutation strength parameter s_i is multiplied by a random variable ξ_i that follows a *lognormal* distribution similar to the one proposed by Schwefel (1974). As recombination operator we use intermediate and dominant recombination (Beyer, 2001), both with the same probability.

The number of offspring is $\lambda = 80$ and the number of parents is $\mu = 20$. For a detailed description of the evolutionary strategy used see Groß (2003).

2.4 The Fitness

The task is to control a group of two simple s-bots so that they cooperate to transport as far as possible a heavy prey within a fixed time period. The direction of movement of the prey is not predetermined. Since the prey cannot be moved by a single s-bot, coordination among s-bots is necessary.

Each individual, that is, a common controller for a group of s-bots, is evaluated by performing S tests $t_1, t_2 \ldots, t_S$ against a sample composed of S configurations specifying the s-bots' initial placements and the position of the beacon. The sampling size is set in our experiments to $S = 5$. The sample is changed

once per generation, so that all the individuals that compete with each other are evaluated under similar conditions. This should increase the comparability of fitness values among individuals within the same generation. Note that the μ parent individuals that are copied into the next generation by default get re-evaluated based on the new sample.

In each test, the simulation lasts 20 simulated seconds. The prey is placed in the center of the environment. The s-bots are placed at random positions and orientations, but not more than 25 cm away from the prey. This ensures that the prey can initially be detected by each s-bot. The beacon is placed at a random position 300 cm away from the center of the environment. This is less than the distance the prey can be moved within the simulation time of 20 seconds.

For each test $t_i, i \in \{1, \ldots, S\}$, a quality measure q_i (i.e., the transport quality) evaluates the exhibited transport performance. The final fitness score f is computed using a weighted average. Let ϕ be a permutation of $\{1, 2, \ldots, S\}$ such that $q_{\phi(1)} \leq q_{\phi(2)} \leq \ldots \leq q_{\phi(S)}$, then the fitness value is given by

$$f = \frac{2}{S(S+1)} \sum_{i=1}^{S} (S - i + 1)\, q_{\phi(i)}. \tag{1}$$

In this way, tests resulting in weaker transport quality q_i contribute more than the others to the fitness value so that fluctuations get punished. The intended effect is that individuals that emerge during evolution get more resistant over time to the randomness of the robots' initial placements as well as to the noise in the robots' sensors and actuators.

To favor the evolution of solutions that let all s-bots participate in the transport activities, the transport quality takes into account also the clustering performance of s-bots around the prey. The clustering quality c_i in a test t_i is defined as

$$c_i = \frac{1}{n} \sum_{j=1}^{n} c_i^{(j)}, \text{ and } c_i^{(j)} = \begin{cases} 0 & \text{if } d_i^{(j)} > 50; \\ 1 & \text{if } d_i^{(j)} < 25; \\ \frac{50 - d_i^{(j)}}{25} & \text{otherwise;} \end{cases} \tag{2}$$

where $d_i^{(j)}$ denotes the Euclidean distance (in cm) observed in test t_i between the final positions of the jth s-bot and the center of the prey minus the radius of the prey.

Therefore, s-bots that are not more than 25 cm away from the perimeter of the prey get the maximum clustering performance value of 1. In this way, any structure of two collaborating s-bots pushing or pulling the prey cannot be punished. S-bots that are more than 50 cm away cannot sense the prey any more and get the lowest possible clustering performance value of 0.

The transport quality q_i in a test t_i is defined as

$$q_i = \begin{cases} c_i & \text{if } D_i = 0; \\ 1 + (R + \sqrt{D_i}) {c_i}^{\rho} & \text{otherwise;} \end{cases} \tag{3}$$

where D_i denotes the Euclidean distance (in cm) observed in test t_i between the prey's initial and final positions, $R = 1$ is a constant reward and $\rho = 5$ is a constant exponent in order to punish solutions exhibiting weak clustering performance. The root function is applied as scaling function.

3 Results

The experimental setup described above has been used in ten independent evolutionary runs of 150 generations each. One run lasted a little bit less than a week on a machine equipped with an AMD Athlon XP^{TM} 1800+ processor. In the following, the transport performance of the evolved controllers and their ability to scale when using larger group sizes are discussed.

Quality of transport. The transport quality of a group (as defined by Equation 3) in a test depends on the genotype specifying the neural network controller, the s-bots' initial positions and orientations, the position of the beacon in the environment, the particular offset and threshold values of each sensor and actuator, and the amount of noise that affects sensors and actuators at each time step. Of course, the ultimate goal is to generate genotypes that perform well under every possible condition. However, there is a very large number of possible conditions, and during evolution the solutions are evaluated using a sample of five different conditions per generation.

Figure 2 displays the development of the best and average fitnesses for certain sets of individuals over all ten evolutionary runs:

- The dark gray boxes correspond to the observed fitness values of the best rated individuals for every population. Due to the random component in the fitness evaluation procedure, individuals exhibit performance fluctuations. Thus, the best fitness values are likely to be over-estimated.
- The average fitness value of the μ parent individuals of each population is illustrated by the light gray boxes. To avoid bias in the evaluation of the parent individuals, they get re-evaluated in the subsequent generation. In this way, although the average fitness of the re-evaluated parent individuals may fluctuate due to the noise, there is no systematic over-estimation caused by previous fitness estimations. In the figure, the average fitness value of the parent individuals is computed based on the set of re-evaluated fitness values.
- The white boxes correspond to the average fitness values of the entire population.

Looking at Figure 2 one can see that, in all cases, the fitness values tend to increase. Since the fitness values are noisy and, additionally, they are computed using a weighted average, it is hard to estimate the attained quality level. Therefore, we measured the transport quality of several individuals of each evolutionary run on a sample of 500 different tests. For each evolutionary run, we selected the $\mu = 20$ best (parent) individuals of the final generation. This set

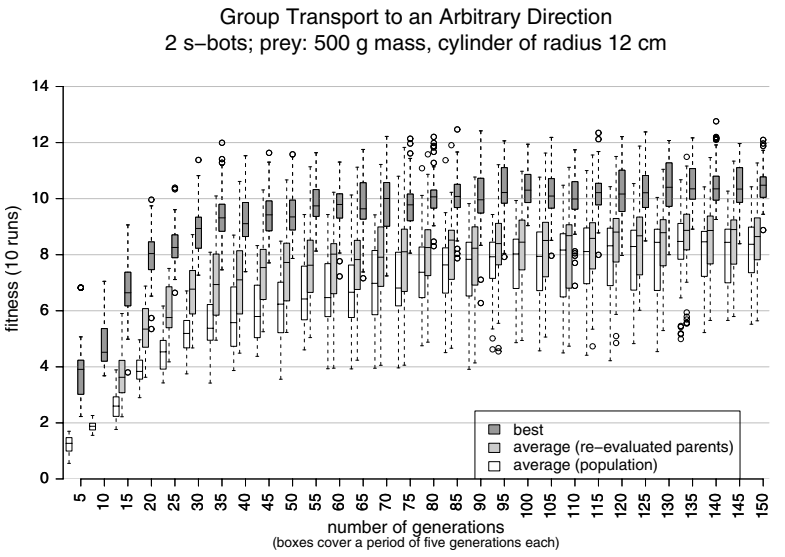


Fig. 2. Box-and-whisker plot visualizing certain characteristics of the evolutionary progress in ten runs. The box comprises observations ranging from the first to the third quartile. The median is indicated by a bar, dividing the box into the upper and lower part. The whiskers extend to the most extreme data point which is no more than 1.5 times the interquartile range. Outliers are indicated as circles. This plot illustrates the development of the best fitness of a population (dark-gray boxes), the average fitness of the set of $\mu = 20$ (re-evaluated) parent individuals (light-gray boxes) and the average fitness of the whole population (white boxes). Each box covers a period of five generations.

of μ parent individuals comprises all genetic material that would be exploited in subsequent generations in case the evolution would be continued. Performing the post-evaluation, it has been observed that parent individuals of the final generation of the same evolutionary run exhibit a quite similar performance. Therefore, we focus on two types of individuals from each evolutionary run: the one with the highest average performance concerning the 500 tests (type 1), and the one with the lowest observed standard deviation (type 2). For every evolutionary run, both types of individuals are post-evaluated for a second time, on a random, but fixed set of 1,000 tests.

Figure 3 shows a box-and-whisker plot presenting the transport quality values observed in these 1,000 tests for both types of individuals for all the evolutionary runs. The first five solutions displayed make an essential use of the gripper element.

According to Equation 3, the transport quality q_i observed in a test t_i is at least 1 if the prey has been moved, and $c_i \in [0, 1]$ otherwise. Looking at Figure 3, it can be seen that the prey has been moved in almost all cases, since almost all observed transport quality values are in the range 2 to 15. The distance D_i

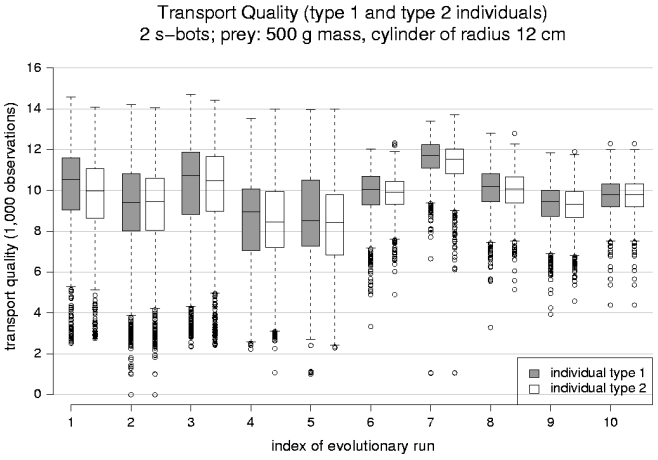


Fig. 3. Box-and-whisker plot visualizing the transport quality observed in 1,000 tests for individuals of type 1 and type 2 in the final generations. The type 1 individual of an evolutionary run is the one with the highest average of the transport quality values observed in the 500 tests during the first post-evaluation and the type 2 individual is the one with the lowest observed standard deviation.

the prey has been moved is at least $(q_i - 2)^2$, if $q_i \geq 2$.[3] If $c_i < 1$, D_i is bigger than $(q_i - 2)^2$. Assuming the worst case for the distance moved, that is, a perfect clustering quality $c_i = 1$, some examples for pairs of transport quality q_i and the corresponding moved distance D_i are given in the following table:

q_i	2	3	4	5	6	7	8	9	10	11	12	13	14
D_i (in cm)	0	1	4	9	16	25	36	49	64	81	100	121	144

Let us consider a setup in which a group of two s-bots, structured in a connected, linearly aligned chain, is pulling the prey for a period of 20 seconds. The first s-bot of the chain is connected to the prey right from the start. All s-bots are controlled by a handwritten controller: each s-bot moves backwards with maximum angular speed of the wheels; the gripper element remains closed. If a chain of two s-bots is placed in such a structural configuration, the distance the object can be moved within the simulation time of 20 seconds is around 149.64 cm. This corresponds to a transport quality of approximately 14.23.

However, during the fitness evaluation the s-bots are not placed in such an initial structure: the s-bots are scattered in the environment at random positions and with random orientation. Therefore, the s-bots have to approach the prey and to coordinate themselves before a structure is formed that can be used to

[3] Let $q_i \geq 2$: $q_i = 1 + (1 + \sqrt{D_i})c_i^5 \leq 1 + (1 + \sqrt{D_i}) = 2 + \sqrt{D_i} \Rightarrow q_i - 2 \leq \sqrt{D_i}$ $\Rightarrow D_i \geq (q_i - 2)^2$.

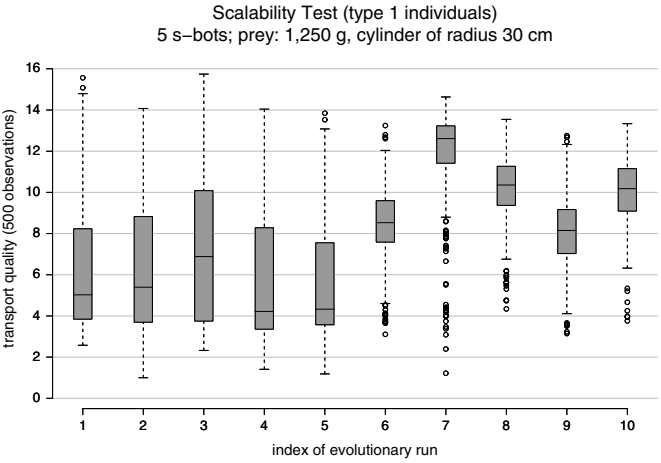


Fig. 4. For each evolutionary run, the individual for which the best average performance was observed in the post-evaluation described previously (the type 1 individual) is evaluated 500 times using a group of five s-bots and a prey of radius 30 cm and mass 1,250 g. The simulation period is extended by ten additional seconds, since it requires more time for each s-bot to move around the larger prey. The first five individuals make essential use of the gripper element.

transport the prey. If we assume that the randomly placed s-bots need ten seconds to form a chain like the one described, this structure could only pull the prey for the remaining ten seconds of simulation time. During this period, the prey can be moved 74.50 cm, corresponding to a transport quality of approximately 10.63.

The median values of transport quality exhibited by the better performing half of the type 1 individuals are in the range [10.03,11.73]. The observed standard deviations concerning the half of the type 2 individuals having lower fluctuations are in the range [0.91,1.30]. In case of the evolutionary runs 6 to 10, we have obtained a number of satisfactory controllers that exhibit low performance fluctuations (see Figure 3). Overall, the controlled groups of s-bots act quite robustly with respect to various kinds of noise concerning the sensors and the actuators and with respect to different initial placements in the environment.

Scalability. The question arises whether these observed behaviors are scalable, that is, whether evolved individuals are able to cooperate in the transport of a heavier prey when the group size becomes larger. To try to give an answer, for each evolutionary run we took the individual with the best average performance and evaluated it using a group of five s-bots, engaged in moving a heavier prey of mass 1,250 g and of the same shape and size as previously. In most cases, the observed performance was low, since, having five s-bots, the strategies seemed not to be able to handle the increased inter-individual competition among s-bots

for the limited space around the prey. However, if the perimeter of the prey is multiplied by the same factor as the number of s-bots is increased, all solutions are able to move the prey, and the ones not relying on the gripper exhibit a satisfactory performance (see Figure 4).

4 Conclusions

In this work, we addressed the problem of controlling a group of two simple, mobile, autonomous robots so that they cooperatively transport a heavy prey as far as possible into an arbitrary direction within a fixed period of time. The robots are able to establish physical connections with each other as well as with the prey. The prey cannot be moved without cooperative behavior. There is no explicit communication among the s-bots. Moreover, the s-bots are not able to directly sense each other. Communication is limited to *interactions via the environment*, that is, stigmergy (Grassé, 1959; Dorigo et al., 2000).

In all of the ten evolutionary runs carried out, controllers were generated which exhibited an acceptable performance under most of the tested 1,000 conditions. In five out of ten runs, the system generated very high quality controllers which reached high performance levels and, at the same time, presented rather low fluctuations in performance. Moreover, many solutions could be applied to larger groups so that prey of bigger size and mass could be moved.

In general, the controllers' performances are very sensitive to the size of the prey. We also observed that the performance of those solutions that make essential use of the gripper element is more fluctuating than that of the other ones. The reason for this behavior is however not clear yet. In the future, we want to encourage the evolution of strategies that let the robots organize into assembled structures in order to transport prey of various shapes and sizes.

Acknowledgments. The authors wish to thank all the members of the "Swarm-bots" project for their support, suggestions and comments. This work was supported by the "Swarm-bots" project, funded by the Future and Emerging Technologies programme (IST-FET) of the European Commission, under grant IST-2000-31010, and by the "Virtual Swarm-bots" project, funded by the Belgian FNRS, contract no. 9.4515.03. Marco Dorigo acknowledges support from the Belgian FNRS, of which he is a Senior Research Associate. 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.

References

Beyer, H.-G.: The Theory of Evolution Strategies. Springer-Verlag, Berlin, Germany (2001)

Deneubourg, J.-L., Goss, S.: Collective patterns and decision-making. Ethology, Ecology and Evolution **1** (1989) 295–311

Deneubourg, J.-L., Goss, S., Sandini, G., Ferrari, F., Dario, P.: Self-organizing collection and transport of objects in unpredictable environments. In: Proc. of Japan–USA Symposium on Flexible Automation, Kyoto, Japan, ISCIE (1990) 1093–1098

Dorigo, M., Bonabeau, E., Theraulaz, G.: Ant algorithms and stigmergy. Future Generation Computer Systems **16** (2000) 851–871

Dorigo, M., Trianni, V., Şahin, E., Labella, T.H., Groß, M.R., Baldassarre, G., Nolfi, S., Deneubourg, J.-L., Mondada, F., Floreano, D., Gambardella, L.M.: Evolving self-organizing behaviors for a *Swarm-bot*. Technical Report IRIDIA/2003-11, Université Libre de Bruxelles, Belgium (2003)

Elman, J.: Finding structure in time. Cognitive Science **14** (1990) 179–211

Grassé, P.: La reconstruction du nid et les coordinations inter-individuelles chez *Bellicositermes natalensis et Cubitermes sp.* La théorie de la stigmergie : essai d'interprétation du comportement des termites constructeurs. Insect Sociaux **6** (1959) 41–81

Groß, R.: Swarm-intelligent robotics in prey retrieval tasks. Technical Report IRIDIA/2003-27, Université Libre de Bruxelles, Belgium (2003) DEA thesis.

Kube, C.R., Bonabeau, E.: Cooperative transport by ants and robots. Robotics and Autonomous Systems **30** (2000) 85–101

Kube, C.R., Zhang, H.: Collective robotic intelligence. In: Second International Conference on Simulation of Adaptive Behaviour (SAB-1992), Cambridge, MA, MIT Press (1993a) 460–468

Kube, C.R., Zhang, H.: Collective robotics: from social insects to robots. Adaptive Behaviour **2** (1993b) 189–218

Kube, C.R., Zhang, H.: Stagnation recovery behaviours for collective robotics. In: Proceedings 1994 IEEE/RSJ/GI International Conference on Intelligent Robotics and Systems, Los Alamitos, CA, IEEE Computer Society Press (1995) 1883–1890

Kube, C.R., Zhang, H., Wang, X.: Controlling collective tasks with an ALN. In: 1993 IEEE/RSJ International Conference on Intelligent Robots and Systems, Los Alamitos, CA, IEEE Computer Society Press (1993) 289–293

Mondada, F., Pettinaro, G.C., Kwee, I., Guignard, A., Gambardella, L., Floreano, D., Nolfi, S., Deneubourg, J.-L., Dorigo, M.: SWARM-BOT: A swarm of autonomous mobile robots with self-assembling capabilities. In: Proceedings of the International Workshop on Self-organisation and Evolution of Social Behaviour, Monte Verità, Ascona, Switzerland, University of Zurich (2002) 307–312

Parker, L.E.: Current state of the art in distributed autonomous mobile robotics. In: Distributed Autonomous Robotic System. Volume 4, Tokyo, Japan, Springer-Verlag (2000) 3–12

Rechenberg, I.: Cybernetic Solution Path of an Experimental Problem. PhD thesis, Royal Aircraft Establishment, Hants Farnborough, UK (1965)

Schwefel, H.-P.: Adaptive Mechanismen in der biologischen Evolution und ihr Einfluß auf die Evolutionsgeschwindigkeit. Technical Report Re 215/3, Technische Universität Berlin, Germany (1974)

Schwefel, H.-P.: Evolutionsstrategie und numerische Optimierung. PhD thesis, Technische Universität Berlin, Germany (1975)

Sudd, J.: The transport of prey by an ant *Pheidole crassinoda*. Behaviour **16** (1960) 295–308

Trianni, V., Groß, R., Labella, T.H., Şahin, E., Dorigo, M.: Evolving aggregation behaviors in a swarm of robots. In: Advances in Artificial Life – Proceedings of the 7th European Conference on Artificial Life (ECAL). Volume 2801 of Lecture Notes in Artificial Intelligence, Springer-Verlag, Berlin, Germany (2003) 865–874

Co-evolution in Artificial Ecosystems: Competition and Cooperation Using Allellopathy

Claude Lattaud

Artificial Intelligence Laboratory of Paris V University
LIAP5 – CRIP5
Paris V University
45, rue des Saints Pères
75006 Paris
France
Claude.Lattaud@math-info.univ-paris5.fr

Abstract. This paper presents simulations of long-term co-evolution in simple artificial ecosystems composed of different plant species. Artificial plants and plant communities, evolving in a 3D environment, are based on a multi-agent model. This multi-agent approach allows to define communication processes existing at plant and organ levels. A model of a particular chemical interaction between plants is developed in this paper : the allellopathy. This intraspecific and interspecific phenomenon is used in nature both in competition and in cooperation in plant community development. This paper focuses particularly on the emergence of competition and cooperation during long time periods. Several results obtained in this paper are close to natural observations.

1 Introduction

Usually, an ecosystem is defined as a community of species that live together and interact often, along with their abiotic environment, i.e. non-living parts of air, water and soil, rocks and minerals, dead organic matter, climate, fire, etc. 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 focuses on co-evolution processes occurring in ecosystems, competition and cooperation between plants for resources.

Competition happens when individuals are negatively affected by contesting a limited resource, and more generally when individuals of the same populations, i.e. intraspecific competition, or different populations, i.e. interspecific competition, use a common resource that is in short supply. Competition between plants takes place at several levels : space, minerals, light, etc. One of the strategies developed by plants to compete and to disturb the growth of other plants is the secretion of chemical substances in their close environment. This is the allellopathy.

P. Liardet et al. (Eds.): EA 2003, LNCS 2936, pp. 319–330, 2004.

Although in the first century B.C., Pline the Ancient noticed in [2] that no plant grows under walnut covers, he didn't fully understand that phenomenon. Allellopathy was first introduced in a 1937 monograph of Molish, [3], and was largely studied three decades later by Elroy L. Rice, [4, 5, 6]. He recognized four stages of plant succession in Oklahoma : the pioneer weed stage, lasting 2-3 years, the annual grass stage, lasting 9-13 years, the perennial bunch-grass stage, lasting 30-50 years, and finally the climax prairie. He established that allellopathy is the major factor causing this succession. He then enlarged the earlier definition of allellopathy to the „direct or indirect effects, positive or negative, of a plant on another plant, using biochemical compounds released in the environment".

At the present time, the allellopathic effects of plants are better known, and because of their important stake from an agronomic and a forestry point of view, that field of research has been developing itself during the last years. One of the best illustrated cases of allellopathy in terrestrial ecosystems is the production of camphor by eucalyptus. It evaporates from the leaves and is subsequently deposited in dew on the nearby vegetation, preventing other plants from growing in the close eucalyptus environment. Since this paper focuses both on competition and cooperation effects of allellopathy, simple competition like that of eucalyptus will not be studied.

Experiments described in this article are based on the observations of B. Boullard on natural fir plantation. In [7], he says that „Natural fir plantation is typically a mixed formation where several species coexist : fir but also spruce and beech […] Then, in this forest, the succession of dominant species is a rotation : fir coming under spruce, spruce under beech and beech under fir. So, the forest behaves like ground juxtapositions with a botanical composition continuously evolving, like a merry-go-round slowly running […] It seems that there is no doubt that the explanations of observed facts are linked to the nature and the properties of released biochemical compounds".

Experiments presented in this paper use a simplified allellopathic model allowing interactions between plants to build a long-term cooperation of different species. In a 3D continuous virtual world, artificial plants evolve during long time periods in order to simulate the evolution of natural competition and cooperation between three plant species on several generations.

Individual plant model is based on a multi-agent system [8]. This method has been chosen because of its capacity to model communication processes between a group of simple entities, the agents. Multi-agent paradigm, useful for modeling large sets of agents, is also used to define the plant community, where several types of information are exchanged between individuals. Co-evolution between large sets of individuals, i.e. competitive and cooperative behaviors, is particularly studied in the multi-agent field of research.

The first part of this paper describes a state of the art of plant modeling as well as previous works in plant community simulations in Artificial Life. The next chapter defines the model used in this paper for individual plants, plant communities and allellopathic interactions. Then in the fourth part, two sets of simulations with their experimental conditions are described, and results are discussed. Conclusions about these experiments and future works are presented in the last section.

2 State of the Art

The first part of this section describes formal models developed in order to simulate the evolution of individual plants. The second part presents models and applications used for the simulation of plant communities.

2.1 Individual Plant Models

2.1.1 L-Systems

L-Systems, developed by Artistid Lindenmayer in [9], are one of the first mathematical models based on a formal grammar to describe plant development. [10] specifically exhibits the mathematical theory beyond the L-Systems. An L-System is defined by one or several production rules. These rewriting rules, consisting of two components, a predecessor and a successor, operate on the entire plant simultaneously. During the derivation step of a production rule, the predecessor is replaced by the successor in the plant.

The plant shown in the Fig. 1, [11], has been realized with two simple rules. The first one increases the internode length, the brown part, while the second specifies that the apex, the green part, creates a branching structure with three new apices and an internode.

Several types of L-Systems based on the original one, like Stochastic or Context-Sensitive L-Systems, have been developed after. [12] offers a complete survey of most of them.

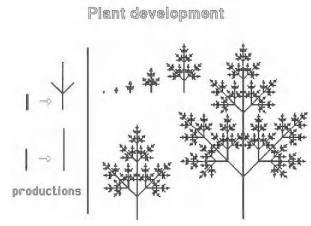


Fig. 1. Plant development using an L-System

2.1.2 Entity-Relation Graphs

Entity-relation graphs are particularly applied to tree modeling, and have been developed by the CIRAD for their AMAP program [13]. This method, [14], is based on the description of the bud operating cycle, i.e., growth, branching and death, by stochastic processes. Each part of a plant is described by an entity - internodes, growth units and an axe - then relations are built to link them - successor, kinship and scale.

Using this coding method, an entity-relation graph is obtained. Event sequences are applied by the use of Markov processes to model the plant growth.

Some of the goals of the AMAP project are to study general principles of plant organization, predict plant productions from a qualitative and a quantitative point of view, and build landscapes to help in town and rural planning.

2.2 Plant Community Models

Whole stand models are probably the oldest models of plant communities. Chang Hui Peng says in [15] that they first appeared in the end of the 19th Century. They describe the growth and the development process of a plant community using equations on plant properties, mean values such as the mean size or mean height of trees. These methods are useful for modeling homogeneous forests, but due to the difficulty in writing rigorous equations for heterogeneous forests, they are not really used for complex simulations. Finally, whole stand models, based on general equations, cannot supply information about particular individuals in the forest.

In order to get information about individual plants during the forest evolution, the plant community model must be based on individuals. The Prognosis system [16] allows to simulate homogeneous and heterogeneous forests. This system is particularly used to predict the development of mixed species and multi-layered conifer stands in British Columbia.

CAPSIS, [17] and [18], integrates several types of forest growth and dynamical models - stand models, distance independent or spatially explicit tree models… - and provides management tools to establish and compare different silvicultural scenarios.

B. Andrieu and C. Fournier developed the ADEL-maize model in 1999, [19]. This crop simulation model considers the canopy as a set of individual plants. The 3D maize development model is combined to physical models computing micro-climates on the 3D structure. The growth process of plants with ADEL-maize is based on individual organs and uses the Graphtal engine developed by Streit, [20]. Graphtal handles classical L-System models and Open L-Systems, [21], that allow interactions between plants and their environment. ADEL-maize is particularly used to simulate plant morphogenesis depending on light availability.

Finally, in the past years, few Artificial Life simulators, appearing on the Web, have been able to build online plants and plant communities. Bruce Damer describes in [22] the Nerve Garden Project [23], and [24] presents a project of plant morphogenesis where artificial plants are integrated in virtual online worlds.

3 Plant Modeling

Most of the models described in section 2 are used to simulate plant community development, but usually, they are not really devised to simulate the evolution of these communities on several generations. This paper focuses on the competition and cooperation behaviors during several generations. Simulations with the models described in the previous part would be definitely too long in order to obtain results in a reasonable amount of time. The next section, then, proposes a definition of a simpler plant model to perform long-term simulations. The main goal of this model is to optimize the possibility of interactions between plants, to observe competitive and cooperative behaviors in long-term simulations.

3.1 Plant Model

The plant model defined in this paper is based on multi-agent systems [8]. 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 agent, as a plant organ, is defined by a section of genetic code determining its properties and the way it grows according to its interactions with its environment. The overall life cycle of the plant is simulated by this model : seed, growth, death and decay. According to its current life phase, a plant is composed either of one unique organ, a seed, or of three organs, a foliage, a stalk and a root. Fig. 2 shows these three organs. 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.

In this model, the foliage represents all the herbaceous leaves and flowers. This organ contributes to the photosynthesis task of the plant and assumes its reproduction role. Although this representation is not realistic from a strictly biological point of view, the foliage has been placed on the plant top, this choice deeply simplifying computations and increasing the effects of sunlight competition between plants. Foliages have several branches, with their own length and height, growing according to the light received and the possible encumbrance of the plant's closed space.

The stalk defined in this model doesn't have any ramifications. Its main role is to transfer resources between the foliage and the root. This organ is particularly fundamental for light competition - longer is the stalk and higher is the foliage.

The root is the only organ interacting with the ground and able to extract minerals and chemical substances. Although in nature, according to coexisting plant communities, several root stratum may appear, this model uses only one, growing vertically and horizontally.

Before developing a root, a stalk and a foliage, a plant's first phase is the seed. This seed can remain asleep for a long period of time, until closed environmental conditions cause its growth and change the phase of the plant life cycle.

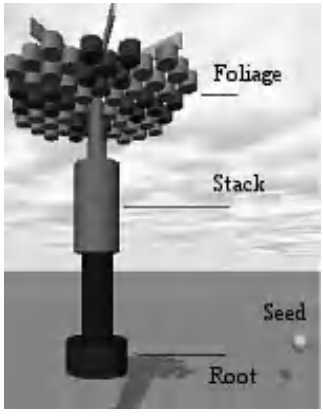


Fig. 2. Plant organs

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 according to its genetic parameters. 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.

Photosynthesis is the process by which plants increase their carbon storage by converting the 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 the computing time was a strong constraint in this application, radiosity and ray-casting algorithms have not been implemented for light diffusion.

Although Genetic Algorithms, first described by Holland [25] and largely developed by Goldberg [26], are not yet implemented in the simulator, plants are coded by an artificial genome. This coding will allow in a further development to simulate artificial selection between individuals and between species along generations. The genotype of a plant is divided into five chromosomes, a global chromosome for the plant itself and one for each of its parts, the seed, the root, the stack and the foliage. Each chromosome is also divided into several genes representing plant or organ properties. Sexual maturity of the plant, resource distribution thresholds between organs, maximum organ age or growth speed are some examples of parameters coded in this artificial genome.

Finally, 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. A genetically coded threshold is defined then when the seed is freed in the plant neighborhood. The distance to which the seed is spread is randomly determined inside an interval, coded in the genome, and the direction is a random value up to 360°.

3.2 Plant Community Model

All the plants are disposed in a virtual environment, defined as a particular agent, composed of 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.

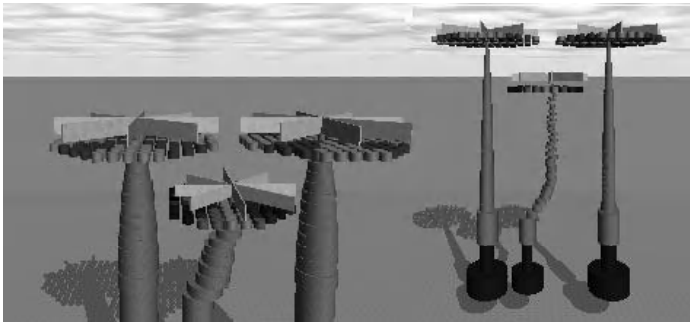


Fig. 3. Plant growth

The sky model is only used at this time to distribute and compute the amount of light plants receive at each stage. Fig. 3 shows three closed plants growing according to light and physical encumbrance.

The ground contains the second resource used by plants to grow, the minerals. The ground space is homogenously cut in several layers composed of voxels, simple 3D cubes. Fig. 4 shows the successive ground layers and voxels where information about

mineral types and concentration is stocked. Minerals are a renewable resource. This parameter is given by the user for each simulation. The resulting mineral concentration in a ground voxel varies according to three parameters :

- The local plant mineral extraction, depending on the root's horizontal and vertical size.
- The plant decomposition when a death occurs.
- The diffusion equation between nearby voxels.

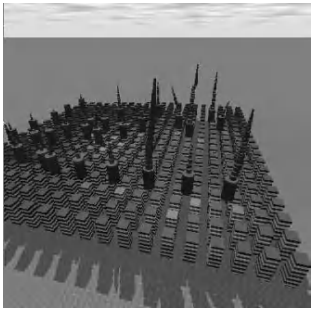


Fig. 4. Ground Layers

3.3 Allellopathy Model

Allellopathy is modeled by the interactions of biochemical substances, noted as BSs, between plants and ground voxels. Foliages, stacks and seeds secrete their BSs on the ground top layer, while roots secrete them on their corresponding layers. The amount of BSs that each organ puts down on ground voxels at each time step is a parameter coded in their respective chromosomes.

At each time step, a BS is diffused in the ground on nearby voxels according to the equation (1) :

$$C^{t+1}_{(x,y,z)} = C^{t}_{(x,y,z)} + \sum_{\substack{x-1\le x'\le x+1 \\ y-1\le y'\le y+1 \\ z-1\le z'\le z+1}} (C^{t}_{(x,y,z)} - C^{t}_{(x',y',z')})^{+} . G_{x-x',y-y',z-z'} \qquad (1)$$

Whereas C(x,y,z) is the concentration of the BS at the voxel position (x,y,z), (x',y',z') is the position of a nearby voxel of (x,y,z) and G is a Gravity matrix of coefficients controlling the diffusion movements in the different ground layers. This matrix allows a greater vertical diffusion than a horizontal diffusion, especially from top to bottom.

Fig. 5 shows the diffusion of BSs in the ground. Only seeds and roots are represented on this figure. Light blue areas indicate a high BS concentration.

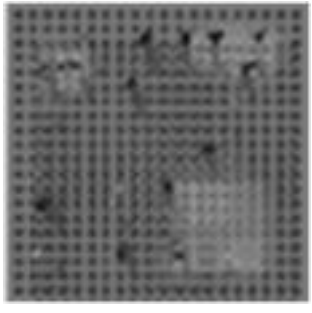


Fig. 5. Biochemical substance difffusion in the ground

At each time step, roots absorb a part of the BSs contained in their corresponding voxels and distribute a part of them to nearby organs, according to genetic parameters present in the root chromosome. Then, if the amount of BSs contained in the plant is included in an interval, genetically coded, a particular effect occurs, like plant death.

Finally, using allellopathy to better compete against other plants has a cost : carbon storage decreases according to the amount of BS in the plant and to a coefficient defined in the genotype, as shown in the equation (2).

$$Allellopathy\_Cost = BS\_stock.Dec\_Carbon\_Coeff \quad (2)$$

4 Simulations

This plant simulator is developed with the Delphi 5 language, and uses OpenGL libraries for the 3D representations and the 3D Open Dynamics Engine built by Russel Smith, ODE, [27]. Two different sets of 20 simulations were performed on PCs – 1.7 GHz, 256 MB RAM.

Previous experiences, described briefly in [28], validated the plant development phases, the ground mineral diffusion processes, and particularly the long-term light competition between two plant species.

The two sections of this chapter show the results of two sets of experiments. The first, a simple simulation, presents the competition of trees for space using allellopathy. The second set of experiments shows two kinds of temporal cooperation of three species during several generations.

4.1 Seed Germination Control Using Allellopathy

The goal of this first experiment is to control the seed germination of plants of the same specie with allellopathy. As shown in [28], plant positions in space are particularly defined by light, mineral and space competition, but allellopathy is also a natural tool used by plants to control the growth of their neighbors. This experiment is based on observations presented in [29]. This website describes the evolution of plants in the desert biome and their relations in the space. It shows that allellopathy is one of the main factors of this competition. Several plant species secrete toxins in their leaves and in their roots to kill other plants trying to implant themselves in their growing area.

In order to impose the fact that allellopathy is the only process to control the space competition, the plant seed lifetime is long enough to be considered as infinite. In that way, conclusions obtained in [28] about the role of seeds in this competition are masked.

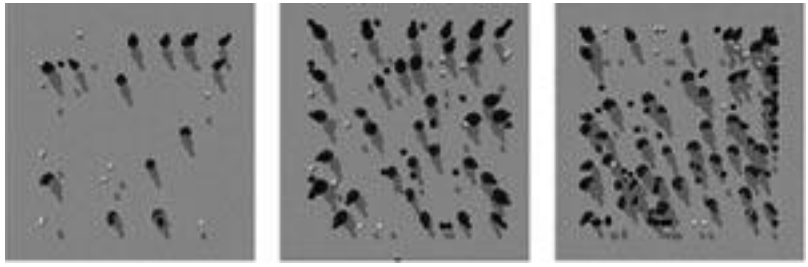


Fig. 6. Space density of plants according to allellopathy

Fig. 6 shows the space density of plants according to three genetic values of the allellopathic secretion parameter : 1.5, 1 and 0.7, the respective number of roots is 13, 35 and 52. In this experiment, the forest density is clearly controlled by the allellopathic secretion value determined by the plant genotype. The higher this parameter is, the lower is the plant density in the simulation space. A seed defines around itself a forbidden area for other plant germination, an area more or less wide according to the BSs produced and to the ground renewal parameter which dissipates BSs with time. Although the plant density is controlled by allellopathy, this process doesn't control precisely a regular repartition of plants in the space.

4.2 Temporal Cooperation of Three Plant Species Using Allellopathy

In nature, some plant species coexist on the same territory with a temporal kind of cooperation : only one specie grows at a time, then another one, then again another one, and so on. Moreover, some cycles between different plant species may emerge from this situation.

At least two situations can cause this phenomenon. The first one is a strict temporal competition between plants for space as described in the previous part. In that way, the cooperation is only a secondary effect of this competition. Plants compete and disturb the evolution of other plants by using BSs. The other situation is a temporal cooperation allowing plants to modify the ground composition during their life. These changes on the ground cause the apparition of other plants, which then modify the ground, and so on, establishing in some cases a life cycle.

These experiments are based on two examples. The first one, in [7], is the fir plantation composition depending on the interactions of firs, beeches and spruces. The second one is the dune evolution, observed in [30], where several plant species succeed each other by changing the dune ground composition.

As plant morphology is not the focus of this paper, no detail is given in this section. Three plant species are defined genetically with different BS values. One seed of each of the species is spread in the environment at the beginning of the simulations. Seed lifetime is set to a high value allowing plant sleeping during long time periods.

In the first set of experiments, each plant BS produces a negative effect on plants belonging to other species, while in the second set of experiments, the BSs only modify the ground composition. This alteration, then, allows other species to better grow, replacing the first ones. Fig. 7 shows a typical result of three plant species simulations using allellopathic negative effects.

Results shown in the Fig. 7 are cut in two phases. In the first phase, between time 0 and 500, plants of all species are growing without any organization, until the simulation space is filled. Then during the second phase, after 500 time steps, the different plant species organize themselves in persistent cycles. In this experiment, if a plant specie disappears, the artificial ecosystem will not collapse but, instead, will auto-organize itself again to enter later in another cycle with eventually the re-apparition of the lost specie.

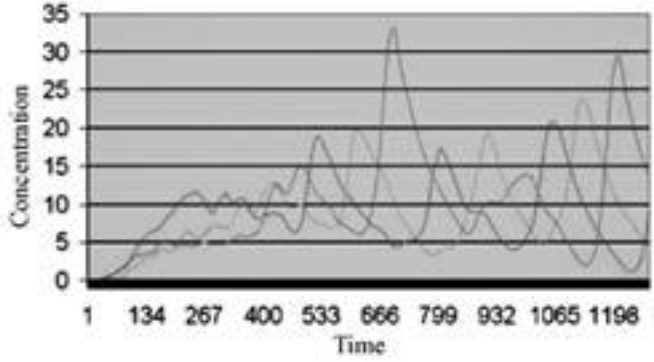


Fig. 7. Three plant species auto-organization

In the next set of experiments, a temporal dependence relation is built between the three species, i.e. to grow, each specie needs a ground modification caused by BSs of another specie. Results are quite similar with the Fig. 7 except for two points : first, the phase between 0 and 500 time steps doesn't exist, the ecosystem is auto-organizing itself immediately, and secondly, if a specie disappears then the whole ecosystem is affected and falls off. This means that if a plant specie disappears, the successor specie is not able to grow and the cycle is broken.

In these experiments, allellopathy is used by plants to auto-organize their development according to other plant species. This competition / cooperation method leads to more or less robust life cycles, depending on the dependency degree between each of the species present in the space simulation.

5 Conclusion and Future Works

This paper presents simulations of heterogeneous plant communities evolving during long time periods. The plant model is based on multi-agent systems in order to distribute specific organ tasks among simple agents and to use communication processes, like in nature, between these agents. Virtual plants grow in a 3D dynamic continuous environment. A specific model of allellopathy is developed in order to produce competitive and cooperative behaviors.

This study is based on observations of real conditions, described in [7], [25] and [26]. Results obtained are close to those observed in nature. The allellopathy model used in these simulations allows plants to compete for space on long time periods, but also to cooperate by interacting with each other using biochemical substances spread in the ground.

Three ways of development will be studied with this plant community simulator. First, simulations will be sharpened by improving the plant and allellopathy models and adding some noise to the forest evolution. Human interventions, like branch or tree cutting and planting, will be implemented to study the effect of external interventions on the forests. These simulations will also be extended to other species,

especially to competitive relationships between plants and pathogen agents like mushrooms. Next studies will focus on mutualism and parasitism of micorhyzae relations between plants and mushrooms.

Secondly, the plant model defined by an artificial genotype may use in a further development Genetic Algorithms in order to study artificial selection on large forest evolutions.

Finally, the artificial ecosystem will be enhanced with mobile agents. As allellopathy is used by plants to interact with each other, biochemical secretions are also used by plants to protect themselves from animal aggressions.

Acknowledgements. The author would like to thank Jean Fack and Gregory Gallet, LIAP5-Alife team students, for the development of the simulation platform and all their work on this project, Russel Smith for the very useful 3D Dynamic Engine ODE and Elsa Serfaty who reviewed this paper.

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. Pline l'Ancien : « Histoire Naturelle », Arléa, Paris (1999).
3. Molish, H : « Der Einfluss einer Pflanze die andere Allelopathie », Fisher, Jena (1937).
4. Rice, E.L. : « Allelopathy », Academic Press, New York (1974).
5. Rice, E.L. : « Pest Control with Nature's Chemicals », University Oklahoma Press (1983).
6. Rice, E.L. : « Biological Control of Weeds and Plant Disease: Advances in Applied Allelopathy », University of Oklahoma Press (1995).
7. Boullard, B. : « Guerre et paix dans le règne végétal », Ed. Ellipse (1990).
8. Ferber, J., « Les systèmes multi-agents », Inter Editions, Paris (1995).
9. Lindenmayer, A. : « Mathematical models for cellular interaction in development », Parts I and II*.,* Journal of Theoretical Biology*,* Vol. 18:280-315 (1968).
10. Rozenberg, G., Salomaa, A. : « The mathematical theory of L-Systems », Academic Press, New York (1980).
11. Prusinkiewicz, P., Hammel, M., Hanan, J., Mech, R. : « Visual models of plant development », In Rozenberg, G., Salomaa, A. (eds.), Handbook of formal languages, Springer- Verlag (1996).
12. http://www.cpsc.ucalgary.ca/Research/bmv/vmm-deluxe/
13. http://www.cirad.fr/presentation/programmes/amap.shtml
14. Bouchon, J., De Reffye, P., Barthelemy, D. : « Modélisation et simulation de l'architecture des végétaux », INRA editions (1997).
15. Peng, C. : « Modelling Global Terrestrial Carbon Storage: Past, present, and future », The Institute of Geology, Chinese Academy of Science (Invited Lecture), Beijing, China (2000).
16. http://www.for.gov.bc.ca/research/gymodels/progbc/
17. De Coligny, F., Ancelin, P., Cornu, G., Courbaud, B., Dreyfus, P., Goreaud, F., Gourlet-Fleury, S., Meredieu, C., Saint-André, L.: « CAPSIS: Computer-Aided projection for strategies in silviculture », Proceedings of the Symposium on Reality, models and parameter estimation, the forestry scenario, Portugal (2002).
18. http://www.cirad.fr/presentation/programmes/amap/logiciels/capsis.shtml

19. Fournier, C., Andrieu, B., « ADEL-maize: an L-system based model for the integration of growth processes from the organ to the canopy. Application to regulation of morphogenesis by light availability », In Agronomie, Vol. 19 : 3/4 (1999) 313-327.
20. Streit, C. : « Graphtal user manual », In Switzerland: SIG computer graphics, University of Berne (1992).
21. Mech, R, Prusinkiewicz, P. : « Visual models of plants interacting with their environment », In Proceedings of SIGGRAPH '96, New Orleans, Louisiana, New York: ACM SIGGRAPH (1996).
22. Damer, B. : « Avatars », Peachpit Press (1998).
23. http://www.biota.org/nervegarden/index.html
24. Steinberg, D., Sikora, S., Lattaud, C., Fournier, C., Andrieu, B. : « Plant growth simulation in virtual worlds : towards online artificial ecosystems », In Proceedings of the First Workshop on Artificial Life Integration in Virtual Environment, Lausanne, Switzerland (1999).
25. Holland, J. : « Adaptation in Natural and Artificial Systems », MIT Press (1975).
26. Golberg, D. E. : « Genetic Algorithms in Search, Optimization, and Machine Learning », Addison-Wesley (1989).
27. http://opende.sourceforge.net/
28. Lattaud, C. : « Long term competition for light in plant simulation », in Proceedings of GECCO2003, in press (2003).
29. http://www.coll-outao.qc.ca/bio/Arizona/ecologie.html
30. Pelt, J.M. : « La vie sociale des plantes », Fayard (1989).

The Evolutionary Control Methodology: An Overview

M. Annunziato[1], I. Bertini[1], M. Lucchetti[2], A. Pannicelli[3], and S. Pizzuti[1]

[1]Italian Agency for New Technology, Energy and Environment (ENEA)
[2]University of Rome "La Sapienza"
[3]CS Communication Systems s.r.l., Italy
contact: mauro.annunziato@casaccia.enea.it

Abstract. The ideas proposed in this work are aimed to describe a novel approach based on artificial life (alife) environments for on-line adaptive optimisation of dynamical systems. The basic features of the proposed approach are: no intensive modelling (continuous learning directly from measurements) and capability to follow the system evolution (adaptation to environmental changes). The essence could be synthesized in this way: "not control rules but autonomous structures able to dynamically adapt and generate optimised-control rules". We tested the proposed methodology on two applications, the Chua's circuit and a combustion process in industrial incinerators which is being carried out. Experimentation concerned the on-line optimisation and adaptation of the process in different regimes without knowing the system equations and considering one parameter affected by unknown changes. Then we let the '*alife*' environment try to adapt to the new condition. Preliminary results show the system is able to dynamically adapt to slow environmental changes by recovering and tracking the optimal conditions.

1 Introduction

The ideas of evolution, complexity, intelligence and life reproduction have long been stimulating the collective thinking. Scientific approaches then become predominant on the formation of hypothesis and practices to answer to these basic questions. Research and development, inspired by mathematical and physical models of intelligence (Artificial Intelligence) and more recently of life itself (Artificial Life), are providing new tools and ideas for the solution of complex problems requiring evolving structures. In problems ranging from traffic regulation to energy process control and optimisation the not-adaptive approaches are not effective to solve the problem over the time. The uncontrolled variables, the process ageing, the unforeseeable effects caused by human errors, the evolution of the process, in most of the cases require the change of the basic model or the objectives, or even the whole strategy. To reach the goal of evolving structures, a continuous learning of the system from the environment is necessary but not sufficient, and the ability of the system to change its internal structure is needed. In short, we need information structures able to *evolve* in parallel to the process we are modelling. Since late 70's a new branch of theory has been introduced in the evolutionary system research: the genetic algorithms, starting from [15] and developed in different directions [13]. In these

P. Liardet et al. (Eds.): EA 2003, LNCS 2936, pp. 331–342, 2004.

approaches the algorithm structure is able to optimise a *fitness* function, or to optimise a winning strategy simulating some mechanisms of the genetic dynamics of chromosomes (reproduction, recombination, mutation, selection). These algorithms have been successfully applied in many technological and engineering problems, in order to solve optimisation [17] or design problems. The limitation of these approaches is that the internal structure of the information is generally static and defined/controlled by the author of the algorithm. The fusion of the concepts of genetics algorithms and the self-organisation brought about the concept of the *artificial life* [8][11][16] started in the 80's. For the first time, it has really opened the possibility to build evolving structures able to develop a completely new organisation of the internal information. Artificial life is generally applied to study biological and social systems using software simulators [16][18] and the basic concept is to leave the system with the necessary degree of freedom to develop an *emergent behaviour,* combining the genetics with other life aspects (interaction, competition, co-operation, food network, etc.). At present, artificial life (or *alife*) is used mainly to study evolution problems, but we think that it has the potential to generate information structures that are able to develop a *local intelligence*. With the term *local intelligence* we refer to an intelligence strongly connected to the environmental context (*the problem*) we need to solve. We are involved in the development of this kind of structures called *artificial societies* [8][12]. The goal of these structures is the solution of a specific class of complex problems (design and engineering [1][2][3]) which require evolving structures. The extensive use of energy presents a severe challenge to the environment and makes indispensable to focus the research on the maximization of the energy efficiency and minimization of environmental impact (in particular the reduction of NOx and CO emissions). In this context the combustion process control assumes an importance much more relevant with respect to the past, especially for the combustion plants where the pollutants emissions, the environmental impact and the energy efficiency are strictly related to the modality of the process management. The proposed methodology is based on dynamics-based classification and evolutionary optimisation. The principal features of the approach are: dynamics based, no intensive modelling (progressive training directly from the measurements), ability to follow the process evolution. In our proposal the process knowledge is developed directly by the system through the observation of the effects that the regulation actions (acted by the operators or any other existing control systems) have on the plant performance. The main processes which we are looking at for application of the evolutionary control in the context of combustion plants are: eco-sustainable energy processes, gas turbines, industrial combustion chambers, engines. At present we are using this methodology in order to develop a prototypal control system for a real waste incinerator plant.

2 The Evolutionary Control

In complex processes, one of the basic problem we have to face for control is the continuous evolution of the plant along the life (aging, maintenance, upgrading, etc..). This is a big problem for traditional control methods: being based on fixed optimisation rules, they don't take care of the evolution of the process during its life. In order to try to overpass this difficulty evolutionary computation techniques have already been applied to non stationary environments [9][14][19] and we developed an

evolutionary adaptive technique, named *evolutionary control*, oriented to the optimisation and control of complex systems in non stationary environments. The basic features of the methodology we propose are no intensive modelling (progressive training and updating directly from measurements) and capability to follow the process evolution. The essence could be synthesized by the sentence: "*not control rules but autonomous structures able to dynamically generate optimised-control rules*". In our proposal, the process knowledge is obtained directly by the system through measurements observation, and it's used to update a dynamic model of the process itself, which we call performance model (see figure 1). The basic concept consists in the realization of an artificial environment that lives in parallel with the process and that asynchronously communicate with it, in order to dynamically control and optimise it. We suppose to always measure from the process its current regulations and the related value of an observable quantity which we call performance and which represents the objective function of our optimisation. In this way measurements are composed by both process variables and performance. The system continuously gets measurements and the dynamic state description from the process and provides the process back with the control actions. The main blocks of the evolutionary control (fig. 1) are the *alife* environment and the performance model.

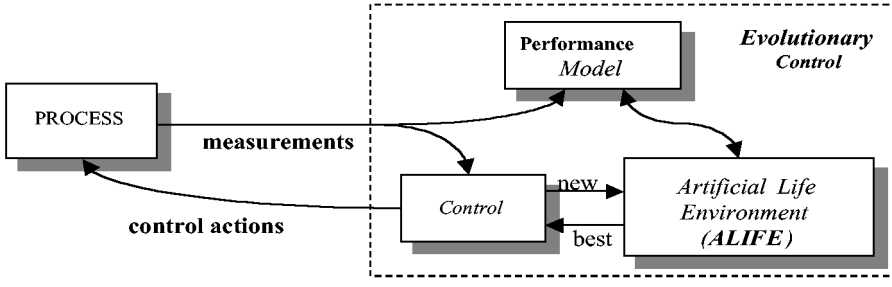


Fig. 1. Scheme of the evolutionary control

The first one is an artificial environment composed by individuals able to find the optimal solutions. The second one is a model of the process performance used by the *alife* environment in order to provide its individuals with a fitness value. The final control actions are the average between the best solution achieved by the *alife* environment and the current regulations. This is because we wish smooth transitions among different states. Each time a new measurement is acquired the performance model is updated with it (continuous learning) and a new individual, representing the new experimented/observed process condition, is inserted in the artificial environment. Thus the system is continuously updated, it follows the process not-monitored changes and drives the evolution towards better performances. Of course, at the beginning the system is not able to give any suggestion but it only learns from the process measurements. The artificial environment starts being active and giving right suggestions when the performance model is trained.

3 The Artificial Life Environment for Online Optimisation

In this paragraph we will describe the artificial life environment for the dynamic selection of the best control configurations. This environment derives from the Artificial Society approach illustrated and tested in [2] with several extensions (energy, filament, control) explained in the following. The *alife* context is a two-dimensional lattice (life space) divided in *nxm* cells (reference values: 200x200). The life space is empty at the beginning of the evolution. In the metaphor of the artificial life, this lattice represents a flat physical space where the artificial individuals (or autonomous agents) can move around. During the single iteration (life cycle) all the living individuals move in the space. Each individual is constituted by one single cell. Every life cycle, the individual moves in the life space, can interact with other individuals and can reproduce generating other individuals. We suppose to periodically carry out a set of measurements (measurement cycle), to calculate the current value of the process performance and to provide such an information to the control system. The performance is the target of the control we want to optimise and it is derived from measurements. At every cycle of measurement, a new individual is built on the base of the measured values and inserted in the environment with a starting value of energy (inlet energy). A measurement cycle corresponds typically to several life cycles (10-1000). This individual is considered as the ancestor of a new family or lineage. All the offspring generated by this individual are marked with the same code of lineage. The data structure of the individual is composed by three blocks: the genotype, the information and the status (see table 1).

Table 1. Parameters involved in the *alife* environment

Genotype	Status	Information
Irrationality	Age	Regulations
Dynamic model	x, y, dir, curv	Measurements
Fecundity	Wire description	
Lineage	Energy	
Generation level	Performance	
Average Lifetime		
Birth Energy		
Fighting energy		
Reproduction model		
Genome mutation		
Mutation rate, intensity		
Interaction model		
Marriage Probability		

a) individual description

Environment, Population & Control
Xsize
Xsize
Ysize
Border type
Measurement period
Inlet Energy
Max individuals
Random initial seed

b) global environment

The genotype includes a collection of behavioural parameters regarding dynamics, reproduction and interaction. These parameters don't change during the individual life. The information block includes a series of parameters related to the process to control: the regulation and measurement values; these variables don't change during the individual life. The status parameters include dynamics and structural parameters (position, direction, curvature, wire description), age, energy and performance values,

These parameters change during the individual life. The performance is continuously updated using an external problem-specific model. This is due to the possible changes in the unknown variables of the process not represented in the genotype.

3.1 Dynamics

The agent movement is the result of the composition of a deterministic component and a random component in order to define the curvature of the individual trajectory in the 2D lattice. The reciprocal importance of the two components is regulated by a parameter named irrationality. High values of this parameter cause a totally random movement; low values cause a totally deterministic movement. Several models for the deterministic component have been experimented. This model is important to give the individuals the chance of a local development (spiral trajectories) or a strong mixing of the overall population (quasi-straight trajectories). The spiral trajectories have given the best results in the experimentation. In the spiral model the trajectory curvature evolves slowly in time generating large and small trajectories. The irrationality parameter and the model type are recorded in the genotype; the x, y co-ordinates (float), the current direction and curvature are recorded in the status block. The dynamics of the individuals are also influenced by the space size (xsize and ysize) and the border type. The border can be open, closed or toroidal. The last has given the best results. In this configuration, when an individual touches one side of the space it enters from the other side. The dynamics parameters and the other probabilistic models in the *alife* environment are affected by the random sequence that is identified by its random initial seed. The short term evolution of the population can be influenced by the initial seed, but after a while, the results are substantially independent on this choice.

3.2 Reproduction

Both haploid and diploid reproduction models have been implemented. Reproduction can occur only if the individual has an energy greater than a specific amount named birth energy. During reproduction, an amount of energy (birth energy) is transferred from the parent(s) to the child. In the haploid reproduction a probabilistic test for self reproduction is performed at every life cycle. The fecundity probabilistic parameter is recorded in the genotype. In the diploid model reproduction is subjected to the event of meeting and marriage of two individuals. Only if the two meeting individuals derive by the same ancestor, a marriage can occur. The information block contains process data in terms of regulations and measurements. These values identify the control configuration pointed by the individual. The block of data is divided in two identical sections: individual features and ancestor features. In the individual features section, the actual values of the individual are recorded. In the ancestor section the values of its original ancestor are recorded. During reproduction, the ancestor section of information is copied in the corresponding section of the child. Only the specific section of the individual is subjected to change in the copying. In this way all the offspring of an ancestor are only one-generation level in respect to the ancestor. The rule can be relaxed accepting more than one generation level. In the haploid reproduction, a probabilistic-random mutation occurs on the regulations in relation to a mutation average rate and mutation maximum intensity. All the mentioned

parameters (lineage, fecundity, birth energy, mutation rate and intensity) are recorded in the genotype. In the diploid reproduction a crossover mechanism combines the regulations values proportionally to the performance value of the two parents. The application of the mutation mechanism on the genotype can change radically the individual behaviour and can increase a lot the possibilities to optimise the search strategy over time and situations. At the moment the mechanism has been applied to the dynamics and reproduction parameters (fecundity and irrationality). In the experimental results this introduction has increased the performance of the environment. Other important parameters for genotype mutation are the mutation rate and intensity. When the system is far from the optimum, high values for these parameters are necessary to speed up the environment to recovery the performance. When the system is close to the optimal low values are necessary to locate the control at the optimal maximum. The performance value of the ancestor has been derived from the measurements, but during reproduction we change regulations through mutations or crossover mechanisms. In this case we don't know the actual performance of the child we have built anymore. In order to solve this problem we need a performance model. This is a problem-specific external model. The inputs of this model are the new regulations and the output is the evaluated performance (see fig. 1). Due to the reproduction cost in terms of energy, the value of inlet energy is important to establish the initial diffusion of the lineage. This parameter is related only to the ancestor individuals. Its value, combined with the periods of measurement cycle determine the amount of energy admitted in the environment. Finally reproduction is limited by a maximum number of individuals in the space. In order to avoid population saturation this number is chosen quite higher than the natural fluctuations in the population dynamics.

3.3 Life Interaction and Selection

At every life cycle, several updates are performed. The individual performance is updated with the mentioned external model; age is increased and wire description (length, position of the wire nodes) is updated. For their transitory nature these parameters are recorded in the status block. When the age reaches a value close to the average lifetime (genotype), the probability of natural death increases. The ageing mechanism is very important to warrant the possibility to lose memory of very old solutions and follow the process evolution. This mechanism takes into effect the ageing of the process models due to the changes of non-monitored variables. Furthermore the value of the average lifetime determines the amount of the energy outgoing from the environment. Another mechanism of death occurs when the individual reaches the null energy. The energy can be lost for reproduction or for interaction with other individuals. When an individual collides with the body (wire) of another individual an interaction occurs. During this interaction, the two individuals compare their lineage. If they have the same lineage, they fight and selection occurs among mutations of solutions derived from the same measurements. If they have different lineage they ignore the interaction because they derive from different process configuration and selection is meaningless. In the scheme of fig. 2 the whole interaction-reproduction mechanism for haploid and diploid reproduction are reported. In case of fight, the winner is the individual characterised by a greater value of performance. The loser transfers an amount of energy (fighting energy) to the

winner. In the case of diploid reproduction model the two meeting individuals compare their regulations. If the difference is small, a marriage and reproduction can occur otherwise they fight. An alternative to this decision mechanism (interaction model) is based on a marriage probability. Figure 2 summarises the interaction model in case of haploid or diploid reproduction.

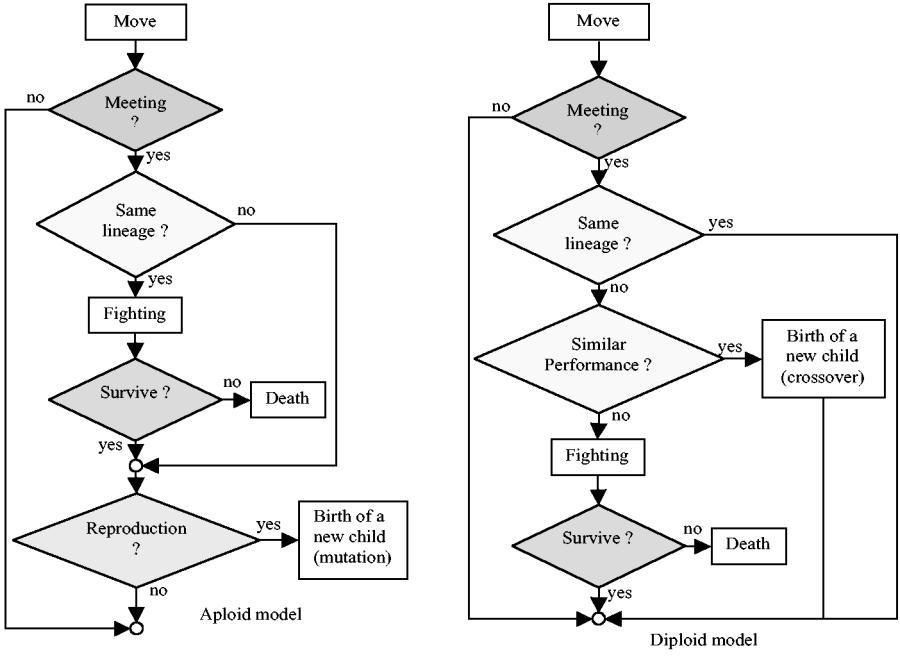


Fig. 2. Interaction-Reproduction scheme for haploid and diploid models

4 The Performance Model

The task of the performance model is to provide the newly generated individuals of the *alife* environment with a performance. The performance evaluator has in input the values of the control parameters and returns an estimate of the corresponding value of the fitness function. At present this module is implemented with a *performance map*. With this implementation the control parameters are discretized and the values we obtain are intended to be the performance table indexes. In the figure we show the situation with two control parameters indexing the X and Y axis of the table. We have two possible cases, depending on this discretization of the input parameters (fig. 3):

1. If the input refers to a cell in which we have already inserted a **measure** then we will consider that quantity as the estimate of the performance we are looking for;
2. If the input refers to an empty cell we use an **interpolation** mechanism, according to which if we point to an empty cell, we have to search for an already measured value in the neighbourhood, within a fixed range. In particular we stated as interpolation rule the following one.

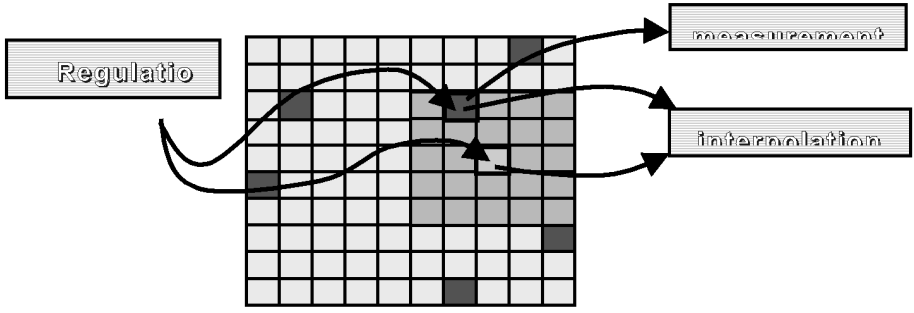


Fig. 3. Performance evaluation

In particular we chose a linear interpolation rule

$$K(d)\cdot M_f + (1-K(d))\cdot R \tag{1}$$

where M_j is the measure found in the neighbourhood of the empty cell, R is a random performance estimation of the point and K(d) is a weight coefficient exponentially decreasing with the distance between M_j and R in the parameters space. The performance map has a twofold task. On one hand it is, as we already said, the long-term memory of the control system; on the other hand, it allows the continuous updating of the reference model and so it lets the control system itself to evolve in parallel with the process. So it roughly represents an *internal knowledge of the real system*. The process of updating of the performance map is rather simple. When there is a new measurement, as we said above, this is inserted in the table in the cell which the discretization of the parameter set refers to. This happens regardless of the previous filling of that particular cell. However this simple knowledge implementation has several drawbacks. First the discretization of the map severely affects the performance of the system. Second it only allows of low dimensional systems, systems with a few control parameters which the performance depends on, because of memory allocation. Third the interpolation mechanism assumes the performance behaviour to be locally linear. In order to overcame these drawbacks a performance module based on *evolutionary neural networks* [4][6][7] is being carried out to replace the performance map.

5 Applications : The Chua's Circuit and Incinerator Plants

At first the *evolutionary control* has been tested on the well known chaotic Chua's electronic circuit [10]. In this experience we used some parameters to control the system, an external parameter as a unknown noise continuously varying and then we let the control system to tune the regulation parameters in order to continuously maximise the performance function (2).

$$\Im(\alpha, \beta, a, b) = 1 - \frac{|\theta - rms(\alpha, \beta, a, b)|}{\theta} \tag{2}$$

where θ is a threshold RMS reference value and rms(α, β, a ,b) is the RMS of the signal referring to the current values of the circuit's parameters α, β, a and b. In [2][3] detailed results about the on-line optimisation capability of the system are reported.

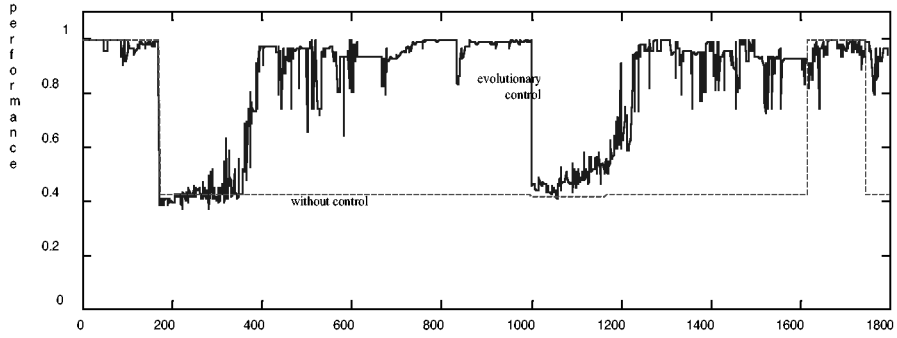


Fig. 4. Typical result of the performance recovery of the evolutionary control.

As example, fig. 4 shows the results of the control on the Chua's system that is characterized by 4 control parameters. In this experiment we try to regulate three parameters in order to recovery the performance of the circuit which is continuously disturbed by external changes on the fourth parameters. The curve of the performance (blue) obtained with the evolutionary control is compared with the curve of the performance (dotted red) in absence of any sort of control. In this experimentation the average performance for the uncontrolled situation is 0.5033 while in the controlled case it is 0.842 achieving a 34% improvement on the average performance.

At present [5] the '*evolutionary control*' approach is being applied on incinerator plants in the frame of the EcoTherm 5FP-EU project (2002-2004: "Evolutionary Control for Thermal sustainable processes"). Two plants for thermo-valorization of solid urban waste are considered (Ferrara plant in Italy and Rotterdam plant in the Netherlands). At present the work concerned the fitness definition (performance model) of the individuals of the *alife* environment and the development of a simulator of the incinerators plants. Control tests have been not carried out yet. As we saw before the performance model is aimed to provide the evolutionary controller with a global index of the performance which takes into account all the variables and constraints. In order to carry out the performance index it has been chosen to use fuzzy sets theory to properly define and compose the different variables and criteria. The fuzzy approach has been chosen because it allows operators transparency, it provides a well established theoretical framework and it is highly flexible (the definition for one plant can easily be transferred to a different plant with little effort). In this context the basic fuzzy sets, with the proper membership functions, have been defined (table 2) with the help of the process engineers, then two different composition criteria have been developed and finally performance has been defined.

Table 2. Fuzzy sets definition

Fuzzy set	Membership function
"Average steam flow rate '*good*' "	trapezium
"Average Steam flow rate '*stable*' "	gaussian
"Average O2 '*good*' "	triangle
"Average temperature '*good*' "	trapezium
"Average Nox emissions '*low*' "	trapezium
"Average CO emissions '*low*' "	trapezium
"Average flue gas rate '*low*' "	trapezium
"Average waste flow rate '*high*' "	ramp

The main idea driving the definition of the fitness criterion is that of having a flexible function capable to manage different criteria. In particular the fitness function will be the composition of two fuzzy sets describing two different requirements : 'optimality' and 'strictness'. The difference between the two lies in the composition of the previously defined fuzzy sets. The membership function of the first one will be defined as the weighted sum of the membership functions of basic fuzzy sets (3).

$$\begin{gathered} F_1 = X1 \oplus X2 \oplus X3 \oplus X4 \oplus X5 \oplus X6 \oplus X7 \oplus X8 \\ \mu_{F1}(x_1,x_2,x_3,x_4,x_5,x_6,x_7,x_8) = \sum w_i \mu_i(x_i) \\ \mu_{F1}(x_1,x_2,x_3,x_4,x_5,x_6,x_7,x_8) \in \Re,\ \mu_{F1}(x_1,x_2,x_3,x_4,x_5,x_6,x_7,x_8) \in [0,1] \\ w_i \in \Re,\ w_i \in [0,1],\ \sum w_i = 1 \end{gathered} \tag{3}$$

Where μ_i are the membership values of the eight basic fuzzy sets and w_i the weights to be properly set. Logically this operator represents a composition standing between AND and OR. This fuzzy set will fulfil the 'optimality' requirement because it allows to set the weights, the importance of each fuzzy set, according to the custom needs. In this way the optimiser will find the optimal solution for that particular setting of the weights giving the system scalability to different needs. The second criterion will concern the strict constraints satisfaction defined in the basic fuzzy sets. The resulting fuzzy set will be logically defined as the AND composition of the basic fuzzy sets (4).

$$\begin{gathered} F_2 = X1 \wedge X2 \wedge X3 \wedge X4 \wedge X5 \wedge X6 \wedge X7 \wedge X8 \\ \mu_{F2}(x_1,x_2,x_3,x_4,x_5,x_6,x_7,x_8) = \prod \mu_i(x_i),\ \mu_{F2}(x_1,x_2,x_3,x_4,x_5,x_6,x_7,x_8) = MIN(\mu_i(x_i)) \\ \mu_{F2}(x_1,x_2,x_3,x_4,x_5,x_6,x_7,x_8) \in \Re,\ \mu_{F2}(x_1,x_2,x_3,x_4,x_5,x_6,x_7,x_8) \in [0,1] \end{gathered} \tag{4}$$

It means that the resulting membership function will be the product or the minimum of the membership functions of the basic fuzzy sets. The final fuzzy set (5) describing the global fitness will be the weighted sum of the last two fuzzy sets.

$$\begin{gathered} F = F_1 \oplus F_2 \\ \mu_F(x_1,x_2,x_3,x_4,x_5,x_6,x_7,x_8) = w\mu_{F1}(x_1,x_2,x_3,x_4,x_5,x_6,x_7,x_8) + (1-w)* \\ *\mu_{F2}(x_1,x_2,x_3,x_4,x_5,x_6,x_7,x_8) \\ \mu_F(x_1,x_2,x_3,x_4,x_5,x_6,x_7,x_8) \in \Re,\ \mu_F(x_1,x_2,x_3,x_4,x_5,x_6,x_7,x_8) \in [0,1],\ w \in \Re,\ w \in [0,1] \end{gathered} \tag{5}$$

Weights will be defined by the developer according to the custom requirements depending on the relative importance of the two criteria. In figure 5 it is shown as example how the global index (5) behaves.

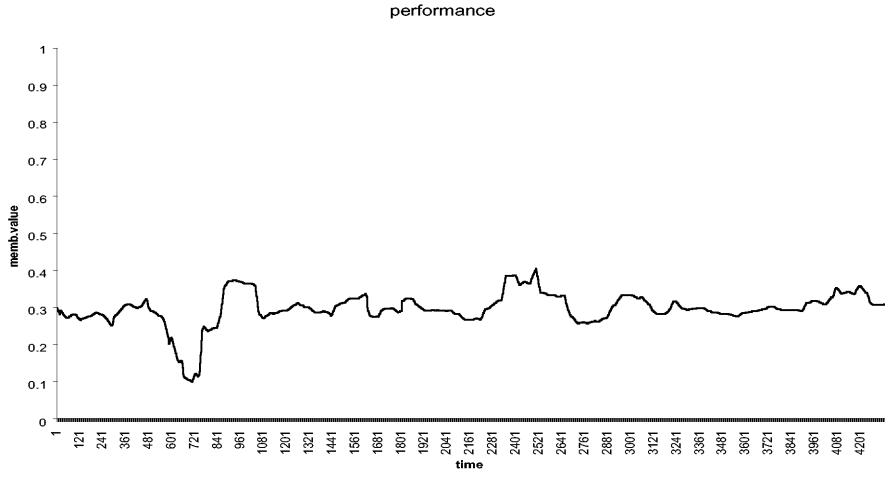


Fig. 5. Incinerator's performance behavior

6 Conclusion

We introduced a smart adaptive technique for control and optimisation of complex processes. We based our proposal on the development of an artificial environment evolving in parallel with the process. Exploiting its characteristics of evolution, biodiversity and adaptivity, we succeeded in achieving an on-line optimisation of the process via a continuous learning and updating of its model. We tested the proposed methodology on a very well known and widely studied dynamical system: the Chua's circuit. We defined a performance to maximise and experimentation concerned the on-line regulation of the energy of the load voltage signal in different regimes. We supposed not to know the equations describing the circuit, working on the rough signal generated by a simulator of the circuit. To test the adaptive capability of the *alife* environment we considered one parameter affected by unknown changes and then we let the *alife* environment to try to adapt itself to the new condition. In our experimentation we achieved a final 44% improvement of the average performance, with respect to the uncontrolled situation. Future developments of the methodology will concern the comparison with different control techniques and will be focused on studying the robustness of the control system to other non-stationary disturbances. In particular we will increase the frequency of variation of the unknown parameter. However, the major efforts will be devoted to construct an on-line learning model (we are thinking of an evolutionary neural network), able to predict the unknown values of performance and so to drive the system along an optimal performance path. Finally it has started the application of the proposed strategy on real waste incinerator in the framework of the EU 'Ecotherm' project.

References

[1] Annunziato M., Bertini I., Piacentini M., Pannicelli A., 1999, “Flame dynamics characterisation by chaotic analysis of image sequences” 36 Int. HTMF Institute, Sacramento/CA, USA

[2] Annunziato M. , Bruni C. , Lucchetti M., Pizzuti S., 2003 , “Artificial Life approach for continuous optimisation of non stationary dynamical systems”, Integrated Computer-Aided Engineering, IOS Press

[3] Annunziato, M., Bertini, I., Lucchetti, M., Pannicelli, A., Pizzuti, S., 2001 , Adaptivity of Artificial Life Environment for On-Line Optimization of Evolving Dynamical Systems, proc. EUNITE01, Tenerife, Spain.

[4] Annunziato M., Bertini I., Lucchetti M., Pizzuti S., 2003, “Evolving Weights and Transfer Functions in Feed Forward Neural Networks”, proc. EUNITE03, Oulu, Finland

[5] Annunziato M., Bertini I., Lucchetti M., Pizzuti S., van Kessel L.B.M., Arendsen A.R.J., 2003, “An evolutionary adaptive model for a MSWI plant”, Deliverable D4 of Task 1.4: Evolutive process model, EU Project ECOTHERM

[6] Annunziato, M., Bertini, I., Pannicelli, A., Pizzuti, S., 2003, “Evolutionary feed-forward neural networks for traffic prediction” , proc. of EUROGEN2003, Barcelona, Spain

[7] Annunziato M., Lucchetti M., Pizzuti S., 2002, “Adaptive Systems and Evolutionary Neural Networks : a Survey”, in Proc. EUNITE02, Albufeira, Portugal.

[8] Annunziato, M., , “Emerging Structures in Artificial Societies”, in Creative Application Lab CDROM, Siggraph, Los Angeles/CA, USA, 1999

[9] Branke J., 1999, “Evolutionary Approaches to Dynamic Optimisation Problems – A Survey”, GECCO Workshop on Evolutionary Algorithms for Dynamic Optimization Problems, A. Wu (ed.), 134-137

[10] Chua L.O., 1992, "The Genesis of Chua's Circuit", AEU 46, 250

[11] Emmeche, 1991, “The Garden in the Machine : The Emerging Science of Artificial Life”, Princeton University Press.

[12] Epstein J.M. and Axtell R.L., 1996, “ Growing Artificial Societies : Social Science from theBottom Up (Complex Adaptive Systems)”, MIT Press, Cambridge/MA, USA

[13] Goldberg D. E., 1989, “Genetic Algorithms in Search, Optimisation and Machine Learning”, Addison Wesley, USA

[14] Grefenstette J. J., 1992, “Genetic algorithms for changing environments”, R. Maenner and B. Manderick, eds, Parallel Problem Solving from Nature 2, North Holland, 137-144.

[15] Holland J. H., 1975, “Adaption in Natural and Artificial Systems”, MIT Press, Cambridge/MA, USA

[16] Langton. C., 1989, “Artificial Life”, C. Langton Ed. Addison-Wesley. pp. 1-47.

[17] Oliver and Smith D. and Holland J. R., 1987, “A study of permutation crossover operators on the travelling salesman problem”, Proc. of the 2nd International Conference on Genetic Algorithms, J.J. Grefenstette ed., Hillsdale/NJ, USA, 224-230.

[18] Rocha L.M., “Evolutionary Systems and Artificial Life”, Lecture Notes, Los Alamos National Laboratory, 1997.

[19] Trojanowski K. and Michalewicz Z., 1999, “Evolutionary Algorithms for Non-Stationary Environments'”, Proceedings of 8th Workshop: Intelligent Information systems, Ustron, Poland, ICS PAS Press, pp 229-240.

Modeling Selection Intensity for Linear Cellular Evolutionary Algorithms

Mario Giacobini[1], Marco Tomassini[1], and Andrea Tettamanzi[2]

[1] Computer Science Institute, University of Lausanne, Switzerland
{mario.giacobini, marco.tomassini}@unil.ch
[2] Information Technologies Department, University of Milano, Italy
andrea.tettamanzi@unimi.it

Abstract. We present quantitative models for the selection pressure on cellular evolutionary algorithms structured as a ring of cells. We obtain results for synchronous and asynchronous cell update policies. Theoretical results are in agreement with experimental values and show that the selection intensity can be controlled by using different update methods.

1 Introduction

Cellular evolutionary algorithms (cEAs) are an example of spatially structured evolving populations that is often used in optimization and other applications [1]. The structure may be an arbitrary graph, but more commonly it is a one-dimensional or two-dimensional grid. This kind of evolutionary algorithm has become popular because it is easy to implement on parallel hardware of the SIMD or MIMD type. Although SIMD machines have almost disappeared except for special purpose computing, cEAs can still be very conveniently implemented on computer clusters with excellent performance gains. However, what really matters is the model, not its implementation. Thus, in this work we will focus on cEA models and on their properties without worrying about implementation issues.

The theory of cEAs is relatively underdeveloped, although several results have been published on selection pressure and convergence speed. Sarma and De Jong performed empirical analyses of the dynamical behavior of cellular genetic algorithms (cGAs) [2,3]. Their work concentrated on the effect that the local selection method, the neighborhood size, and neighborhood shape have on the selection pressure. Rudolph and Sprave [4] have shown how cGAs can be modeled by a probabilistic automata network and have provided proofs of complete convergence to a global optimum based on Markov chain analysis for a model including a fitness threshold. We have recently studied the selection pressure behavior in cEAs on two-dimensional, torus-shaped grids [5].

Our purpose here is to investigate selection pressure in one-dimensional systems in detail. We study two kinds of dynamical systems: synchronous and asynchronous. For synchronous cEAs, some results are available, such as Sprave's hypergraph model [6], Gorges-Schleuter's study of evolution strategies [7], and

P. Liardet et al. (Eds.): EA 2003, LNCS 2936, pp. 345–356, 2004.

Rudolph's theoretical analysis [8]. We complete these results and extend the investigation to asynchronous linear cEAs, which, to our knowledge, have never been studied before from this perspective. In particular, we would like to be able to model observed takeover-time curves with simple difference equations describing the propagation of the best individual under probabilistic conditions.

Section two introduces linear synchronous and asynchronous cEAs. Next we define the concept of takeover time and the models predicting the selection pressure curves in the synchronous and asynchronous update methods. Empirical results and model accuracy are then discussed comparing the experimental curves to the predicted curves. Section 6 gives our conclusions and the future lines of this research.

2 Linear Cellular Genetic Algorithms

In a linear cEA the individuals (also called cells) are arranged along a line. Depending on whether the last and the first individuals communicate or not we have a ring or an array topology. Here we assume the first case, which is more common. Each individual has the same number of neighbors on both sides, and this number depends on the *radius* r. We will only consider the simplest case, $r = 1$, which means that there are three neighbors, including the central cell itself. Let us call S the (finite) set of the states that a cell can assume: this is the set of points in the (discrete) search space of the problem. The set N_i is the set of neighbors of a given cell i, and let $|N_i| = N$ be its size. The local transition function ϕ can then be defined as:

$$\phi : S^N \to S,$$

which maps the state $s_i \in S$ of a given cell i into another state from S, as a function of the states of the N cells in the neighborhood N_i. In our case, namely a line of cells with $r = 1$, ϕ takes the following form:

$$\phi(\cdot) = P\{x_i(t+1) \mid x_{i-1}(t),\ x_i(t),\ x_{i+1}(t)\},$$

where P is the conditional probability that cell x_i will assume at the next time step $t + 1$ a certain value from the set S, given the current (time t) values of the states of all the cells in the neighborhood. We are thus dealing with probabilistic automata, and the set S should be seen as a set of values of a random variable. The probability P will be a function of the particular selection and variation methods; that is, it will depend on the genetic operators. In this paper we model cEAs using two particular selection methods: binary tournament and linear ranking, but the same framework could easily be applied to other selection strategies.

A cEA starts with the cells in a random state and proceeds by successively updating them using evolutionary operators, until a termination condition is met. Updating a cell in a cellular EA means selecting two parents in the individual's neighborhood, applying genetic operators to them, and finally replacing

the individual if an offspring has a better fitness. Cells can be updated *synchronously* or *asynchronously*. In synchronous, or parallel, update all the cells change their states simultaneously, while in asynchronous, or sequential, update cells are updated one at a time in some order.

There are many ways for sequentially updating the cells of a cEA (for a discussion of asynchronous update in cellular automata see [9]). We consider four asynchronous update methods [9]:

- In *fixed line sweep* (LS), the n grid cells are updated sequentially $(1, 2 \ldots n)$.
- In *fixed random sweep* (FRS), the next cell to be updated is chosen with uniform probability without replacement; this will produce a certain update sequence $(c_1^j, c_2^k, \ldots, c_n^m)$, where c_q^p means that cell number p is updated at time q and $(j, k, \ldots, m)$ 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 used for each sweep through the array.
- In *uniform choice* (UC), the next cell to be updated is chosen at random with uniform probability and with replacement. This corresponds to a binomial distribution for the update probability.

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.

3 Takeover Time

To study the induced selection pressure without introducing the perturbing effect of variation operators, a standard technique is to let selection be the only active operator, and then measure the time it takes for a single best individual to conquer the whole population i.e., the takeover time [10]. A shorter takeover time thus means a higher selection pressure. Takeover times have been derived by Deb and Goldberg [10] for panmictic populations and for the standard selection methods. These times turn out to be logarithmic in the population size, except in the case of proportional selection, which is a factor of n slower, where n is the population size.

It has been empirically shown in [2] that when we move from a panmictic to a square grid population of the same size with synchronous updating of the cells, the selection pressure induced on the entire population is weaker.

A study on the selection pressure in the case of ring and array topologies in one dimensional cEAs has been done by Rudolph [8]. Abstracting from specific selection methods, he splits the selection procedure into two stages: in the first stage an individual is chosen in the neighborhood of each individual, and then, in the second stage, for each individual it is decided whether the previously chosen individual will replace it in the next time step. Rudolph derives the expected takeover times for the two topologies as a function of the population size and

the probability that in the selection step the individual with the best fitness is selected in the neighborhood.

Following Rudolph's hypothesis of non-extinctive selection methods, in this paper we study in detail the one-dimensional case. Moreover, we obtain results not only in the synchronous update case, but also for the asynchronous models presented in the previous section. Models for the growth of the best individual in the form of difference equations are presented in the next section. These models will then be compared with the experimental results in Section 5.

4 Models

Let us consider the random variables $V_i(k) \in \{0, 1\}$ indicating the presence in cell $i\,(1 \leq i \leq n)$ of a copy of the best individual ($V_i(k) = 1$) or of a worse one ($V_i(k) = 0$) at time step k, where n is the the population size. The random variable

$$N(k) = \sum_{i=1}^{n} V_i(k)$$

denotes the number of copies of the best individual in the population at time step k. Initially $V_i(1) = 1$ for some individual i, and $V_j(1) = 0$ for all $j \neq i$.

Following Rudolph's definition [8], if the selection mechanism is non-extinctive, the expectation $E[T]$ with $T = \min\{k \geq 1 : N(k) = n\}$ is called the takeover time of the selection method. In the case of spatially structured populations the quantity $E_i[T]$, denoting the takeover time if cell i contains the best individual at time step 1, is termed the takeover time with initial cell i. Assuming a uniformly distributed emergence of the best individual among all cells, the takeover time is therefore given by

$$E[T] = \frac{1}{n} \sum_{i=1}^{n} E_i[T].$$

In the following sections we give the recurrences describing the growth of the random variable $N(k)$ in a cEA with ring topology for the synchronous and the four asynchronous update policies described in Section 2. We consider a non-extinctive selection mechanism that selects the best individual in a given neighborhood with probability $p \in (0, 1)$.

4.1 Synchronous Takeover Time

In a synchronous cEA, at each time step k the expected number of copies $N(k)$ of the best individual is independent from its initial position. Since we consider neighborhoods of radius 1, the set of cells containing a copy of the best individual will always be a connected region of the ring. Therefore at each time step, only two more individuals (the two adjacent to the connected region of the ring) will contain a copy of the best individual with probability p. The growth of the quantity $N(k)$ can be described by the following recurrence:

$$\begin{cases} N(0) = 1, \\ E[N(k)] = \sum_{j=1}^{n} P[N(k-1) = j]\ (j + 2p). \end{cases}$$

Since $\sum_{j=1}^{n} P[N(k-1) = j] = 1$, and the expected number $E[N(k-1)]$ of copies of the best individual at time step $k-1$ is by definition $\sum_{j=1}^{n} P[N(k-1) = j]\ j$, the previous recurrence is equivalent to

$$\begin{cases} N(0) = 1, \\ E[N(k)] = E[N(k-1)] + 2p. \end{cases}$$

The closed form of this recurrence is $E[N(k)] = 2pk + 1$, therefore the expected takeover time $E[T]$ for a synchronous ring cEA with n individuals is

$$E[T] = \frac{1}{2p}\ (n-1).$$

Rudolph [8] gave analytical results for the ring with synchronous update only and for a generic probability of selection p. Although obtained in a different way, the previous expression and his equation (2) give nearly the same results for large population sizes n. In fact, his equation, for large n, reduces to $\frac{n}{2p} - \frac{1}{4}$, while our equation gives $\frac{n}{2p} - \frac{1}{2p}$. Given that the first term quickly dominates the second for large n, the two expressions are equivalent.

4.2 Asynchronous Fixed Line Sweep Takeover Time

Let us consider the general case of an asynchronous fixed line sweep cEA, in which the connected region containing the copies of the best individual at time step k is $B(k) = \{r, \ldots, s\}$, $1 < r \leq s < n$. At each time step the cell $r-1$ will contain a copy of the best individual with probability p, while the cells $s+j$ (with $j = 1, \ldots, n-s$) will contain a copy of the best individual with probability p^j. The recurrence describing the growth of the random variable $N(k)$, is therefore

$$\begin{cases} N(0) = 1, \\ E[N(k)] = \sum_{j=1}^{n} P[N(k-1) = j]\ \left(j + p + \sum_{i=1}^{n-j} p^i \right). \end{cases}$$

Since $\sum_{i=1}^{n-j} p^i$ is a geometric progression, for large n we can approximate this quantity by the limit value $p/(1-p)$ of the summation. The recurrence is therefore equivalent to the following one:

$$\begin{cases} N(0) = 1, \\ E[N(k)] = E[N(k-1)] + p + \frac{p}{1-p} = E[N(k-1)] + \frac{2p-p^2}{1-p}. \end{cases}$$

The closed form of the previous recurrence being

$$E[N(k)] = \frac{2p - p^2}{1 - p}\ k + 1,$$

we conclude that the takeover time for an asynchronous fixed line sweep cEA with a population of size n is

$$E[T] = \frac{1 - p}{2p - p^2}(n - 1).$$

4.3 Asynchronous Fixed and New Random Sweep Takeover Time

The mean behaviors of the two asynchronous fixed and new random sweep update policies among all the possible permutations for the sweeps are equivalent. We therefore give only one model describing the growth of the random variable $N(k)$ for both policies.

Let us again consider the general case in which the connected region containing the copies of the best individual at time step k is $B(k) = \{r, \ldots, s\}$ (with $1 < r \leq s < n$). The cells $r - 1$ and $s + 1$ have a probability p of containing a copy of the best individual at the next time step. Because of symmetry reasons, we consider only the part of the ring at the right side of the connected region. The cell $s + 2$ has a probability $1/2$ to be contained in the set of cells after cell $s + 1$ in the sweep, so it has a probability $(p/2)p$ to contain a copy of the best individual in the next time step. In general, a cell $s + j + 1$ has a probability $1/2$ to be after cell $s + j$ in the sweep, so it has a probability $(p/2)^j p$ to contain a copy of the best individual in the next time step. The recurrence describing the growth of the random variable $N(k)$, is therefore

$$\begin{cases} N(0) = 1, \\ E[N(k)] = \displaystyle\sum_{j=1}^{n} P[N(k-1) = j]\ \left(j + 2 \sum_{i=1}^{n-j} p \left(\frac{p}{2}\right)^{i-1} \right), \end{cases}$$

which can be transformed into the recurrence

$$\begin{cases} N(0) = 1, \\ E[N(k)] = \displaystyle\sum_{j=1}^{n} P[N(k-1) = j]\ \left(j + 4 \sum_{i=1}^{n-j} \left(\frac{p}{2}\right)^{i} \right). \end{cases}$$

Since $\sum_{i=1}^{n-j} (p/2)^i$ is a geometric progression, for large n we can approximate this quantity by the limit value $p/(2 - p)$ of the summation. The recurrence is thus equivalent to the following one:

$$\begin{cases} N(0) = 1, \\ E[N(k)] = E[N(k-1)] + \frac{4p}{2-p}. \end{cases}$$

The closed form of the previous recurrence being

$$E[N(k)] = \frac{4p}{2-p}\ k + 1,$$

we conclude that the expected takeover time for an fixed (or new) random sweep asynchronous cEA with a population of size n is

$$E[T] = \frac{2-p}{4p}\ (n-1).$$

4.4 Asynchronous Uniform Choice Takeover Time

To model takeover time for asynchronous uniform choice cEAs it is preferable to use cell update steps u instead of time steps in the recurrences. As for the other update policies the region containing the copies of the best individual at update step u is a connected part of the ring $B(u) = \{r, \ldots, s\}$ (with $1 < r \leq s < n$). At each update step the two cells $r-1$ and $s-1$ have probability $1/n$ to be selected, and each cell has a probability p, if selected, to contain a copy of the best individual after the selection and the replacement phases. The recurrence describing the growth of the random variable $N(u)$, counting the number of copies of the best individual at update step u thus becomes:

$$\begin{cases} N(0) = 1, \\ E[N(u)] = \displaystyle\sum_{j=1}^{n} P[N(u-1) = j]\ \left(j + 2\ \frac{1}{n}\ p\right), \end{cases}$$

which can be transformed into

$$\begin{cases} N(0) = 1, \\ E[N(u)] = E[N(u-1)] + 2\ \frac{1}{n}\ p. \end{cases}$$

We can easily derive the closed form of the previous recurrence:

$$E[N(u)] = \frac{2}{n}\ p\ u + 1.$$

Since a time step is defined as n update steps, where n is the population size, the expected takeover time for an uniform choice asynchronous cEA is

$$E[T] = \frac{1}{2p}\ (n-1).$$

We notice that the expected takeover time for a uniform choice asynchronous cEA is equal to the expected takeover time for a synchronous cEA.

It should be noted that the present asynchronous uniform choice update model is very similar to what goes under the name of *nonlinear voter model* in the probability literature [11].

5 Empirical Results

Since cEAs are good candidates for using selection methods that are easily extensible to small local pools, we use binary tournament and linear ranking in our experiments. Fitness-proportionate selection could also be used but it suffers from stochastic errors in small populations, and it is more difficult to model theoretically since it requires knowledge of the fitness distribution function. The cEA structure has ring topology of size 1024 with neighborhood of radius 1. Only the selection operator is active: for each cell it selects one individual in the cell neighborhood (the cell and its two adjacent cell at its right and at its left), and the selected individual replaces the old individual only if it has a better fitness.

5.1 Binary Tournament Selection

We have used the binary tournament selection mechanism described by Rudolph [8]: two individuals are randomly chosen with replacement in the neighborhood of a given cell, and the one with the better fitness is selected for the replacement phase.

Figure 1 shows the growth curves of the best individual for the synchronous and the four asynchronous update methods. We can see how, as the models derived in the previous section predict, the mean curves for the synchronous and the asynchronous uniform choice cases are superposed. Also the mean curves for the two asynchronous fixed and new random sweep show a very similar behavior. The graph shows that the asynchronous update methods give an emergent selection pressure greater than that of the synchronous case, growing from the uniform choice to the line sweep, with the fixed and new random sweep in between.

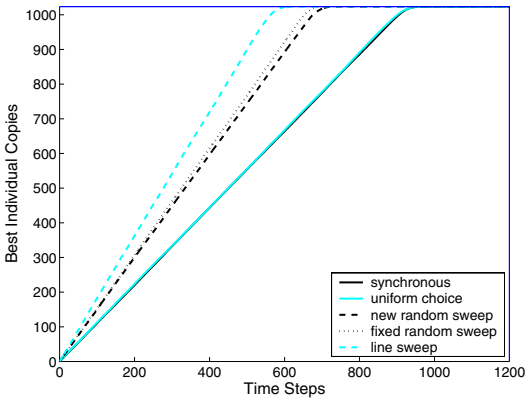


Fig. 1. Takeover times with binary tournament selection: mean values over 100 runs. The vertical axis represents the number of copies $N(k)$ of the best individual in each population as a function of the time step k.

The numerical values of the mean takeover times for the five update methods, together with their standard deviations are shown in Table 1, where it can be seen that the fixed random sweep and new random sweep methods give results that are statistically indistinguishable. The same can be said for the synchronous and the uniform choice methods.

Table 1. Mean takeover time and standard deviation of the tournament selection for the five update methods.

	Synchro	LS	FRS	NRS	UC
Mean Takeover Time	925.03	569.82	666.18	689.29	920.04
Standard Deviation	20.36	24.85	17.38	20.27	26.68

Since we use a neighborhood of radius 1, at most one individual with the best fitness will be present in the neighborhood of a considered cell, except for the last update when there are two of them. It turns out that the probability for an individual having a copy of the best individual in its neighborhood to select it is equal to $p = 5/9$. Using this probability in the models described in Section 3, we calculated the theoretical growth curves. Figure 2 shows the predicted and the experimental curves for the five update methods, and the mean square error between them.

Looking at the curves, it is clear that the models faithfully predict the observed takeover times. Moreover, the equivalence between new random sweep and fixed random sweep, as well as that of synchronous and uniform choice are fully confirmed.

5.2 Linear Ranking Selection

We have used a standard linear ranking selection mechanism. The three individuals in the neighborhood of a considered cell are ranked according to their fitnesses: each individual then has probability $(s - i)/s$ to be selected for the replacement phase, where s is the number of cells in the neighborhood ($s = 3$ in our case) and i is its rank in the neighborhood.

Figure 3 shows the growth curves of the best individual for the synchronous and the four asynchronous update methods. We can observe in the linear ranking case the same behavior that emerged in the binary tournament case: the mean curves for the synchronous and the asynchronous uniform choice cases are superposed, and the mean curves for the two asynchronous fixed and new random sweep show very similar behaviors. The graph shows that the asynchronous update methods give an emergent selection pressure greater than that of synchronous one, growing from the uniform choice to the line sweep, with the fixed and new random sweep in between. The numerical values of the mean takeover times for the five update methods, together with their standard deviations are shown in Table 2. Again, the results show that the two random sweep methods are statistically equivalent, which is also the case for the synchronous and uniform choice methods.

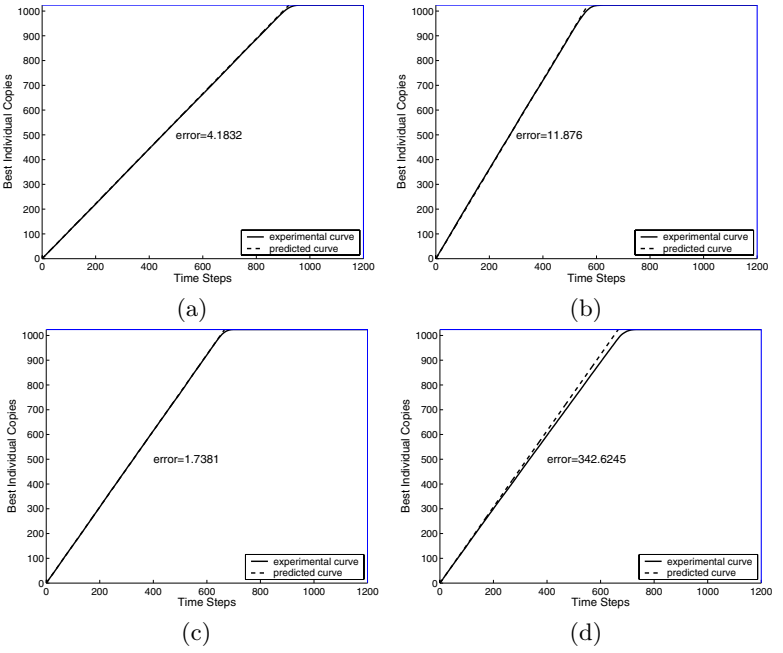


Fig. 2. Comparison of the experimental takeover time curves (full) with the model (dashed) in the case of binary tournament selection for four update methods: synchronous (a), asynchronous line sweep (b), asynchronous fixed random sweep (c), asynchronous new random sweep (d). Asynchronous uniform choice gives the same curve as the synchronous update, therefore it is omitted.

Table 2. Mean takeover time and standard deviation of the linear ranking selection for the five update methods.

	Synchro	LS	FRS	NRS	UC
Mean Takeover Time	768.04	387.09	519.92	541.14	766.5
Standard Deviation	17.62	19.21	14.26	14.48	25.44

With this linear ranking selection method, a cell having a copy of the best individual in its neighborhood has a probability $p = 2/3$ of selecting it. Using this value in the models described in Section 3, we can calculate the theoretical growth curves. Figure 4 shows the predicted and the experimental curves for the five update methods, and the mean square error between them. The agreement between theory and experiment is very good.

6 Conclusions and Future Work

We have presented quantitative models for the takeover time in cellular evolutionary algorithms structured as a ring with nearest neighbor interactions only. New results have been obtained for asynchronous cell update policies. The models are based on simple difference probabilistic equations. We have studied two

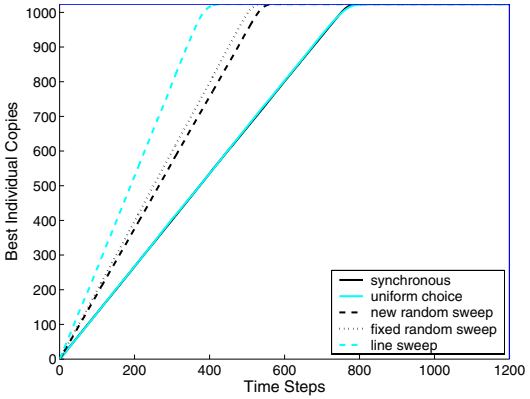


Fig. 3. Takeover times with linear ranking selection: mean values over 100 runs. The vertical axis represents the number of copies $N(k)$ of the best individual in each population as a function of the time step k.

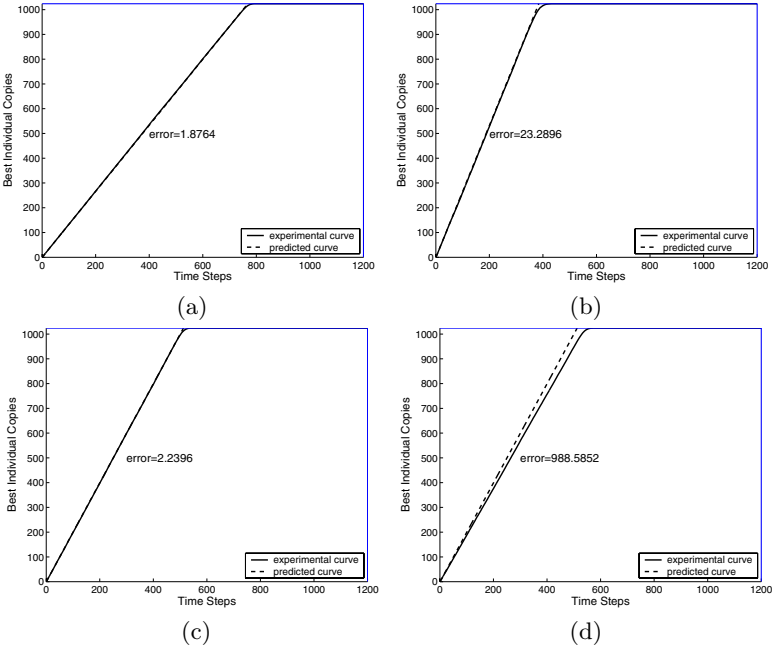


Fig. 4. Comparison of the experimental takeover time curves (full) with the model (dashed) in the case of linear ranking selection for four update methods: synchronous (a), asynchronous line sweep (b), asynchronous fixed random sweep (c), asynchronous new random sweep (d). Asynchronous uniform choice gives the same curve as the synchronous update, therefore it is omitted.

types of selection mechanisms that are commonly used in cEAs: binary tournament and linear ranking. With these selection methods, our results show that there is a good agreement between theory and experiment; in particular, we

showed that asynchronous cell update methods permit to control the selection intensity in an easy and principled way, without using ad hoc parameters.

In the future, we intend to extend this type of analysis to larger neighborhoods, and to more complex topologies such as two and three-dimensional grids, and to general graph structures. Moreover, we intend to investigate Markov chain modeling of our system and the relationships that may exist with probabilistic particle systems such as voter models.

References

1. Gorges-Schleuter, M.: On the power of evolutionary optimization at the example of ATSP and large TSP problems. In Husbands, P., Harvey, I., eds.: Fourt European Conference on Artificial Life, Cambridge, Massachusetts, The MIT Press (1997).
2. Sarma, J., Jong, K.A.D.: An analysis of the effect of the neighborhood size and shape on local selection algorithms. In Voigt, H.M., Ebeling, W., Rechenberg, I., Schwefel, H.P., eds.: Parallel Problem Solving from Nature (PPSN IV). Volume 1141 of Lecture Notes in Computer Science., Springer-Verlag, Heidelberg (1996) 236–244.
3. Sarma, J., Jong, K.A.D.: An analysis of local selection algorithms in a spatially structured evolutionary algorithm. In Bäck, T., ed.: Proceedings of the Seventh International Conference on Genetic Algorithms, Morgan Kaufmann (1997) 181–186.
4. Rudolph, G., Sprave, J.: A cellular genetic algorithm with self-adjusting acceptance thereshold. In: In First IEE/IEEE International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications, IEE, London (1995) 365–372.
5. Giacobini, M., Alba, E., Tomassini, M.: Selection intensity in asynchronous cellular evolutionary algorithms. In et al., E.C.P., ed.: Proceedings of the genetic and evolutionary computation conference GECCO'03, Springer Verlag, Berlin (2003) 955–966.
6. Sprave, J.: A unified model of non-panmictic population structures in evolutionary algorithms. In: Congress on Evolutionary Computation (CEC'99), IEEE Press, Piscataway, NJ (1999) 1384–1391.
7. Gorges-Schleuter, M.: An analysis of local selection in evolution strategies. In: Genetic and evolutionary conference, GECCO99. Volume 1., Morgan Kaufmann, San Francisco, CA (1999) 847–854.
8. Rudolph, G.: On takeover times in spatially structured populations: Array and ring. In et al., K.K.L., ed.: Proceedings of the Second Asia-Pacific Conference on Genetic Algorithms and Applications, Global-Link Publishing Company (2000) 144–151.
9. Schönfisch, B., de Roos, A.: Synchronous and asynchronous updating in cellular automata. BioSystems **51** (1999) 123–143.
10. Goldberg, D.E., Deb, K.: A comparative analysis of selection schemes used in genetic algorithms. In Rawlins, G.J.E., ed.: Foundations of Genetic Algorithms, Morgan Kaufmann (1991) 69–93.
11. Durrett, R.: Lecture Notes on Particle Systems and Percolation. Wadsworth, Belmont CA (1988).

Research of Complex Forms in Cellular Automata by Evolutionary Algorithms

Emmanuel Sapin, Olivier Bailleux, and Jean-Jacques Chabrier

Université de Bourgogne, 9 avenue A. Savary, B.P. 47870, 21078 Dijon Cedex, France
{olivier.bailleux, jjchab}@u-bourgogne.fr
emmanuelsapin@hotmail.com

Abstract. This paper presents an evolutionary approach for the search for new complex cellular automata. Two evolutionary algorithms are used: the first one discovers rules supporting gliders and periodic patterns, and the second one discovers glider guns in cellular automata. An automaton allowing us to simulate *AND* and *NOT* gates is discovered. The results are a step toward the general simulation of Boolean circuits by this automaton and show that the evolutionary approach is a promising technic for searching for cellular automata that support universal computation.

1 Introduction

Cellular automata are discrete systems in which a population of cells evolves from generation to generation on the basis of local transition rules. They can simulate simplified forms of life [1,2] or physical systems with discrete time and space and local interactions [3,4,5].

Wolfram showed that one-dimensional cellular automata could present a large spectrum of dynamic behaviours. In "Universality and Complexity in Cellular Automata" [6], he introduces a classification of cellular automata, comparing their behaviour with that of some continuous dynamic systems. He specifies four classes of cellular automata on the basis of qualitative criteria. For all initial configurations, Class 1 automata evolve after a finite time to a homogeneous state where each cell has the same value. Class 2 automata generate simple structures where some stable or periodic forms survive. Class 3 automata's evolution leads, for most initial states, to chaotic forms. All other automata belong to Class 4. According to Wolfram, automata of Class 4 are good candidates for universal computation [6].

Among the 2-D automata with uniform neighborhoods, the only binary one currently identified as supporting universal computation is Life, which is in Class 4. Conway et al proved in [2] its ability to simulate a Turing machine, using gliders and glider guns. The gliders are periodic patterns which, when evolving alone, are reproduced identically after some shift in space. Glider guns emit a glider stream that carries information and creates logic gates through collisions. The identification of new automata able to simulate logic circuits is consequently a

P. Liardet et al. (Eds.): EA 2003, LNCS 2936, pp. 357–367, 2004.

promising lead in the search for new automata supporting universal computation. In this paper, we show how evolutionary algorithms can be used to look for automata accepting gliders and periodic patterns and for glider guns in these automata. In the following, we present a new automaton, discovered by evolutionary algorithm, simulating logic gates.

Section 2 describes the framework of our study and the used convention. Section 3 describes an evolutionary algorithm seeking new automata supporting gliders and periodic patterns. The search, by evolutionary algorithm, for glider guns is described in Section 4. Section 5 describes a discovered automaton simulating logic gates. Finally, in Section 6 we summarize our results and discuss directions for future research.

2 Framework

2.1 Cellular Automata

In this work, we explore only cellular automata with the following specifications:

-Cells have 2 possible values, called 0 and 1,

-they evolve in a 2D matrix, called *universe*, and

-the state of a cell at a generation i depends only on the states of itself and its eight neighbors at the generation *i-1*.

We call *context of a cell* the states of the cell and its eight neighbors [7]. Thus, a cell can have 512 different contexts. A transition rule is defined as a Boolean function that maps each of the 512 possible contexts to the value that will be taken by the concerned cell at the next generation. Therefore the underlying space of automata includes 2^{512} rules among which are the automata of Bays Space [8] including Life.

A context A can be defined by a matrix of which the elements are $A_{i,j}$, i and j included in {*-1, 0, 1*}. This is represented in Figure 1.

$A_{-1,1}$	$A_{0,1}$	$A_{1,1}$
$A_{-1,0}$	$A_{0,0}$	$A_{1,0}$
$A_{-1,-1}$	$A_{0,-1}$	$A_{1,-1}$

Fig. 1. Representation of the cells of a context

Two contexts A and B are *symmetrically equivalent* iff there exist n such that for every i and j, $A_{i,j}$ is equal to the n^{th} element of the list *(*$B_{i,j}$, $B_{-i,j}$, $B_{i,-j}$, $B_{-i,-j}$, $B_{j,-i}$, $B_{-j,i}$, $B_{j,i}$, $B_{-j,-i}$*)*. A group of symmetrically equivalent contexts is a set of contexts including all the symmetrically equivalent contexts of one context in the set. There are 102 groups of symmetrically equivalent contexts.

The simulation of logic circuits by Life uses a glider moving in any direction, so we choose to consider only *isotropic rules*. An isotropic rule is a rule in which

symmetrically equivalent contexts have the same associated value. In the representation of a rule, each group is identified by one of its elements (cf. Figure 3) and the associated value of each context is represented by the absence or the presence of a point at its right. This representation is used in the following.

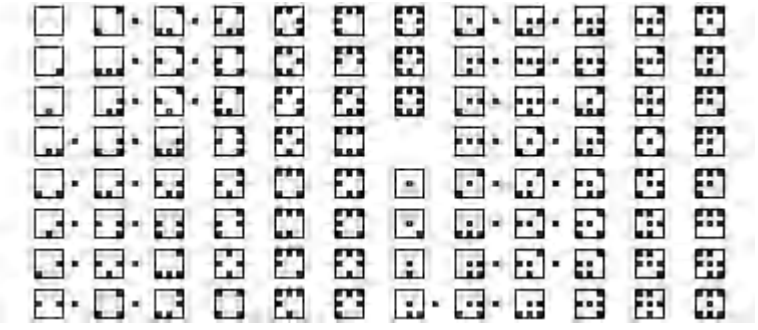


Fig. 2. Representation of a rule

We call *motif* a finite suit of Boolean matrix of dimension *k*k*. An automaton accepts a motif if said motif represents the successive states of the evolution of the first matrix by the transition rule of this automaton. The Life glider [9] is, for example, a motif accepted by Life. When every automaton supporting a motif maps the same values to a context *c*. We call *c* a *critical context* of this motif. For example, if a rule *t* supports a glider then all rules mapping the same values as *t* to the critical contexts of the glider support this glider.

3 A New Rule

This section describes the utilization of an evolutionary algorithm for the search for new rules accepting gliders and periodic patterns.

3.1 The Evolutionary Algorithm

Encoding and operators. Rules have been encoded by 512-bit strings in which all the bits of symmetrically equivalent contexts have the same value. The algorithm manages *A* rules. The bits of their transition function are randomly determined with a 0.5 probability. A mutation consists of modifying a randomly chosen bit, with the same weight for each of the 512 bits of a rule and likewise the bits of symmetrically equivalent contexts. Then, the isotropy of rules is kept. We implemented a simple crossover operator at a median point.

Fitness Function. The computation of the fitness function is based on the evolution, during *E* transitions, of a "primordial soup", randomly generated in

a square of C^*C centered in a U^*U space with a density D. After this evolution the primordial soup is the object of the following test, inspired by Bays' test [8] to search gliders [10]: each group of connected cells is isolated in order to determine if it is a glider or a periodic pattern. In this way, the following algorithm tests each cell using the parameter R (maximum radius of a motif):

```
N=2
While (there is at least one living cell in a space of radius N
       from the central cell) AND (N<R)
  N=N+1
End while
```

Figure 3 gives an example in which some cells are in a space of radius 2 from the central cell (i.e. crossed by the large inner square) and neither in a space of radius 3. N will be equal to 3 at the end of the test, we call motif of radius 3.

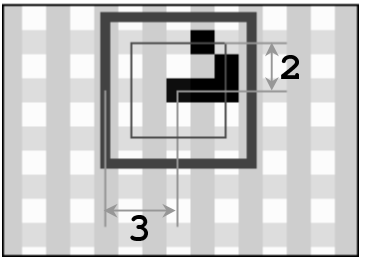


Fig. 3. Illustration of the search of a connected cells group.

Each group is inserted in an empty square of size T (size of the test space) and evolves during S generations (time of test). At each transition, the original pattern is sought in the test universe. Three cases can happen:

-The initial pattern has reappeared at its first location (this pattern is periodic).

-It has reappeared at another location (this pattern is a glider).

-It has not reappeared (it is then evolving).

The fitness function is evaluated as the multiplication of the number of occurrences of gliders by the number of occurrences of periodic patterns.

3.2 Evolutionary Algorithm

After the initialization of the A rules, the following cycle is iterated:

-The fitness function evaluates the rules.

-The X rules with the highest fitness function are kept.

-Y rules are created in the following manner, iterated Y times: choose one rule among the kept rules and mutate this rule.

-Z rules are created in the following manner, iterated Z times: choose two rules among the kept rules and apply crossover to these rules.

-A new population is obtained with the kept rule, the ones created by crossovers, and the ones created by mutations.

3.3 Parameters

Parameters of the fitness function. Cellular automata that we study evolve in infinite space. In the simulation, space must be closed, thus we chose the size U in order not to close the evolution of the primordial soup within the universe's limits. As well as this, we chose the test space size T in order not to close the motif evolution test with the test space limits.

The size C and the density D of the primordial soup were chosen such that automata evolution promote the appearance of the highest number of periodical patterns and gliders. We assume that some size and density values that promote glider appearance in Life may also fit other automata.

If during the soup evolution in Life the average number of cells decreases too quickly, gliders cannot appear, and thus it must decrease as slowly as possible. We notice that the cell number stabilization during the soup evolution in Life show that only gliders and periodic forms survive, and thus new gliders can not appear.

In Life (cf. Figure 4) we look at, for several soups size with density 0.5, the ratio evolution of living cell number per the initial cell number. This ratio is evaluated during 80 generations. For a primordial soup with size 20, the curve decreases quickly and then stabilizes whereas for the other sizes the decrease is slower and the stabilization does not happen before 80 transitions. The curves for primordial soups of size 40, 60, and 80 merge. The soup size increases the time of evaluation of the fitness function, thus we chose to generate the primordial soup in a size 40 square.

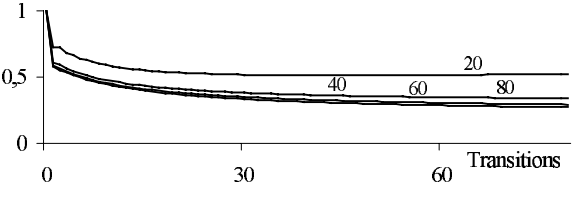


Fig. 4. Average over 100,000 evolutions of the ratio of the number of cells to the initial number of cells, during the evolution of primordial soups with density 0.5 by Life.

The evolution of primordial soups with size 40 by Life shows that the greatest number of cells must survive in order to increase the number of appearances of gliders and periodic patterns. Figure 5 gives the average number of living cells

during the evolution by Life of soups with size 40 and several densities. The decrease in the number of living cells is slowest for a 0.4 density, thus we chose this value for the D parameter.

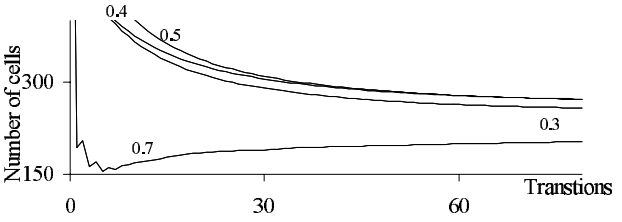


Fig. 5. Average number of cells during the evolution by Life of primordial soups of size 40 and of the indicated densities.

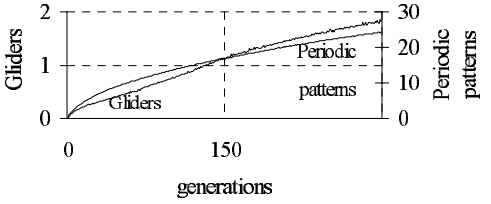


Fig. 6. Number of gliders (left axis) and periodic patterns (right axis) during the evolution by Life of primordial soups of size 40 and of density 0.4 evolving during the transitions from 0 to 300, for a maximum radius of 15 and a test time of 20.

The evolution time of a primordial soup is linked to the time of evaluation of the fitness function, thus it is important this time is not too long. However, during the evolution of a soup a certain time is necessary for gliders and periodic patterns to be created. We counted the average number of gliders and periodic patterns (cf. Figure 6) that appeared during the evolution of 100,000 primordial soups by Life. We can see that a glider appears on average after 150 generations. Thus, we chose this value for the E parameter.

The maximum radius of a motif must be determined in order to maximize the number of periodic patterns and gliders detected. A maximum radius that is too large slows down the calculation of the fitness function. In order to choose a radius, we generated several primordial soups evolving by randomly generated rules. During these evolutions, we counted how many gliders and periodic patterns had been detected with radius from 1 to 28. Figure 7 shows the percentage of motifs found for maximum radii from 1 to 28 (i.e., for a maximum radius of n only the motif with a radius less than or equal to n are counted). A maximum radius of 8 allows us to detect 85 percent of motifs; this value is chosen for the parameter R.

The test time S must be determined in order to maximize the number of periodic patterns and gliders detected. Periodic patterns and gliders with a period

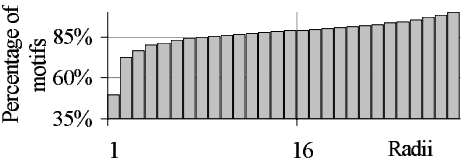


Fig. 7. Percentage of gliders and periodic patterns found for radius from 1 to 28 during the evolution of primordial soups of size 40 and of density 0.4 evolving until the transition 150 with r 20.

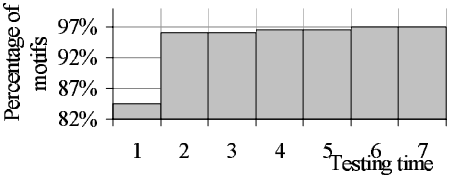


Fig. 8. Percentage of gliders and periodic patterns found for test time from 1 to 7 during the evolution of primordial soups of size 40 and density 0.4, evolving until the transition 150 with randomly generated rules, for a maximum radius of 8.

greater than S will not be detected whereas a test time that is too long slowed down the calculation of the fitness function. In order to choose the test time, we generated several primordial soups evolving with randomly generated automata. During these evolutions, we counted how many gliders and periodic patterns were detected with test time from 1 to 20. Figure 8 shows the percentage of motifs found for test time from 1 to 20. We notice that a test time of 6 allows us to detect 97 percent of motifs; thus this value is chosen for the test time.

Parameters of the selection. In order to determine the rate of mutation and crossover, different rates are tested by evolving our algorithm 100 times during 200 generations with a population of 50 automata. Figure 9 sums up the number of found automata for the following rates: 0, 20, 40, 60, and 80. The greatest number of automata is found for a rate of 80 for the mutation and a rate of 0 for the crossover. Thus, the simple crossover operator at a median point seems to be unsuitable for this algorithm.

3.4 Experimental Results

Figure 10 shows the number of discovered rules for 100 evolutions of our algorithm during 10, 20, 50, and 100 generations for population sizes 100, 50, 20, and 10, respectively. The number of discovered rules varies between 64 and 69 for different sizes of population. Thus this number is relatively constant for every population size tested but for a same number of evaluated automata.

The size A of the population must be chosen in order to optimize the number of automata found for a same number of automata evaluated by the fitness function. This result is presented is the next section.

Crossover \ Mutation	0%	20%	40%	60%	80%
0%	0	78	86	90	**98**
20%	41	67	79	93	
40%	30	69	88		
60%	28	87			
80%	10				

Fig. 9. Number of rules detected with 100 evolutions of our algorithm during 200 generations along with the indicated rates of crossover and mutation.

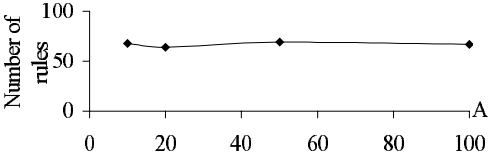


Fig. 10. Number of rules discovered by our algorithm for population sizes 10, 20, 50, and 100.

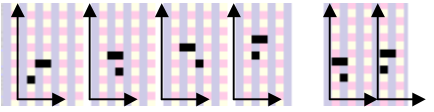


Fig. 11. Two gliders appear in 5.7 and 4.4 of the discovered rules by our algorithm and with, respectively, 11 and 9 critical contexts.

Each discovered automaton accepts one or more gliders. Some gliders appear in different automata. Figure 11 shows the gliders which appear the most often in the discovered rules with their appearance rate. Among the gliders discovered, these gliders have the least critical contexts. Their evolution goes through steps in which there are only three living cells that are the minimum (two cells define a line but not a direction).

4 Research of New Glider Guns

In the discovered rules, using evolutionary algorithms with parameters and structure like the one in Section 3, we looked for glider guns that spontaneously emerged.

4.1 New Evolutionary Algorithm

The initial population is now made up of several occurrences of a rule discovered by the algorithm in the previous sections. A glider of this automaton is identified. The bits associated to its critical contexts are frozen (i.e. a non critical context is chosen for the mutation). Thus every discovered automaton accepts this glider. The fitness function is now evaluated by the division of the number of glider

appearances by the total number of cells after the evolution time. During the detection of gliders, we look for glider guns. If the initial pattern has reappeared at its first location then it is removed and the leftover patterns are tested:

–If none rest then the pattern is periodic.

–If there is one or more gliders then a visual exam will determine if the initial pattern is a glider gun.

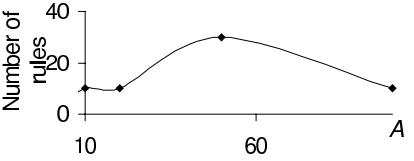


Fig. 12. Number of times when a glider gun is discovered by our algorithm for populations of sizes 10, 20, 50, and 100.

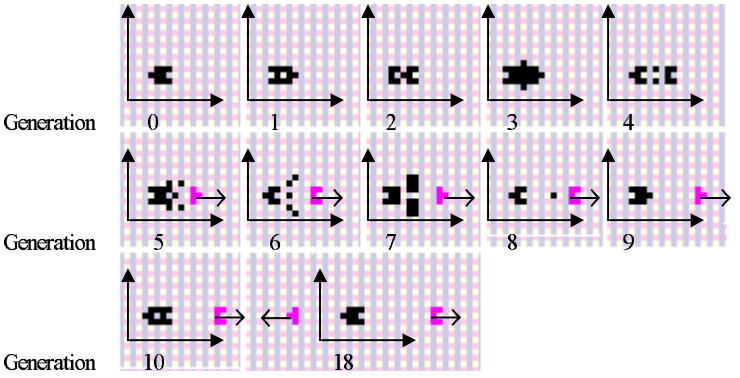


Fig. 13. A glider gun, discovered by evolutionary algorithm, with the gliders (lightly-colored) and its direction shown by the arrows.

4.2 Results

Figure 12 shows the number of glider guns discovered for 100 evolutions of our algorithm during 50, 100, 250, and 500 generations for population sizes respectively 100, 50, 20, and 10.

One of the discovered glider guns is shown figure 13.

Moreover, we found breeders which are patterns emitting one or more gliders but moving themselves.

5 Simulation of Logic Gates

In this section, we prove that one of the rules obtained using the previous algorithms, called R, can be used to simulate a logic gate. To the best of our

knowledge, it is the first time that such a result is shown with a 2D automaton with two states and Moore's neighbor, different from Life.

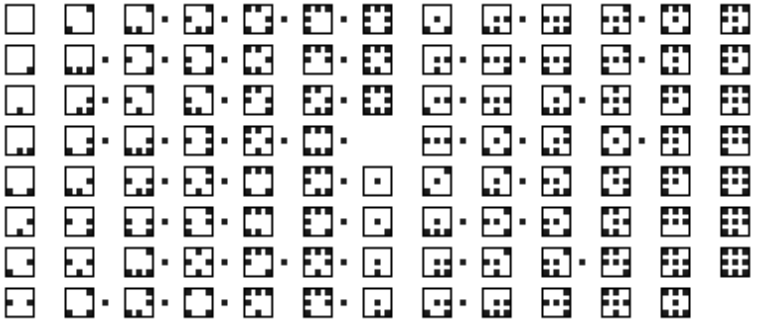


Fig. 14. The rule R.

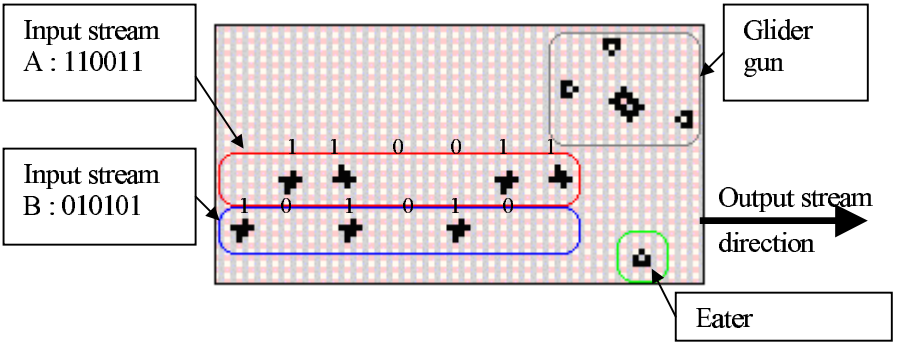


Fig. 15. A AND gate simulated by R.

The motifs used, in [9], to simulate a logic gate by Life are a glider gun, a glider, and an eater, i.e., a pattern that destroys the glider that crash it [11]. In R, we simulate a logic gate with these elements, Figure 15.

In this simulation, the streams of gliders symbolize the streams of information (i.e., the absence of gliders symbolizes the value 0 and the presence of gliders the value 1). The glider stream created by the gun crashes the stream A. If one glider is present in the stream A, this collision destroys the two gliders else the glider emitted by the gun continue its run. Thus the resulting stream of this first collision is equal to $NOT(A)$, and this collision creates an NOT gate. This new stream crashes the stream B and two streams are the results of this second collision including one which is the result of the operation *"A AND B"* and another one which is destroyed by the eater. The synchronization and the position of the different elements are, of course, essential for the good working of the simulation. Thus the automaton R is able to simulate a AND and a NOT gate. This simulation of logic gate is a step toward the general simulation of

logic circuits by R. This result is described in [12] using new patterns allowing us to duplicate glider streams [13] and to change their directions.

6 Synthesis and Perspectives

We developed an evolutionary algorithm that discovers new automata supporting gliders, periodical patterns, and glider guns. We adopted an elitist strategy for which the best results are obtained without crossover. We plan to test other evolution strategies and to try several crossover operators. Using evolutionary algorithms, we identified a new automaton that is able to simulate logic gates *AND* and *NOT*. This simulation is a step toward the general simulation of logic circuit. We plan to design an evolutionary algorithm able to discover automatically simulation of logic gates by automata. Later on, our aim will be the use of evolutionary algorithm for the search of new universal automata.

References

1. M. Gardner. "The fantastic combinaisons of john conway's new solitaire game " life ",". *In Scientific American*, 1970.
2. M. Gardner. "On cellular automata, self-reproduction, the garden of eden, and the game of life". *In Scientific American*, 224:112–118, 1971.
3. C. Dytham and B. Shorrocks. "Selection, patches and genetic variation: A cellular automata modeling drosophila populations". *In Evolutionary Ecology*, 6:342–351, 1992.
4. I. R. Epstein. "Spiral waves in chemistry and biology". *In Science*, 252, 1991.
5. Ermentrout, G. Lotti, and L. Margara . "Cellular automata approaches to biological modeling". *In Journal of Theoretical Biology*, 60:97–133, 1993.
6. S. Wolfram. "Universality and complexity in cellular automata". *In Physica D*, 10:1–35, 1984.
7. E. Sapin, O. Bailleux, and J.J. Chabrier. "Research of complexity in cellular automata through evolutionary algorithms". *Submitted to publication.*
8. C. Bays. "Candidates for the game of life in three dimensions". *In Complex Systems*, 1:373–400, 1987.
9. E. Berlekamp, J.H Conway, and R.Guy. "Winning ways for your mathematical plays". *Academic press, New York*, 1970.
10. M. Magnier, C. Lattaud, and J-C. Heudin. "Complexity classes in the two-dimensional life cellular automata subspace.". *In Complex Systems*, 11, 1997.
11. E. Sapin, O. Bailleux, and J.J. Chabrier. "Research of a cellular automaton simulating logic gates by evolutionary algorithms". *In EuroGP03.Lecture Notes in Computer Science*, 2610:414–423, 2003.
12. E. Sapin, O. Bailleux, and J.J. Chabrier. "Rapport technique lersia,université de bourgogne". *Publication interne*, 2003.
13. E. Sapin, O. Bailleux, and J.J. Chabrier. "A new approach of stream duplication in 2d cellular automata". *Proceeding of SCI2003.*

Genetic Feature Learning Algorithm for Fluorescence Fingerprinting of Plants

Marius C. Codrea[1,2], Tero Aittokallio[1,3], Mika Keränen[4], Esa Tyystjärvi[4], and Olli S. Nevalainen[1,2]

[1] Turku Centre for Computer Science, Lemminkäisenkatu 14 A, FIN-20520 Turku, Finland
[2] Department of Information Technology,
[3] Department of Mathematics,
[4] Department of Biology, Laboratory of Plant Physiology and Molecular Biology, University of Turku, FIN-20014 Turku, Finland
{marcod, teanai, mkeranen, esatyy, olli.nevalainen}@utu.fi

Abstract. Feature learning aims at automatic optimization of features to be used in the classification process. We consider the situation where, given a parameterized algorithm for extracting the features from the data, the optimizer tunes the parameters so that classification accuracy is maximized. The present paper extends our previous study [4] on feature learning problem by including two important mechanisms. First, an improved genetic algorithm (GA) with variable length chromosomes controls the size of the feature set. Second, the GA operates in conjuction with a neural network classifier for maximizing the identification accuracy. The performance of the feature learning algorithm is demonstrated with a problem of automatic identification of plant species from their fluorescence induction curves. The general approach should also be useful in other types of pattern recognition applications where *a priori* unknown characteristics are inferred from large feature spaces.

1 Introduction

The performance of machine learning in classification, function approximation or time series prediction is heavily influenced by the selection of the input variables. Numerous studies have addressed the reduction of input dimensionality or learning of parameters. Dimensionality reduction can be regarded at least from three perspectives: *feature extraction*, *feature selection* and *feature learning*.

Feature extraction refers to the problem of choosing a mapping f by which a sample x in an n-dimensional measurement space is transformed into a d-dimensional feature pattern y ($d < n$), such that some criterion J is optimized. Examples of common feature extraction paradigms are principal component analysis (PCA) and linear discriminant analysis (LDA) [6]. Mao and Jain [13] identified four categories of feature extraction methods (supervised and unsupervised, using linear or non-linear mapping functions) and compared various Neural Networks (NN) for feature extraction.

P. Liardet et al. (Eds.): EA 2003, LNCS 2936, pp. 371–383, 2004.

Feature selection means selecting a subset S out of a larger set F of candidate features under some selection criterion $J(S)$. There are two main approaches for feature subset selection, *filter* and *wrapper techniques* (for reviews, see [5], [10]). In filter techniques, the evaluation function relies on the properties of the data itself, independently of the target induction algorithm. In the wrapper approach, the intended algorithm is used in the evaluation of the subsets.

Feature learning adjusts a parameter vector θ of the given feature extractor $g(x, \theta)$ from the raw data x, with respect to some criterion C:

$$\theta^* = \underset{\theta}{\operatorname{argmax}}\ C[g(x, \theta)]\ . \tag{1}$$

Again, the optimization criterion can be either the accuracy of the induction algorithm itself (wrapper technique) or other appropriate, faster evaluation function (filter technique). In this general form, the concept of feature learning can be used in various machine learning applications. If the mechanism of generating features from the measurement space through a parameterized procedure is known, the process of learning features aims at discovering the parameter setting θ^* that provides an optimal feature set. In this framework, the search space expands from the implicit feature space by including all possible parameter values of θ. Because the search space is often very large, application specific heuristics can be used to reduce the computational effort.

To the best of our knowledge, little work has been done concerning feature learning despite the large number of open questions and possible applications. As an example, the problem of optimal approximation of digital curves by fitting primitives like straight lines, circular arcs or spline curves can be regarded as a learning process. Digital curves are decomposed into a minimum number of components by optimizing the parameters (e.g., endpoints, radius, control points) under an approximation quality constraint. Genetic algorithms (GAs) have been reported to perform well in this context, both using an arbitrary number and a fixed number of vertices [3],[8]. Piater and Grupen [16] introduced a feature learning process using feature primitives (2-D Gaussian filter responses) to construct increasingly complex and specific compound features like geometric and topological relations. These features were incorporated into a Bayesian network recognizer and the experimental results indicate good potential of the approach.

In a previous study [4] we applied feature learning in the context of plant species identification on the basis of their fluorescence induction curves. A GA was developed for determining a fixed-size set of features describing optimally the fluorescence curves for a NN classification. In the present paper, we extend our system by including the NN classifier in the evaluation process, while searching for an arbitrary number of features. The measurement space is explored and the parameters of the feature extraction process are optimized by constructing new features and modifying or discarding the existing ones, toward maximization of the classification accuracy. We propose a hierarchical, hybrid wrapper-filter, evolutionary method for automatic detection of an optimal set of discriminative features. At the first level of the method, independent features are evaluated using the Fischer interclass separability criterion that is based on the statistical

properties of the data (filter technique). The estimated contribution of individual features is used to reduce the search range by increasing the efficiency of the GA operators which traditionally proceed on random basis. At the second level, the complete set of features is evaluated using the NN classifier itself, which makes the overall process wrapper-like. We chose the NNs for fitness computation because it has been shown that, for this particular problem, they outperform other standard methods like k - nearest neighbors or Bayesian minimum distance [19]. In addition to the problem of feature optimization, we also investigate how short slices of the signal can be used to extract the features so that they still provide acceptable classification accuracy through the feature optimization process. Minimization of the length of the signal implies shorter data acquisition and processing time, which is desirable for real-time applications.

2 Feature Extraction from Fluorescence Induction Curves

The photosynthetic light reactions, occurring in the thylakoid membranes of plant chloroplasts or cyanobacterial cells, convert the energy of sunlight into chemical form. Chlorophyll *a* (Chl *a*) fluorescence is red and far-red light emission from the photosynthesis machinery, and changes in the fluorescence yield reflect changes in the photochemical energy conversion reactions of photosynthesis. Chl *a* fluorescence is a widely used research tool in plant biology (for reviews see [7],[12],[14]). The Chl *a* fluorescence fingerprint technique has been developed for automatic plant identification in precision farming and environmental monitoring [19]. In this method, a plant leaf or an algal or cyanobacterial sample is illuminated with a sequence of light pulses of different duration, colour and intensity. The time-course of chlorophyll *a* fluorescence yield, measured during the illumination sequence, is called a *fluorescence fingerprint curve* (FFC). It has been shown that accurate classification results can be obtained using regression lines as features that represent the FFC for pattern recognition [9],[19].

In this paper, by a *time interval* (TI) we refer to the time-points of the digital signal (FFC) within the endpoints defining the TI. Linear regression is performed over these signal values (y-coordinate) against the time-points (x-coordinate). The data is parametrized by computing two features for every regression line (reg): the *slope* (α) and the *y-coordinate cross point* (y_0), see Fig. 1C. In other words, a FFC is represented by a feature vector containing a pair (α, y_0) for each TI (in the simple case of Fig. 1C, only one TI and the corresponding regression line are shown). The six curves in Fig. 1B are the averages of 200 FFCs from six different plant species (nettle, tobacco, maize, rye, pine and haircap moss). Remarkable variation occurs among individual curves even from the same plant species. Differences in the average curves reflect average differences in the properties of the photosynthetic machinery. The difficulty of choosing an appropriate number of TIs and their positions becomes obvious by visual inspection of graph B of Fig. 1. Some TIs make good distinction among several plant species but lead to poor discrimination between others.

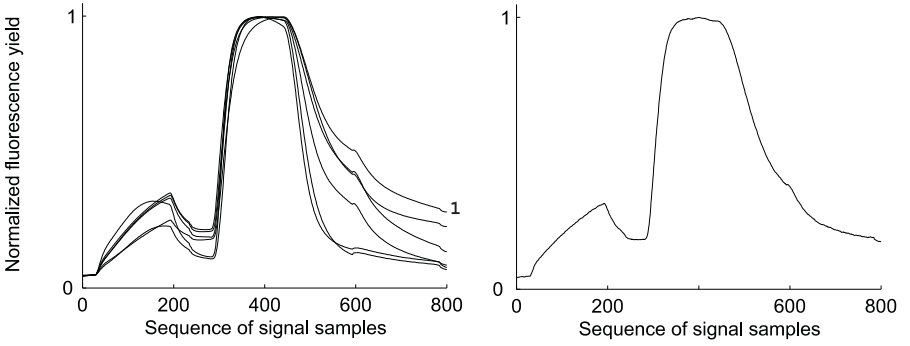


Fig. 1. (A) Three possible time intervals for estimating regression lines from FFCs. (B) Averages of normalized FFCs of six plant species: 1-nettle, 2-tobacco, 3-maize, 4-rye, 5-pine, 6-haircap moss. The sharp turnpoints of the original curves reflect shifts from one illumination phase to another, and the total illumination time is 3 seconds. (C) Extraction of features from a FFC measured from a tobacco leaf according to the TI of (D). The regression line (*reg*) with slope α and y-coordinate cross point y_0 corresponds to the samples of the original signal between the left (L) and right (R) endpoints of the TI

3 Feature Learning by a GA with Chromosomes of Variable Length

Due to their ability to explore the feature space in a global manner, GAs are excellent tools for feature selection tasks [1],[11],[17],[20],[22]. Unlike standard GAs, our method uses variable length, integer valued genotype representation for controlling simultaneously, in an adaptive manner, both the number of the selected features and their extraction parameters (the endpoints of the TIs). The evolutionary process adds new features, fine-tunes the existing ones and deletes irrelevant ones by adjusting the TIs.

3.1 Chromosome Representation

A chromosome encodes pairs of left and right endpoints of the TIs. Let *maxTI* be a predefined maximum number of TIs. A *valid chromosome* is an array of $2maxTI$ nonnegative integers, containing at least one pair of valid endpoints. A pair of endpoints is valid if the left endpoint is smaller than the right one and they both fall in the signal range. Pairs of zeroes denote the absence of features. Fig. 2 illustrates a valid chromosome containing the three TIs of Fig. 1A.

3.2 Initialization

An initial population of individuals is generated at random such that all the chromosomes are valid. For saving computation effort at early iterations, initial

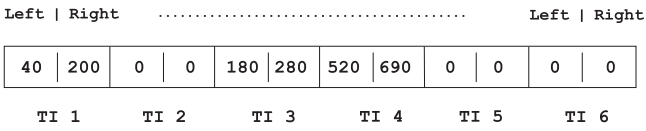


Fig. 2. A sample chromosome of maximum length six (TI 1-6), encoding the three time intervals of Fig. 1A. The time intervals TI 2, 5 and 6 are empty and denote no feature. Note that the order of the intervals can be arbitrary

chromosomes contain only few TIs (e.g. less than 75% of *maxTI*). This also facilitates the evolution since, according to our experience, it is easier for the GA to add new good features than to discard bad ones.

3.3 Genetic Operators

Fitness Function. Features are extracted from FFCs according to the TIs coded by an individual, as described in Section 2. The feature vectors are divided into three sets (learning, validation and testing) and a NN is trained using the backpropagation method [15]. The fitness function ranks the individuals of the population in terms of their percent *classification accuracy (CA)* on the test set:

$$CA = (\#\text{ of correct classifications } / \#\text{ of all samples}) * 100\ . \tag{2}$$

Among individuals sharing the same CA value, the best fitness value is assigned to the one containing the smallest number of features. The fitness value of an individual l is defined as:

$$fitt_l = CA_l - NTI_l/maxTI\ , \tag{3}$$

where NTI_l is the number of TIs encoded in the chromosome. The ratio $NTI_l/maxTI$ is a 'light' penalty term that ensures the discriminative behavior of the fitness function [1]. As $0 \leq CA \leq 100$ and $0 < NTI_l/maxTI \leq 1$, the penalty term interacts when comparing two individuals that have CAs within 1% apart.

Crossover. The performance of crossover operations can be improved by systematic selection of the best genes from two parents instead of random ones [8]. Our method ranks individual TIs (pairs of features) by the *scatter decomposition* to guide the *Best crossover* operator. The slope and the y-crossing point of the regression line are calculated for all FFCs according to the endpoints encoded by a pair of successive genes (left and right). The covariance matrix of the feature vectors measures the total scatter around the mean. The *within cluster scatter (W)* is the covariance matrix of the feature vectors of the same class. The *between clusters scatter (B)* is the covariance matrix of the class mean vectors against the mean of all feature vectors [18]. One can design different evaluation functions aiming at maximization of the between-cluster scatter and minimization of the within-cluster scatter. A suitable choice for such a function

is an expression comprising of the non-zero eigenvalues of the covariance matrices. The *trace* of a matrix is equivalent to the sum of its eigenvalues. We use the ratio $trace(W)/trace(B)$ as a measure of class segregation power of individual TIs [6]. The ratio is used to rank pairs of features encoded in chromosomes [4].

Let $X = (X[1], X[2], \ldots, X[2maxTI])$ and $Y = (Y[1], Y[2], \ldots, Y[2maxTI])$ denote two individuals selected from the population and $R = (R[1], R[2], \ldots, R[maxTI])$ and $Q = (Q[1], Q[2], \ldots, Q[maxTI])$ denote the ranks of the TIs of X and Y, respectively. We use two different crossover methods in our GA:

Best crossover derives a new individual (X') as follows:

$$\begin{aligned}&(X[2i-1], X[2i]) \rightarrow (X'[2i-1], X'[2i]) \text{ if } R(i) > Q(i),\\ &(Y[2i-1], Y[2i]) \rightarrow (X'[2i-1], X'[2i]) \text{ if } R(i) \leq Q(i),\ i = 1, \ldots, maxTI. \quad (4)\end{aligned}$$

One-point crossover takes an odd random number t from the interval [1, $2maxTI$ -1] and derives two new individuals by interchanging genes of X and Ysuch that index t is the common cutting point.

Mutation. The endpoints of the TIs are slightly changed by the mutation operations. The ranking mechanism ensures the perpetuation of the newly discovered good settings through the Best crossover operations. We employ three types of mutations which all alter randomly chosen individuals of the current population.

Random mutation selects a nonzero gene $X[i]$ (either left or right endpoint of a TI). With equal chance (1/2), the value $X[i]$ is either increased or decreased by a random small proportion (e.g. less than 1/3) of the current length of the corresponding interval. Possible illegal TIs are discarded.

Deterministic mutation uses the ranks of the TIs when selecting the genes to be changed. If the chromosome contains at least two TIs, the one with the lowest rank is deleted. Furthermore, with certain probability (e.g. 1/3), gradually better genes become subject to Random mutation (shrinking or enlargement). However, the best TI within the individual is never changed.

Mass mutation operates in a similar manner as Random mutation except that *all* nonzero genes are altered regardless of their ranks or number.

Selection Operator. The selection of the individuals to survive and reproduce in successive generations plays an important role in a GA [2]. *Stochastic selection* is performed on the basis of the fitness values of the individuals such that the best ones have higher chances to perpetuate. We use normalized geometric ranking in selection. The ranking is done by sorting the individuals in decreasing order of their fitness values and letting the rank be the index in the ordered list. The probability P_i of selecting individual i is defined as:

$$P_i = q_0(1-q)^{r_i - 1}, 1 \leq i \leq N\ , \quad (5)$$

where q is the probability of selecting the best individual (predefined, e.g. 0.2), r_i is the rank of individual i, N is the population size and $q_0 = q/[1-(1-q)^N]$.

3.4 Termination

The GA runs either up to a maximum number of iterations (*ItMax*) or until the classification accuracy of the best individual fails to improve during a specified number of iterations (*FailMax*).

3.5 The Structure of the GA

The GA takes as input the labeled data samples (the FFCs and their plant species membership) and outputs the optimal set of features that provides the highest classification accuracy. The solution is expressed as a set of TIs encoded in the best individual of the final population. The parameters of the GA are the population size N, the number of genetic operations (*bc* Best crossover, *opc* One-point crossover, *rm* Random mutation, *dm* Deterministic mutation and *mm* Mass mutation) that are performed at each iteration and the termination parameters (*ItMax* and *FailMax*).

Step 1: *Initialization*. An initial population of N individuals is generated at random. Let the initial population be the Current Population (CP).
Step 2: *Evaluation*.

2.a Rank the individual line segments (pairs of features) contained in the individuals in CP according to the estimated interclass separability;

2.b Estimate the classification accuracy, compute and assign the fitness value for all individuals in CP.
Step 3: Copy CP into the Mating Pool (MP).
Step 4: *Crossover*. Select at random $bc + opc$ pairs of parents from the CP;

Perform *bc* Best crossovers using the ranks of individual TIs;

Perform *opc* One-point crossovers.
Step 5: *Mutations*. Select at random $rm + dm + mm$ individuals from CP;

Perform *rm* Random, *dm* Deterministic and *mm* Mass mutations.
Step 6: Evaluate the new offspring as in Step 2 and add them to the MP.
Step 7: *Selection*. Allocate perpetuation probabilities to the individuals in the MP using their fitness values. Select N chromosomes and populate CP.
Step 8: *Termination*. If the convergence is not achieved return to Step 3.

4 Experimental Results

4.1 Parameter Settings

All subsequent trainings of NNs were performed using 40% of the data set for learning, 10% for validation and 50% for testing. These sets were initially chosen at random without replacement from the whole data set. Therefore, all chromosomes were evaluated using the same sets of samples. NNs were trained using the backpropagation method up to a maximum number of iterations (1000) or until the performance over the validation set did not improve during a predefined number of consecutive iterations (50). The accuracy of the NNs on the test set

is used for fitness computation (Eq.2,3). We let the GA operate on a population of N=30 individuals and a mating pool of 60. A predefined number of genetic operations were performed at each iteration: bc=5 Best crossovers, opc=2 One-point crossovers, rm=5 Random mutations, dm=7 Deterministic mutations and mm=4 Mass mutations. The GA ran up to $ItMax$=300 iterations and the convergence measure was set to $FailMax$=50 iterations of no improvement of the best fitness value. All computations were carried out with a 800 MHz Pentium computer equipped with 256 MB of RAM, by an original developed software, using Matlab 5.3 environment.

4.2 The Two Class Problem

The data set consisted of 400 fluorescence induction curves (FFCs) measured from 100 *maize* and 100 *tobacco* leaves with distinct curves from both sides of the leaves. The details of the acquisition of the FFCs are described in [19] (the 3-s experiment design). Among all six species in the data set we chose these particular ones because both are angiosperms and their FFCs show high similarity with each other and less similarity with gymnosperms or cryptogams (see Fig. 1B). Fig. 3 depicts 50 FFCs from each plant species, (A) maize and (B) tobacco. It turns out that the variation among the FFCs from the same plant species and the similarity between the two species make the identification by visual inspection very difficult, if not impossible. However, a rough pattern can be observed as the FFCs measured from maize leaves show a more rapid increase (first 200 time-points) and a steeper decrease (time-points 600-800) compared to the curves from tobacco leaves.

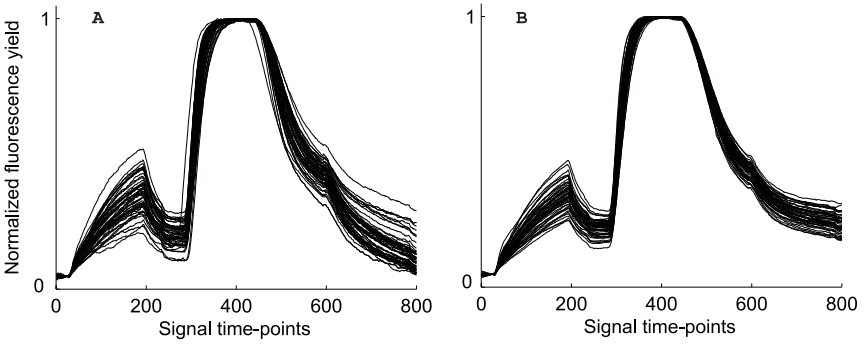


Fig. 3. (A) Fluorescence curves measured from 50 maize and (B) 50 tobacco leaves. Large intra-species variation occurs, yet rough differences can be observed at the beginning (first increasing phase) and at the end of the curves from the two plant species

The preliminary results using full-length FFCs were outstanding (100% classification accuracy, data not shown). We therefore investigated how the accuracy

deteriorates when using only the first half of the signal, that is, time-points up to 400, corresponding to 1.5 seconds. We repeatedly optimized (ten times) the feature set with the proposed GA and the results show that still acceptable identification accuracy is achieved. The recognition rate using optimized features was in the range 90 - 93.5% with an average of 92.2%. The average running time was 14.5 s per evaluation of a solution which means about 24 hours for a GA run.

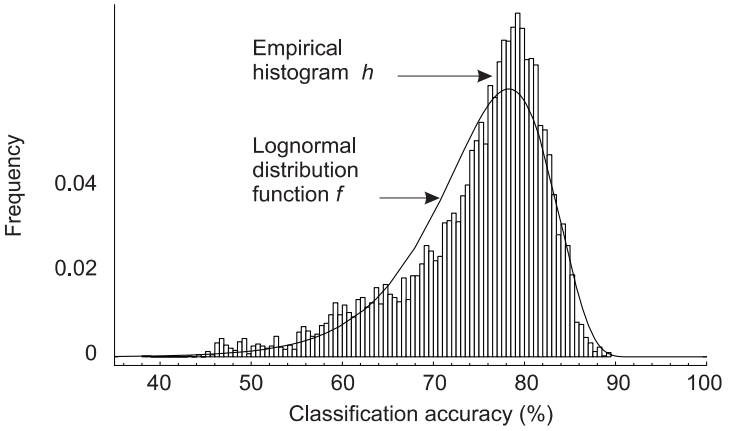


Fig. 4. Fit between the reversed lognormal distribution function f and the histogram h generated from the classification accuracies of 7932 random settings of the line segments. The random solutions had accuracies ranging from 38.25% to 89.75% with average of 74.68%. The GA gave maximum accuracy of 93.5%

A pure random search algorithm was compared to the GA in order to evaluate the power of the proposed method. This was done by generating a series of random TI settings and constructing a histogram of the classification accuracy when using these sets of TIs. This gives us an approximation for the performance of the pure random search (Fig. 4, empirical histogram h). We are interested in events of obtaining higher accuracy than a given value a. In statistical terms, the experiment consists of n independent Bernoulli trials with a constant probability p of success. If M is the number of trials needed for the first success, then the random variable M has a geometric distribution with parameter p. In particular, the mean number of random runs required for obtaining an accuracy larger than a is $1/p$. Next question concerns the estimation of the probability p. Since we lack the information about the upper bound on the classification accuracy, we assume that the accuracy of pure random search is distributed as a reversed lognormal random variable with a cut-off threshold of 100 (Fig. 4, distribution function f). Given this stochastic model, the probability p can be estimated as an integral $p = \int_a^\infty f(x)\mathrm{d}x$. In the present experiment with $n = 7932$ and a=93.5 the model gives p= 6.5×10^{-13}. Hence, $1/p = 1.5 \times 10^{12}$ random runs would be needed for the first occurrence of a result better than obtained with the GA. Typically, 200 iterations proved to be sufficient for the GA to converge

for N=30 individuals and a mating pool of 60. The equivalent effort of random search computation is 6000 evaluations. The estimated number of random runs needed for obtaining a solution that outperforms the GA is of the order of 10^{12} and therefore random search is not practical.

4.3 The Crop-Weeds Problem

An automatic herbicide sprayer is an example of a possible practical application of a plant identification system. Such a system needs only to detect the cultivated plant among various others, regardless of their diversity. In order to simulate the distinction between crop and weeds we constructed an artificial class ('Mixed') by grouping together 50 FFCs measured from four different plant species (nettle, rye, haircap moss and pine). We selected maize as the cultivated plant and carried out similar feature optimization experiments as in the two-class problem.

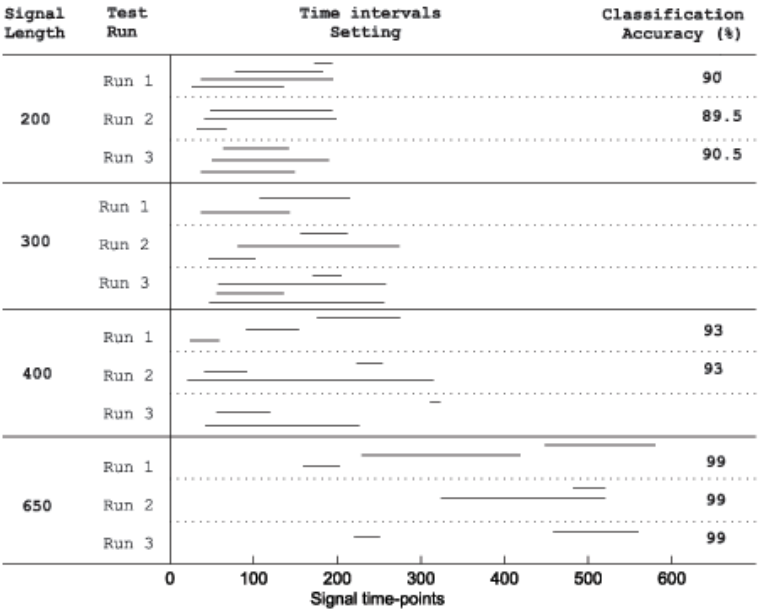


Fig. 5. The effect of shortening the length of the input signal on the classification accuracy between maize and Mixed (nettle, rye, haircap moss and pine). Four shortening steps were considered: to 650, 400, 300 and 200 time points. The best individuals in the final populations from three replicate runs of the GA are illustrated in each case

Signal shortening caused no severe deterioration of the classification accuracy (Fig. 5). Because the FFC describes the time-course of fluorescence yield, measured under a complex, sequential illumination pattern, shortening is possible only from the end of the curves. As an example, shortening from 400 to 300 time points forces the optimization process to detect the differences among FFCs

excluding the phase of rapid increase in fluorescence yield. Similarly, reducing the input signal length to 200 time points discards another phase (corresponding to the illumination pattern) where the curves show different patterns of the decreasing intensity (Fig. 2). Interestingly, the number of features discovered by our method was much lower (two to four TIs) than that chosen manually (eight TIs) [9], [19]. A reduced number of features increases the speed of identification without loosing the power of the fingerprinting method. However, in earlier studies the overall classification was carried out using several classes.

4.4 Application to Unseen Data

In order to test the generalization power of the optimized set of features, classification was tested on distinct data sets. When applied to unseen data using the first 400 data points, NNs achieved 85-90% recognition rates both in the two-class and crop-weeds classification tasks (results not shown). This is slightly less than the 92-93% accuracy obtained with the original test set (Fig. 5).

5 Conclusion and Discussion

We considered an approach to feature analysis, called feature learning, and designed a suitable evolutionary framework. Its effectiveness was highlighted in the context of plant fluorescence fingerprinting. The proposed framework may also be useful in other applications where the model of the feature extraction procedure is known and there is a large amount of irrelevant data. Illustrative examples could be automatic detection of the optimal number of eigenvectors required by principal component analysis; the level of decomposition of signals using (e.g., wavelet) transforms and selection of the coefficients; the number and size of patches for automatic image segmentation.

At the early stage of the system development we tested several fitness functions that include size penalty factors [1],[22]. The most promising of these was chosen and the results confirmed its performance. By analyzing the behavior of the genetic operations, we observed the efficiency of the Best crossover operator which methodically considers the best combination of genes from two parents. A simple local search mechanism, simulated by the mass mutation operator, proved to be extremely efficient, especially at the late iterations when the diversity of the population diminishes.

Since the main purpose of the present study was to design a general framework for the proposed feature learning method, application of the optimized features on unseen data was tested only shortly. These tests show that the optimized features perform almost as well with unseen data as with the original test data.

NNs were selected for classification because of their previous success in plant fingerprinting. It would be possible to employ some other classification technique either in the optimization step or in the final recognition system. In the first case,

some computation time might be saved by using a simpler classifier in the optimization process. However, the optimization is performed off-line and therefore the time of the execution is not critical but the performance of the selected features is crucial. That is why we used the same classifier both in the optimization and in the final recognizer and omitted further investigation of other classifiers.

There are many variants of feature analysis methods that combine partial results (e.g. feature weighting [11]) or embed the selection of features in other tasks like learning algorithms - pruning input nodes of NNs or, more generally, evolutionary NN design [21]. However, in this context, the main objectives are the optimization of the NN architecture, weights and/or learning parameters.

The fluorescence fingerprinting method is being developed for use in precision agriculture [9],[19]. The final aim is to equip herbicide sprayers and other agricultural devices with the ability to automatically distinguish cultivated plants from weeds when the device travels on a field. The development of the method is currently in the phase of laboratory experiments tuning the illumination protocol and the recognition algorithms. The method proposed here serves as an automatic tool for discovering the most relevant features from FFCs. In the present paper, we also studied how to reduce the length of the signal because speed would be a main issue in real-life precision farming applications.

References

1. Bandyopadhyay, S.: Pattern classification using genetic algorithms: Determination of H. Pattern Recognition Letters, **19** (1998) 1171-1181
2. Blickle, T., and Thiele, L.: A comparison of selection schemes used in Evolutionary Algorithms. Evolutionary Computation 4, MIT-Press (1996) 361-394
3. Chen, K-Z., Zhang, X-W., Ou, Z-Y., Feng, X-A.: Recognition of digital curves scanned from paper drawings using genetic algorithms. Pattern Recognition, **36** (2003) 123-130
4. Codrea, M.,C., Aittokallio, T., Keränen, M., Tyystjärvi, E., Nevalainen O.,S.: Feature learning with a genetic algorithm for fluorescence fingerprinting of plant species. Pattern Recognition Letters, **24** (2003) 2663-2673
5. Dash, M., Liu, H.: Feature Selection for Classification. Intelligent Data Analysis, **1** (1997) 131-156
6. Fukunaga, K.: Introduction to Statistical Pattern Recognition. 2nd edn., Academic Press, San Diego, California (1990)
7. Govindjee: Sixty-three years since Kautsky: Chlorophyll *a* fluorescence. Australian Journal of Plant Physiology, **22** (1995) 131-160
8. Ho, S-Y., Chen, Y-C.: An efficient evolutionary algorithm for accurate polygonal approximation. Pattern Recognition, **34** (2001) 2305-2317
9. Keränen, M., Aro, E.-M., Tyystjärvi, E., Nevalainen, O., S.: Automatic plant identification with chlorophyll fluorescence fingerprinting. Precision Agriculture, **4** (2003) 53-67
10. Kohavi, R., John, G., H.: Wrappers for feature subset selection. Artificial Intelligence, **97** (1997) 273-324
11. Komosiński M., Krawiec, K.: Evolutionary weighting of image features for diagnosing of CNS tumors. Artificial Intelligence in Medicine, **19** (2000) 25-38

12. Lazár, D.: Chlorophyll *a* fluorescence induction. Biochimica et Biophysica Acta, **1412** (1999) 1-28
13. Mao, J., Jain, A.,K.: Artificial neural networks for feature extraction and multivariate data projection. IEEE Trans. Neural Networks, **6** (1995) 296-317
14. Maxwell, K., Johnson G.,N.: Chlorophyll fluorescence - a practical guide. Journal of Experimental Botany, **51** (2000) 659-668
15. Nilsson, N., J.: Artificial Intelligence: A new Synthesis. Morgan Kaufmann Publishers Inc. (1998)
16. Piater, J., H, Piater, J., H., Grupen, R., A.: Feature Learning for Recognition with Bayesian Networks. Proceedings of International Conference on Pattern Recognition (2000) 1017-1020
17. Siedlecki, W., Slansky, J.: A note on genetic algorithms for large-scale feature selection. Pattern Recognition Letters, **10** (1989) 335-347
18. Späth, H.: Cluster analysis algorithms for data reduction and classification of objects. UK: Ellis Horwood Ltd. (1980)
19. Tyystjärvi E., Koski A., Keränen M., Nevalainen O.,S.: The Kautsky curve is a built-in barcode. Biophysical Journal, **77** (1999) 1159-1167
20. Questier, F., Walczak, B., Massart, D., L., Boucon, C., de Jong, S.: Feature selection for hierarchical clustering. Analytica Chimica Acta, **466** (2002) 311-324
21. Yao, X.: A review of evolutionary artificial neural networks. International Journal of Intelligent Systems, **8** (1993) 539-567
22. Yu, S., De Backer, S., Scheunders, P.: Genetic feature selection combined with composite fuzzy nearest neighbor classifiers for hyperspectral satellite imagery. Pattern Recognition Letters, **23** (2002) 183-190

ROC-Based Evolutionary Learning: Application to Medical Data Mining

Michèle Sebag, Jérôme Azé, and Noël Lucas

TAO : Thème Apprentissage et Optimisation
Laboratoire de Recherche en Informatique, CNRS UMR 8623
Université Paris-Sud Orsay, 91405 Orsay Cedex
{sebag,aze,lucas}@lri.fr

Abstract. A novel way of comparing supervised learning algorithms has been introduced since the late 90's, based on Receiver Operating Characteristics (ROC) curves.
From this approach is derived a NP complete optimization criterion for supervised learning, the area under the ROC curve.
This optimization criterion, tackled with evolution strategies, is experimentally compared to the structural risk criterion tackled by quadratic optimization in Support Vector Machines. Comparable results are obtained on a set of benchmark problems in the Irvine repository, within a fraction of the SVM computational cost.
Additionally, the variety of solutions provided by evolutionary computation can be exploited for visually inspecting the contributing factors of the phenomenon under study. The impact study and sensitivity analysis facilities offered by *ROGER* (ROC-based Genetic LearneR) are demonstrated on a medical application, the identification of Atherosclerosis Risk Factors.

1 Introduction

Supervised machine learning (ML) is interested in estimating a nominal or numerical variable based on some set of labeled examples, or training set.

The learning performance is usually measured from the predictive accuracy of the estimator or hypothesis, i.e. the percentage of correctly identified labels in another set of examples, the test set [Die98].

Though predictive accuracy was commonly used to compare learning algorithms, it suffers from several shortcomings regarding skewed example distributions (e.g. discriminating a 1% positive examples from 99% negative examples) and asymmetric mis-classification costs [Dom99].

A remedy to these limitations was offered by Receiver Operating Characteristics (ROC) analysis [Bra97,PFK98,LHZ03], as will be detailed in the next Section. ROC curves, originated from the signal theory, are popular in Medical Data Analysis as they offer a synthetic representation of the trade-off between the true positive rate (TP) and the false positive rate (FP) depending on how a medical test is interpreted, e.g. which thresholds are used to tell pathological from normal cases (Fig. 1).

P. Liardet et al. (Eds.): EA 2003, LNCS 2936, pp. 384–396, 2004.

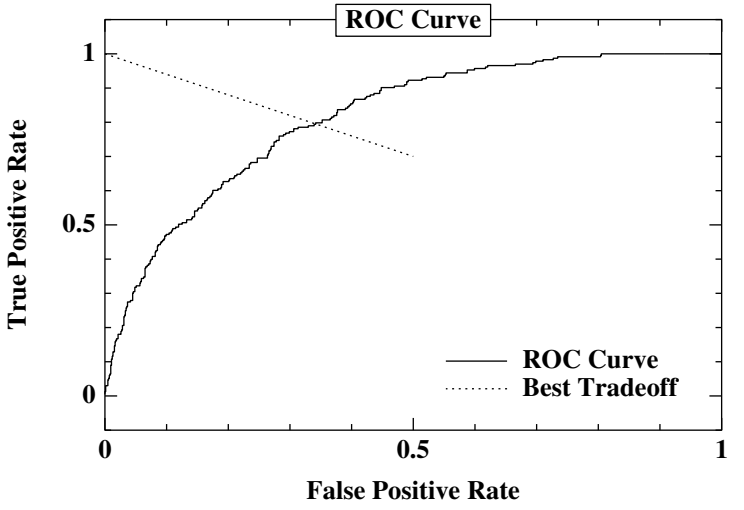


Fig. 1. The ROC curve illustrates the hypothesis trade-off between true and false positive rates. The best performance for a given misclassification cost ratio is found on the Best Trade-off line.

Along these lines, it came naturally to consider the area under the ROC curve (AUC) as a learning criterion [MDC$^+$01,FFHO02]. As this criterion induces a mixed, NP complete optimization problem [CSS98], its optimization was tackled within greedy search [FFHO02] or genetic algorithms [MDC$^+$01].

Independently, the foundational approach of Statistical Learning offered a variety of learning optimization criteria, drastically renewing the Machine Learning perspective [Vap98,SBS98,CST00]. Such criteria derive well-posed optimization problems, such that the optimum solution offers statistical guarantees of learning performance; for instance, Support Vector Machines (SVMs) are determined by quadratic minimization of the structural risk criterion.

The present paper presents the *ROGER* (*ROC-based Genetic Learning*) algorithm, implementing the evolution-strategy based optimization of the AUC criterion; *ROGER* is compared to a state-of-art SVM algorithm known as *SVMTorch* [CB01].

Both algorithms demonstrate comparable learning performance on a subset of benchmark problems in the Irvine repository [BKM98]; however, *ROGER* requires a fraction of *SVMTorch* computational effort. The differences in the learning behavior of both algorithms are discussed and some interpretations are proposed.

Finally, this paper presents a novel exploitation of the variety of hypotheses provided by evolutionary optimization, through impact study and sensitivity analysis visual facilities. As noted by [CMS99], visual representations can provide the expert with a wealth of easy-to-understand and yet precise information. These visual facilities are illustrated and discussed on a medical data mining

application, the identification of Atherosclerosis Risks, presented in the PKDD 2002 Challenge [LAS02].

This paper is organized as follows. Section 2 presents ROC analysis and reviews related work. Section 3 describes the *ROGER* algorithm. Section 4 reports on comparative experiments, using *SVMTorch* as reference algorithm. Section 5 finally describes and discusses how to exploit the variety of evolution-based solutions in a Visual Data Mining perspective.

2 ROC Curves and Machine Learning

This section briefly situates the use of ROC curves from a machine learning perspective.

2.1 Robustness of ROC Curves

As mentioned in the introduction, predictive accuracy might be a poor performance indicator when the problem domain suffers from skewed class distributions and involve asymmetric mis-classification costs. For instance, medical or text retrieval applications commonly present negative examples outnumbering positive examples by a factor 100, with a false positive cost (mistaking a negative example for a positive one) usually much lower than the false negative cost (missing a true positive).

Specific heuristics are devised to resist such characteristics, e.g. through oversampling the rare class, incorporating the mis-classification costs in the learning criteria, and/or relabeling the examples [Dom99].

An alternative would be to see learning as a multi-objective optimization problem (see [Deb01] and references therein), simultaneously maximizing the true positive and true negative rates. From this perspective, ROC curves simply depict the Pareto front associated to a set of hypotheses and/or learners (Fig. 1) [Bra97,PFK98,SKB99].

Three particular points in the ROC space correspond to the *All positive* hypothesis (point $(1, 1)$), *All negative* hypothesis (point $(0, 0)$), and discriminant hypotheses (point $O = (1, 0)$).

By construction this representation does not depend on the class distribution, since it uses normalized coordinates, the true positive and false positive rates.

Furthermore, this representation immediately allows one to select the best hypothesis depending on the error costs. Assuming that a false negative error costs r times more than a false positive one, the best hypothesis lies at the intersection of the ROC curve with the straight line Δ of slope $\frac{-1}{r}$ (Fig. 1).

2.2 Related Work

As argued in [Bra97,MDC+01,FWB+98], ROC curves also allow one to deal with practical requirements on the minimal TP or maximal FP rates. For instance, the detection of churners within a given sensitivity (TP) and precision (1-FP) range

was achieved in [MDC+01], with a GA-based optimization of the area under the ROC curve in the desired ranges. The detection of malignant mammograms with a minimum sensitivity and precision was similarly tackled by EP-based optimization of ANN and linear hypotheses in [FWB+98].

When such desired ranges are unknown, the whole area under the ROC curve can be taken as learning criterion (AUC), with two warnings. Firstly, the AUC criterion is not "better" than the predictive accuracy [LHZ03]; rather, both criteria define distinct optimization landscapes. Secondly, the AUC criterion defines an NP complete, combinatorial optimization problem; to our best knowledge, this optimization problem was only addressed through greedy or evolutionary search, respectively learning decision trees [FFHO02], or linear hypotheses [MDC+01].

Nevertheless, optimizing the AUC criterion enforces the learning stability with respect to the misclassification costs. Learning stability is most generally desirable, for the target hypothesis should be independent as much as possible from fortuitous effects in the problem domain. To our best knowledge, learning stability has mostly been investigated in relation with the training example distribution [BE02]. But stability wrt misclassification costs is desirable as well, for two reasons. On one hand, the expert usually sets the misclassification costs by trials and errors; optimizing the ROC curve provides optimal hypotheses for various misclassification costs, which allows the expert for a more informed and better choice.

On the other hand, the appropriate misclassification costs might vary, depending on additional information on the case at hand (e.g. the "normal" range for a bio-chemical exam might depend on the age and mobility of a patient). The decision making based on a ROC curve can thus be locally adjusted depending on the case at hand.

Conversely, the use of ROC spaces offers a geometrical and intuitive representation for the behavior and dynamics of a learning strategy on a given domain [Fla03]; for instance, experimentations with different learning criteria (m-estimate, Gini criterion, entropy) offer new insights into how they trade-off the FP and TP rates [FF03].

Less related to ROC analysis, the *Learning to Order Things* approach developed by Cohen et al. [CSS98] searches for ranking hypotheses, compatible with the preferences of some experts in a Web-based and text retrieval context. Indeed, any ordering hypothesis that would rank positive examples first, would reach the optimum of the AUC criterion.

3 Genetic ROC-Based Learning

This section describes the *ROGER* algorithm, implementing an evolution-strategy based optimization of the AUC criterion, and discusses its limitations.

3.1 Overview

Let the data set be given as $\mathcal{E} = \{(x_i, y_i), i = 1 \ldots n,\ x_i \in X, y_i \in \{1, -1\}\}$, where X denotes the instance space. In the following, we restrict ourselves to attribute-value learning, e.g. $X = \mathbb{R}^d$.

The hypothesis space considered in the following is the set of real-valued functions on X.

For the sake of comparison, only linear functions are considered in the following; the hypothesis space $\mathcal{H}$ is set to $\mathbb{R}^d$. The extension to richer function spaces using kernel-based representations is planned for further research.

Any real-valued hypothesis h on X induces by thresholding a family of binary classifiers $\{h_t,\ t \in \mathbb{R}\}$, with

$$h_t(x) = \begin{cases} 1 & \text{if } h(x) > t \\ -1 & \text{otherwise} \end{cases}$$

It is straightforward to see that the true positive and the false positive rates monotonically increase as t decreases: the curve defined by $(FP(h_t), TP(h_t), t \in \mathbb{R})$ is a ROC curve.

The fitness of h is set to the area under the above ROC curve, computed with complexity $n \log n$ where n is the number of examples (Table 1). Normalization is omitted as of no effect on the optimization problem.

Table 1. Fitness of h = Area Under the Roc Curve of h

Fitness function of hypothesis h

Input
  Data set $\mathcal{E} = \{(x_i, y_i), i = 1 \ldots n,\ x_i \in X, y_i \in \{1, -1\}\}$
  Hypothesis $h : X \mapsto \mathbb{R}$
Init
  Sort $\mathcal{E} = \{(x_i, y_i)\}$ by decreasing order, where $i > j$
    iff $(h(x_i) > h(x_j))$ or $((h(x_i) = h(x_j)$ and $(y_i > y_j))$.
  $p = 0$
  $\mathcal{F} = 0$
For $i = 1$ to n
  if $y_i = 1$, increment p;
  else $\mathcal{F} = \mathcal{F} + p$
EndFor
Return $\mathcal{F}$

The optimization of fitness function $\mathcal{F}$ on the search space $\mathcal{H} = \mathbb{R}^p$ is achieved using evolution strategies (ES) with self-adaptive mutation [Sch81], well suited to parametric optimization [Bäc95].

We use the $(\mu+\lambda)-ES$ selection/replacement mechanism; μ parents generate λ offspring, and the best individuals among the μ parents + λ offspring are selected as parents for the next generation.

3.2 Discussion and Limitations

Since the 90's, the use of real-valued hypotheses has been investigated in two major areas of machine learning, namely statistical learning [Vap98] and ensemble learning [SFPL97]. The efficiency of both approaches is explained from the optimization of the minimal margin (the diagnostic confidence, or distance to the discrimination threshold t).

Based on structural risk minimization, SVMs actually depend on a few selected examples, the support vectors; they achieve stable learning as they pay no attention to (the distribution of) other examples.

The difference with AUC optimization is twofold. On one hand, the AUC criterion depends on the whole example distribution. On the other hand, AUC is an order-based criterion, reputed more stable under statistical noise than real-valued criteria. In counterpart, AUC maximization defines an ill-behaved optimization landscape, as it maps a continuous search space onto a finite integer set, while structural risk defines a convex, quadratic optimization landscape.

However, AUC minimization achieves learning stability wrt misclassification cost, as desired and discussed in Sect. 2; a single real-valued hypothesis h is learned, and the misclassification cost only governs the discrimination threshold t. In opposition, modifying the misclassification cost would require to retrain SVMs and result in a different hypothesis.

In conclusion, AUC-based learning presents two main drawbacks. One, shared with SVMs, is that it does not provide an intelligible hypothesis, though experts are often willing to sacrifice some accuracy for more intelligible hypotheses. The other drawback results from the fact that AUC defines an under-specified optimization problem, admitting infinitely many solutions.

We shall see in Sect. 5 that this multiplicity of solutions can be exploited and provide the expert with some facilities for inspecting the hypotheses, "for free".

4 Comparative Validation

ROGER is experimentally validated on eight problems from the Irvine repository [BKM98], and compared to a state-of-art SVM, *SVMTorch* [CB01].

4.1 Experimental Setting

On each problem, the dataset is splitted into a training set and test set with same class distribution as the global dataset, and 11 independent splits are considered. The split is done with 2/3 of the data in the training set, and 1/3 in the test set; in some cases (see Table 3), the size of the training set has been reduced such that *SVMTorch* learning computational cost be less than 15 minutes on Pentium-II, 400 MHz.

On each training set, *SVMTorch* is run with default parameters, and computes the SRM-optimal hyperplane on the training set, $h(\boldsymbol{x}) = \boldsymbol{w}.\boldsymbol{x} + b$. Hypothesis h is assessed from the ROC curve associated to $\boldsymbol{w}.\boldsymbol{x}$ on the test set.

On the same training set, *ROGER* is run 21 times, providing 21 independent solutions h to the AUC maximization problem (Table 1). Same parameters, summarized in Table 2, are used for all runs and all problems.

For each training set, we consider the median of the ROC curves on the test set, over the 21 runs. As already noted by [PFK98], the representativity of the median ROC curve is difficult to assess since different portions of the curve correspond to distinct hypotheses.

Finally, we take the mean of the above medians over the 11 splits for each dataset.

Table 2. *ROGER* parameters

population size	# parents μ	10
	# offspring λ	50
max nb evaluations		10,000
crossover	uniform	rate .6
mutation	self-adaptive	rate 1

Table 3. The AUC values and computational time of *ROGER* and *SVMTorch* on eight datasets from Irvine Repository

	nb att	nb att lin	#Train	#Test	*ROGER* AUC	*ROGER* time	*SVMTorch* AUC	*SVMTorch* time
Breast Cancer	9	42	189	97	.674 ± .05	7”	.672 ± .05	1”
Crx	15	47	70	620	.816 ± .06	7”	.839 ± .04	886”
German	25	25	100	900	.712 ± .03	6”	.690 ± .02	96”
Promoters	59	229	70	36	.863 ± .07	2”	.974 ± .02	< 1”
Satimage	36	36	139	1237	.918 ± .01	4”	.876 ± .02	14”
Vehicle	18	18	125	291	.994 ± .005	1”	.993 ± .007	< 1”
Votes	16	32	287	148	.993 ± .004	7”	.989 ± .005	$> 1{,}000$
Waveform 1-2	22	22	211	3321	.971 ± .004	4”	.963 ± .008	2”

The experiment goal is to compare AUC-based learning with SVMs in terms of predictive accuracy and learning stability. At the moment, only linear hypotheses are considered; *SVMTorch* is run with a linear kernel.

4.2 Experiments

Table 3 summarizes the datasets considered, the size of the training and test sets, the initial number of attributes and the size of the hypothesis search space, being reminded that a nominal attribute with k modalities is expressed as k boolean attributes, and accounts for k coefficients in the linear hypothesis h. As already mentioned, the size of the training set was limited to restrict the computational cost of *SVMTorch* to a maximum of 15 minutes on Pentium II, 400 MHz. The computational cost of *ROGER* is lower by one or several orders of magnitude than *SVMTorch* in the worst cases.

The average and standard deviation of the areas under the ROC curve, averaged over 11 runs for *SVMTorch* and 11×21 runs for *ROGER*, are reported in Table 3. Similar AUCs values are obtained. Likewise, *ROGER* and *SVMTorch*

have similar predictive accuracies, as can be observed from Fig. 2. *SVMTorch* significantly outperforms *ROGER* on Promoters; *ROGER* significantly outperforms *SVMTorch* on Satimage. In most other cases, the median curves are almost indistinguishable, except sometimes in the beginning of the curve.

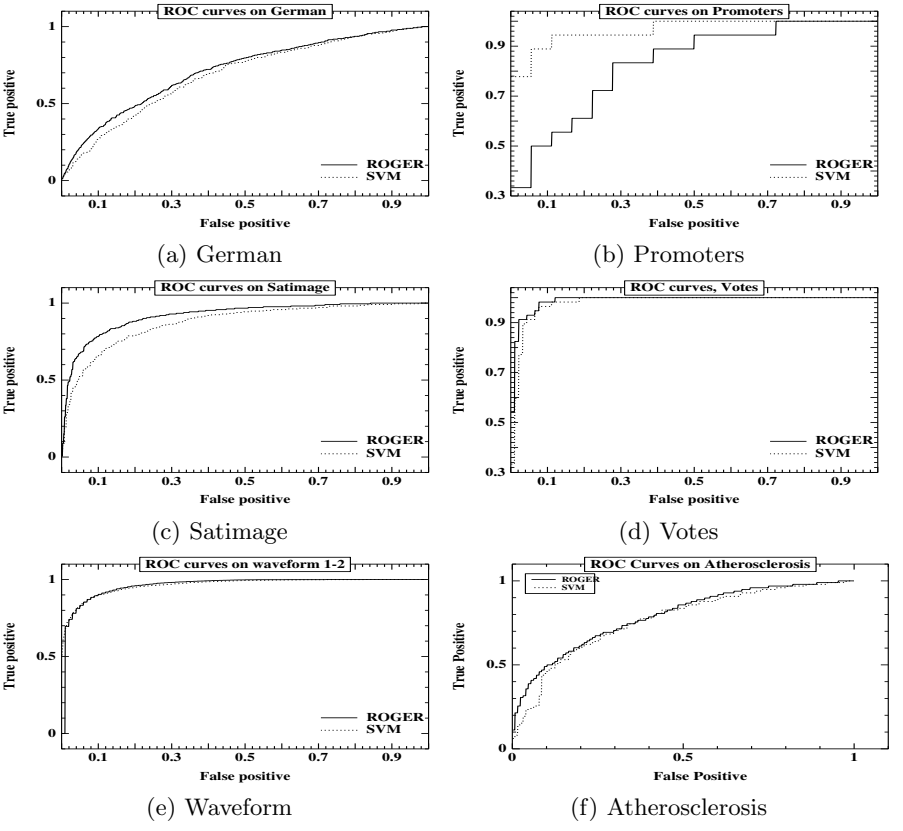


Fig. 2. Comparison of the median ROC curves obtained with *SVMTorch* and *ROGER*

These experiments suggest that AUC maximization is competitive with respect to *SVMTorch*, which is much encouraging given the maturity and the strong mathematical foundations of SVMs. A fortiori, *ROGER* compare favorably to more traditional learners such as C4.5, naive Bayes, and k-NN on these same problems, after [PFK98].

The scalability of the approach with respect to the size of the dataset, in $n \log n$, is quite satisfactory. Empirically, the computational cost of AUC evolutionary minimization is much lower on average than for *SVMTorch*, with a very low standard deviation.

However, the AUC scalability with respect to the number of attributes is questionable. The worst performances are obtained for the Promoters problem,

with 229 attributes. On-going experiments are underway to investigate and understand this limitation.

5 Impact Studies and Sensitivity Analysis

This section shows how the multiplicity of solutions provided by AUC evolutionary optimization can be exploited by the expert to gain some insights into the phenomenon at hand. The approach is illustrated on the PKDD 2002 Challenge dataset[1], concerned with the identification of risk factors for atherosclerosis and cardio-vascular diseases (CVD).

5.1 The Data and Learning Goal

Two databases have been made publicly available for the PKDD2002 Challenge. The Entry database describes the personal and family case for 1419 middle aged men.

This database involves 219 attributes, which have been manually compressed into 28 boolean and numerical attributes [LAS02].

The Control database presents the longitudinal study over 20 years of a sample of men. Using the medical expertise of the third author, this sample was divided into three classes, depending whether their health after 20 years is good, bad, or other (the later class includes in particular all men who disappeared from the study)[2].

5.2 Experimental Setting

The goal is to predict from the individual description given in the Entry database, his health state after twenty years.

The dataset is splitted into a 2/3 training set and 1/3 test set with same class distribution as the global dataset, and 11 independent splits are considered.

On each training set, *SVMTorch* is run with default parameters, and the optimal hypotheses are again evaluated from their median ROC curve.

On each training set, 21 independent *ROGER* runs are launched with parameters given in Table 2). The median ROC curves over 21 runs are averaged over the 11 splits of the data, and the mean ROC curve is displayed in Fig. 2.(f).

ROGER shows good performances, with an average AUC of .79 $\pm$.012 to be compared with .76 $\pm$.045 for *SVMTorch*.

Interestingly, the main difference between the two curves occurs close to the origin. It appears that some negative examples are classified as positive with high confidence by *SVMTorch*. Indeed, SVMs make no difference between misclassified examples provided that their confidence is above the cost threshold; and one would not increase the cost threshold too much, as this would increase

[1] http://lisp.vse.cz/challenge/ecmlpkdd2002/

[2] The prepared dataset is available at `http://www.lri.fr/~aze/PKDD2002/`.

the sensitivity to noise of the algorithm. In contrast, the AUC criterion offers a finer-grained evaluation of mis classifications, as the cost of an error actually depends on its rank; improving the example order, even in extreme regions, is rewarded. Accordingly, the *True Positive* rate increases abruptly at the beginning of the *ROGER* curve: individuals classified as the most at risk are on average in bad shape. In medical terms, the sensibility of the *ROGER* hypothesis is better compared to *SVMTorch* on this problem.

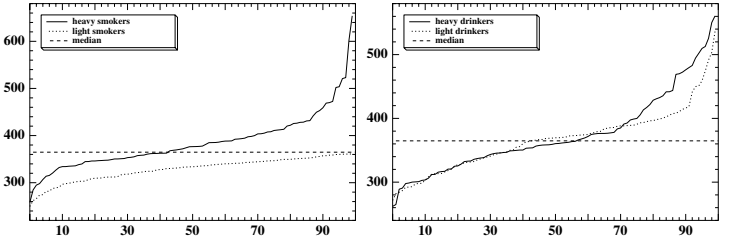
5.3 Impact Studies

A well known limitation of SVMs (also incurred by *ROGER*) is that it does not provide an easy-to-read hypothesis. An alternative to the analytic inspection of hypotheses is offered by diagrammatic representations, as investigated in Visual Data Mining [CMS99]. Along these lines, we explore some graphical interpretations of the *ROGER* hypotheses.

A first graphical exploitation concerns the impact study, analyzing the contribution of a given feature on the concept under examination; classically, this contribution is measured using correlation, chi-square or entropy.

However, *ROGER* hypotheses (and more generally, any ordered hypothesis) provide a more detailed, intuitive and yet precise picture, about the contribution of a feature (attribute, function of attributes). As an example, let us investigate the impacts of the tobacco and alcohol intoxication on atherosclerosis risk factors.

These impacts are graphically assessed, using the following protocol.



(a) Tobacco (b) Alcohol

Fig. 3. Tobacco and Alcohol Impacts on Atherosclerosis Risks

For each feature (here, an attribute), the 10% individuals in the test set with maximal (resp. minimal) values for this attribute, are considered. In both subsets, the individuals are ranked by increasing value of h, and the curves $(i, h(x_i))$ are displayed.

Each curve shows globally the risk range for the individuals with high (resp. low) intoxication (though the risk might be due to other factors, correlated with the intoxication). It is believed that such curves convey a lot more information than the correlation factor or quantity of information.

Furthermore, they allow for an intuitive comparison of the factors, by superposing the curves. For instance, the impact of tobacco can be argued from the fact that the non-smoking individuals all lie in the better half of the population (their risk is less than the median risk). The heavy smoker risk is always higher than for non-smokers; 2/3 of the heavy smokers show an above-average risk and the risk rises sharply for the worst 20% of the heavy smokers.

The apparently lesser impact of alcohol must be taken with care. On one hand, it is true that a small amount of red wine was found beneficial against some CVD. On the other hand, it appeared from the data that the men considered "light drinkers"... were not drinking so lightly.

5.4 Sensitivity Analysis

The multiplicity of optimal solutions for the AUC criterion and/or the variability of stochastic optimization, can also be exploited for sensitivity analysis.

Let us represent a model h as a curve i, w_i, where i stands for the index of the attribute and w_i is the associated weight. Fig. 4 displays 21 models learned from the total dataset, showing that some attributes play a major role for the target concept (typically the tobacco factor, attribute 9). Conversely, some other attributes can be considered as weakly relevant at best. Last, the inspection of the curves suggests that some attributes might be inversely correlated, hinting at the creation of compound attributes.

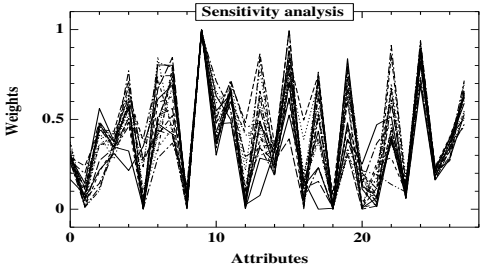


Fig. 4. Sensitivity analysis

6 Conclusion and Perspectives

This paper investigates a recent learning criterion, the maximization of the area under the ROC curve. A simple ES-based maximization of this criterion appears to be competitive with well-founded statistical learning algorithms, SVMs [Vap98].

The real-valued nature of the hypotheses allows for visual impact studies, assessing the contribution of any attribute to the concept at hand; as shown in Sect. 5, such visual representations provide much richer information than a correlation factor.

Moreover, the intrinsic variability of evolutionary results can be exploited to provide "for free" a sensitivity analysis.

Further research is concerned with extending *ROGER* to more complex instance and hypothesis languages, using for instance kernel representations. In parallel, the sensitivity analysis will be exploited for feature selection and construction.

Acknowledgment. Our thanks go to Dr Maria Temeckova and R. Collobert, for providing very valuable free data and algorithmic resources.

We also warmly thank M.-C. Jaulent, Dr. I. Colombet, Dr. F. Gueyffier and Pr. G. Chatellier, for strong multidisciplinary interactions.

References

[Bäc95] T. Bäck. *Evolutionary Algorithms in theory and practice*. New-York:Oxford University Press, 1995.

[BE02] O. Bousquet and A. Elisseeff. Stability and generalization. *Journal of Machine Learning Research*, 2:499–526, 2002.

[BKM98] C. Blake, E. Keogh, and C.J. Merz. *UCI Repository of machine learning databases*. http://www.ics.uci.edu/∼mlearn/MLRepository.html, 1998.

[Bra97] A.P. Bradley. The use of the area under the ROC curve in the evaluation of machine learning algorithms. *Pattern Recognition*, 1997.

[CB01] R. Collobert and S. Bengio. Svmtorch: Support vector machines for large-scale regression problems. *Journal of Machine Learning Research*, 1:143–160, 2001.

[CMS99] S. K. Card, J. D. Mackinlay, and B. Shneiderman. *Information Visualization: Using vision to think*. Morgan Kaufmann, 1999.

[CSS98] W. W. Cohen, R. E. Schapire, and Y. Singer. Learning to order things. In *Advances in Neural Information Processing Systems*, volume 10. The MIT Press, 1998.

[CST00] N. Cristianini and J. Shawe-Taylor. *An introduction to Support Vector Machines*. Cambridge University Press, 2000.

[Deb01] K. Deb. *Multi-Objective Optimization Using Evolutionary Algorithms*. John Wiley, 2001.

[Die98] T.G. Dietterich. Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Computation*, 1998.

[Dom99] P. Domingos. Meta-cost: A general method for making classifiers cost sensitive. In *Knowledge Discovery from Databases*, pages 155–164. Morgan Kaufmann, 1999.

[FF03] J. Furnkranz and P. Flach. An analysis of rule evaluation metrics. In *Proc. of the 20^{th} Int. Conf. on Machine Learning*. Morgan Kaufmann, 2003.

[FFHO02] C. Ferri, P. A. Flach, and J. Hernandez-Orallo. Learning decision trees using the area under the ROC curve. In Morgan Kaufmann, editor, *Proceedings of the 19^{th} International Conference on Machine Learning*, pages 179–186, 2002.

[Fla03] P. Flach. The geometry of ROC space: Understanding machine learning metrics through ROC isometrics. In *Proc. of the 20^{th} Int. Conf. on Machine Learning*. Morgan Kaufmann, 2003.

[FWB+98] D.B. Fogel, E.C. Wasson, E.M. Boughton, V.W. Porto, and P.J. Angeline. Linear and neural models for classifying breast cancer. *IEEE Trans. Medical Imaging*, 17:3:485–488, 1998.

[LAS02] N. Lucas, J. Azé, and M. Sebag. Atherosclerosis risk identification and visual analysis. In *Discovery Challenge ECML-PKDD 2002*. http://lisp.vse.cz/challenge/ecmlpkdd2002/, 2002.

[LHZ03] C.X. Ling, J. Hunag, and H. Zhang. AUC: a better measure than accuracy in comparing learning algorithms. In *Proc. of 16th Canadian Conference on AI 2003*, 2003. to appear.

[MDC+01] M.C. Mozer, R. Dodier, M. C. Colagrosso, C. Guerra-Salcedo, and R. Wolniewicz. Prodding the ROC curve: Constrained optimization of classifier performance. In *Advances in Neural Information Processing Systems*, volume 13. The MIT Press, 2001.

[PFK98] F. Provost, T. Fawcett, and R. Kohavi. The case against accuracy estimation for comparing classifiers. In *Proc. of the 15^{th} Int. Conf. on Machine Learning*, pages 445–553. Morgan Kaufmann, 1998.

[SBS98] B. Schölkopf, C. Burgess, and A. Smola. *Advances in Kernel Methods*. MIT Press, 1998.

[Sch81] H.-P. Schwefel. *Numerical Optimization of Computer Models*. John Wiley & Sons, New-York, 1981. 1995 – 2^{nd} edition.

[SFPL97] R.E. Shapire, Y. Freund, P.Bartlett, and W.S. Lee. Boosting the margin: a new explanation for the effectiveness of voting methods. In *Proc. of the 14^{th} Int. Conf. on Machine Learning*, pages 322–330. Morgan Kaufmann, 1997.

[SKB99] A. Srinivasan, R.D. King, and D.W. Bristol. An assessment of submissions made to the Predictive Toxicology Evaluation Challenge. In *Proc. of Int. Joint Conf. on Artificial Intelligence, IJCAI'99*, pages 270–275. Morgan Kaufmann, 1999.

[Vap98] V. N. Vapnik. *Statistical Learning Theory*. Wiley, 1998.

Social Learning through Evolution of Language

Dimitar Kazakov and Mark Bartlett

Department of Computer Science, University of York,
Heslington, York YO10 5DD, UK,
lastname@cs.york.ac.uk,
http://www-users.cs.york.ac.uk/~*lastname*/

Abstract. This paper presents an approach to simulating the evolution of language in which communication is viewed as an emerging phenomenon with both genetic and social components. A model is presented in which a population of agents is able to evolve a shared grammatical language from a purely lexical one, with critical elements of the faculty of language developed as a result of the need to navigate in and exchange information about the environment.

1 Introduction

This paper discusses the evolution of language in societies of learning agents exposed to evolutionary pressure, and the impact the emerging social phenomenon of language has on the agents' performance on the task being learned.

There has been a lot of interest in adaptive and learning agents and multi-agent systems recently, with learning of co-ordination, timeliness of learning, combining evolutionary adaptation with individual learning and the learning of communication among the issues at the focus of attention [1]. Language has a dual rôle in societies of learning agents, as its concepts are a means of communicating knowledge, yet they also represent a repository of knowledge distributed among its speakers, but almost never present as a whole in the memory of anyone of them. Language is a powerful tool where co-operation is needed. In an evolutionary set-up where individuals of a species compete for survival, a speech act disclosing the location of valuable resources can be seen as altruistic, as it helps the listener at the speaker's expense. Altruism directed to relatives has been observed in nature [2] and proven to be evolutionary viable in computer simulations [3]. Here language will be seen as a form of kinship-driven altruism.

This paper builds on our earlier suggestion that some of the origins of human language can be found in the need to share information about the environment, which could result in *a proto-language, the grammatical structure of which copies the structure of the landscape* [4]. In a research review published shortly after, Hauser, Chomsky and Fitch [5] consider navigation as one possible computational problem, the solution to which required the evolution of recursion, which they assume to be the only uniquely human component of the faculty of language. Their paper also suggests a possible link between the abilities needed for optimal foraging and computationally efficient processing of language. In a more

P. Liardet et al. (Eds.): EA 2003, LNCS 2936, pp. 397–408, 2004.

general way, the suggestion that *"language could have developed as a means of transferring information about the spatial aspects of the environment (how to get somewhere, how to find food)"* appears as early as 1978, in a footnote in O'Keefe and Nadel's book [6]. We shall describe one possible mechanism through which language grounded in the environment could emerge, and study the impact that language has on the generic learning task of any living creature, namely, to increase success in survival and reproduction.

2 Simulations of the Evolution of Language

In recent years, there has been much research carried out in attempting to model the evolution of language through computer simulation. This research falls broadly into two classes, simulations in which language emerges in a single generation and simulations concerned with evolving a language over several generations.

In the former class, one of the most prominent researchers is Steels [7]. In his simulations, a population manages to arrive at a single, shared lexical language through participating in a series of 'language games'. In a language game, two agents both discuss an object visible to them. If they can agree on a word (or set of words) to describe that object, then they both increase the weight they associate with that word/meaning pair. After many language games involving different agents, a shared global lexicon emerges. Batali also presents an interesting simulation in which a language emerges [8]. In this case, a compositional language occurs through a series a language games. The mechanisms for playing the games and the memory that the agents possess is significantly different in the two simulations, but in both cases a globally understood language emerges from a series of local interactions.

Amongst those studying languages which are created over several generations, Kirby's simulations [9] are amongst the most compelling, though Zuidema & Hogeweg [10] and Oliphant & Batali [11] also present interesting results. In Kirby's simulations, a single agent attempts to express (resorting to invention if necessary) a subset of meanings sampled from a set of meanings expressed in predicate calculus while another agent attempts to learn to speak based on the linguistic output paired with the expressed meaning of the first agent. The agent which listened is then required to speak in the same way as the first agent, while its output is learned by yet another agent. After thousands of cycles of this expression/induction behaviour, a grammar with the minimum number of necessary rules is seen to emerge and persist from generation to generation. Kirby attributes this to the 'linguistic bottleneck' that prevents the observation of all meanings by a single agent. Only compositional grammars can successfully pass through this bottleneck, as idiosyncratic phrases present in a grammar may fail to be expressed at some cycle and be lost from the language. This explanation for the emergence of grammar differs considerably from the views expressed by many linguists, most notably Chomsky, who suggest that humans have become adapted to language, rather than language adapting to humans. Chomsky suggested that a learning bias facilitating the acquisition of language

is present in humans in the form of a highly specialised Language Acquisition Device (LAD) [12,13]. Learning a language by children exposed to it then would only require setting the specific parameters of the LAD in order to acquire an appropriate model (grammar) [14].

Our approach differs from all of those simulations mentioned above in several important aspects. Primarily, we see language as a tool to achieve some purpose. This means that we can consider issues such as when language will come to be used by a population, whereas other researchers have simply assumed that language is beneficial and sidestepped these issues. We also differ in our approach by assuming that the meaning of an utterance must be inferred from its context. The majority of previous research has explicitly given agents the signal and the meaning that it represents. We propose to remove this, instead having agents deduce the meaning of a signal based on the context within which it is used. Finally, we have agents subject to natural selection based on properties not directly linked to their linguistic ability. In other research, either agents have been selected directly for their ability to speak (making the results striking yet inevitable) or no selection of agents has been used (either the population is static or all agents are removed with equal probability).

3 Altruism and Neo-Darwinism

Our research is based within the domain of simulating learned communication systems. Here, of particular interest is the issue of why any individual should develop or choose to use such a system to speak. More specifically, we present a framework in which the urge to use language is seen as an inherited feature selected by evolution, while language itself is a social phenomenon that is passed on through interaction rather than genetically inherited. Clearly, an entity that is able to use such a communication system to understand the meanings of others' speech is at an advantage, as it can gain information through the work of others rather than its own toil. However, it is less apparent why an individual should choose to speak, when this will clearly give others an advantage that this one has had to work hard to gain.

In Darwinian terms, by helping others with no obvious benefit to itself, the individual has acted altruistically, decreasing its own fitness relative to that of others, and therefore we would expect such behaviour to be selected against by nature. However, the existence of human language clearly shows that in at least one case natural selection has acted opposite to this expectation. Researchers studying learned languages have not studied this question, but several researchers [15,16] have looked at similar problems in the domain of innate communication systems and we look to this work for possible approaches. They have found that, in their most abstract models, communication does indeed seem to be selected against if an agent can choose not to speak without penalty. However, there are possible modifications to these systems that seem to encourage communication to occur. A spatial distribution is one such modification that can be applied, with agents interacting more with those spatially adjacent to them.

This promotes reciprocal altruism, in which both entities benefit by cooperating rather than competing.

Another possible explanation is to look at the issue of altruism from a Neo-Darwinian perspective. Hamilton [17] shows us that if we view the basic element of evolution not as the individual, but as the gene, we find that natural selection may actually favour selfless acts in the form of kinship-driven altruism. This form of altruism involves helping relatives proportionally to their degree of kinship to the altruistic entity. For instance, should such an entity die saving the lives of three of its children, there will probably be more copies of its genes remaining alive than if it had preserved its own life. Through this mechanism, a hypothetical gene promoting altruism would be able to spread itself. We begin our current work assuming that altruism has been promoted and fixed in the environment as has been shown in our previous work [3].

4 Evolution of Language in Multi-agent Systems

We have chosen to simulate the evolution of language within a multi-agent system (MAS) setting. This allows us to simulate with ease many of the potentially relevant phenomena, such as the density and spatial distribution of agents (language speakers) and resources, along with the degree of agents' mobility. The MAS framework also permits to study the behaviour and social (both verbal and non-verbal) acts of each and every individual, if necessary.

Simulations of the evolution of language using the multi-agent paradigm can be of interest to the designer of any general-purpose agent-based applications. In a dynamic environment that changes considerably during an agent's lifetime, the faculty of learning could be essential to its success. Machine learning techniques could be used for this purpose [1]. Learning biases that specify the range of possible hypotheses are indispensable in machine learning, and their choice is crucial to the success of any of its algorithms. In an evolutionary MAS setting, sexual reproduction and mutation can be used to explore a range of possible learning biases, from which natural selection would choose the best. Evolution in the MAS can follow the Darwinian principle that individual experience cannot change one's genes. One would expect Darwinian evolution to advance in small steps, and select only very general concepts, as they would have to remain useful for many generations. One could also implement Lamarckian evolution: use a MAS in which the parents' experience can be reflected in the learning bias inherited by their offspring. Lamarckian evolution is faster, but brings the risks of inheriting too specific concepts based on the parents' personal experience.

This work explores yet another, third, way of evolving a learning bias that is open to populations of agents able to communicate. Language uses concepts that are specific enough to be useful in the description of a variety of aspects of the agent's environment (including other agents), yet general enough to correspond to shared experience. In this way, the concepts of a language serve as a bias used to describe the world that is inherited through social interaction rather than genes. However, to preserve the additional advantage that the use of language

brings about in the case of a changing environment, the MAS designer should allow the language to evolve.

5 The York Multi-agent Environment

The York Multi-Agent Environment [18] is a general-purpose Java-based platform for the simulation of learning and natural selection in a community of agents. The system draws a parallel with the world of nature. It provides tools for the specification of a two-dimensional environment with a range of features, such as different types of terrain, landmarks, food and water resources. The environment consists of a number of squares, each of which can hold up to four agents. The user can edit to a large extent the specification of each of a number of species of agents—what they eat, what types of sensors they possess. A snapshot of an ongoing experiment is shown in figure 1.

The agents have a default, hard-coded behaviour, which can be used as a baseline reference in the evaluation of learning. The behaviour is based on the notion of "drives", which represent the intensity of each of the agents' basic needs. There are three such drives used in the current implementation: hunger, thirst and sex drive. At each step, after a payment of energy is made to remain alive, the drives are evaluated and an appropriate action is taken. If at the beginning of a time step an agent finds itself to be hungry, it will attempt to find food. Likewise, if it is thirsty it will endeavour to find water. Should an agent require both food and water it will deal with the more pressing need. Utilising a resource will reduce the appropriate drive back to its minimum level.

If two agents sharing a location have adequately low levels of hunger and thirst, they can have offspring if their sex drive is sufficiently high and they can afford the cost of reproduction, which is subtracted as a payment from their energy levels. The offspring created has initial energy levels equal to the parents' cost of mating.

6 Evolving Songlines

Our approach was initially inspired by Chatwin's description of *songlines* [19], a form of oral tradition among Australian aboriginal tribes. Songlines reflect a belief that "*Ancestral Beings roamed once the face of the earth, creating the features of the landscape... along the tracks that mark their journeys* [20]." Songs and dances often function as title deeds to land, recording myths about the creation of particular sites. In the grouping of songs into series, *"the most pervasive is the geographical organisation of songlines, where each 'small song' equates with a different site along an ancestral journey, and the series as a whole constitutes a sung map* [20]." In our simplified interpretation, a songline describes a fixed migration route of a group of individuals as a list of landmarks, the evocative names (and descriptions) of which help unassisted recognition.

Disclosing the songline to outsiders is often strictly prohibited. One could see this as an attempt to protect the tribal knowledge about the location of scarce

resources. Nevertheless, there are situations in which members of different tribes may decide to exchange parts of their songlines. From our point of view, sharing the knowledge of a songline within the tribe can be seen as a form of kinship-driven altruism, and inter-tribal exchange of songlines as reciprocal altruism. The main characteristics of our approach are as follows:

- Use a MAS to simulate the evolution of language;
- Assume that altruistic behaviour between relatives (i.e., kinship-driven altruism) exists in the population, e.g., has been promoted by natural selection as demonstrated in our previous work [9]. In fact all agents behave as if clones sharing a single genotype;
- Set the agents to perform an altruistic act of sharing information about the location of resources in the environment. These resources are exhaustible, but when depleted will renew themselves after a period of time.

Information about resource location is shared in the form of paths, consisting of sequences of landmarks that are to be seen along the way to the target destination. Though the act of communication incurs no direct energy cost for the speaker, sharing information about resource location is clearly an altruistic act for two reasons. Firstly, it is likely to increase the fitness of the receiver, who will subsequently know the location of more resources and, secondly, the speaker is likely to have more difficulty in surviving, as it has to share the exhaustible resources with the receiver.

Landmarks are identified by unique names, which are shared by the whole population. The names are not derived from the landmark properties. There are two ways in which an agent may acquire a new path; it may find one through exploration, or it be given one linguistically by another agent. In both cases, rules are stored internally as a grammar rule of the form:

$$goto(TargetResource) \rightarrow goto(Pos(X)), L_1, L_2, \ldots, L_n$$

where L_i are landmark names, and *Pos(X)* is the current position of the listener defined by the snapshot of its surroundings, stored as a matrix of the visible landmarks with the vision range limited. Rules of the above form can be interpreted either as procedural rules guiding the agent from location X to resource of the given type or, alternatively, as grammar rules of a regular language.

It should be noted that the description of path chosen is a relatively impoverished one. So, for instance, no absolute or relative co-ordinates of landmarks are used, neither is the direction to follow or distance between consecutive landmarks described. The assumption made is that each landmark would be visible from the previous (or random exploration would be needed to find it).

In addition to the rules of the form presented above, it is vital that an agent can travel to the beginning point of a path: without a rule that states how to arrive at *Pos(X)*, the rule is of little practical use. To this end, agents collect and store information about the landmarks they pass while wandering around the environment in a short-term memory, which is distinct from its knowledge of grammar rules. When a position of some importance is reached, this sequence of landmarks is used to add a new grammar rule to the agent's knowledge stating the path between the locations in question. For example, an agent may begin

at position *Pos(A)* and randomly pass landmarks L_1, L_2 and L_3 before arriving at a significant position, *Pos(B)*. The following rule will then be added to the agent's knowledge base:

$$goto(Pos(B)) \rightarrow goto(Pos(A)), L_1, L_2, L_3.$$

This rule states that to travel to *Pos(B)* is equivalent to travelling to *Pos(A)* and then passing the given landmarks in order. Additionally the inverse of the rule is also added allowing the agent to return from *Pos(A)* to *Pos(B)*. A rule will also be added to explain why this point is of interest, namely which resource it contains:

$$goto(food) \rightarrow goto(Pos(B))$$

To access the information stored in the grammar in order to reach resources, the agent will need to generate a sequence of landmarks from the grammar, which will form a path that the agent can follow. This is done by using the grammar to generate a list consisting of only landmark names. The starting rule for this process is one of the rules with the left-hand side of *goto(TargetResource)*. These rules are kept in a queue with the first rule taken when one is needed and then returned to the rear. Enqueuing the rules in this way ensures agents do not become dependent on a small number of resources, but explore the environment more fully. In order that the parsing does not recurse indefinitely, an additional rule is needed stating that going to the current location of the agent can be rewritten as an empty sequence of landmarks. The parsing takes place using an A* search, with the metric based on the length of the path; this guarantees agents take the shortest route of which they are aware. As a route between the agent's current position and the resource it wishes to find may consist of following several sections of path (e.g. an agent may have to go from point A to point B to point C), the grammar is compositional: an agent will need to compose the rules to get from A to B with the rule to get from B to C in order to get from A to C. Furthermore, an agent may initially use one route to get from A to C, but on acquiring a shorter route between B and C, will change the route it takes between A and C appropriately.

When a pair of agents meet at a point, they may exchange routes to resources. As stated earlier, this is viewed as an altruistic act that has been previously promoted by evolution. Generating a route for exchange is carried out using exactly the same process that an agent would implement if generating the route for its own use. This route is subsequently passed to the other agent, who stores it as a rule of a similar form to those shown above. Whether the route generated leads to food or water is decided before the interaction occurs based on the needs of the agents concerned. This ensures that the agent receiving the route knows whether to follow the route when it is hungry or when it is thirsty.

7 Results and Evaluation

Intuitively, one would expect to find a relationship between the range of sight that the agents possess and the frequency of landmarks in the environment, on one hand, and the usefulness of the path representation chosen, on the other. For

instance, communication might be expected to provide the greatest benefit when the resources are spaced so they are frequently out of sight, and if one landmark is not so far from another than meaningful descriptions of paths become impossible. If, in a given environment, linguistic descriptions prove useful, i.e., storing and exchanging them promotes the survival of the agents involved, one would expect to observe the following two phenomena:

- Agents possess sets of rules describing paths, which ultimately lead to a useful resource;
- This set of rules can be seen as a *proto-language, the grammatical structure of which copies the structure of the landscape.*

Evaluation of the benefits of language was conducted using the environment shown in figure 1. Agents had a range of sight set to two squares in all directions (including diagonally). This prevented agents from observing a food resource when close to a water resource, but ensured plentiful landmarks by which to navigate. To compensate for the wide spacing between resources, agents had their exploration algorithm biased to give them a tendency to travel in a straight line with a probability of deviating left or right at each step. The environment was initially loaded with many more (uniformly distributed) agents than the resources were able to sustain in the long term. This overloading of the environment ensured that meetings between pairs of agents was frequent early in the simulation, since without these meetings language would not have been used and thus be of no consequence to the simulation.

The criteria used to measure any benefit that language provides to the community of speakers was to record population size and total energy (and water) of the population, and to compare this to a control population in which there was no exchange of information, but otherwise behaved identically: our hypothesis was that *language evolution is a form of multi-agent learning which allows a population to improve performance as measured by its exploitation of resources.*

The results of our experiment (as shown in figures 2, 3 and 4) clearly show a significant improvement in performance by the population of agents capable of using language. In this experiment, the maximum lifetime of an agent was set to be 300 cycles: if an agent had not died of hunger or thirst after this time, it died as a result of old age. At the end of the simulation run shown, the agents remaining in the population must be at least the fourth generation. Though both populations initially decline greatly in number (as was predicted due to the overloading of the environment), the population able to use language fairs better over the course of several generations than the other population, which in many cases dies out entirely by the third generation.

It is not difficult to see some of the potential benefits of language *a priori* due to the way in which an agent is able to survive. In the system without language, the only method of survival is for an agent to locate both resources by randomly discovering them while wandering in the environment, and then following the memorised paths. The probability of an agent surviving in a later generation is very similar to the probability of an agent surviving in the initial population, as it faces the same challenges of locating resources that the earlier generation

had. In the population where language was not suppressed, agents in the initial population had to locate the resources through exploration in much the same way as those in a population without language (though some agents unable to find the resources may be directed to them by more successful agents). However in later generations an agent is able to learn of the location of resources from its parents through language, and hence the survival rate of agents born into later generations of the speaking population is far greater due to the fact less exploration is needed. It was possible however, that sharing knowledge about resources could have lead to a population with similar, very low levels of energy, below the minimum needed for reproduction: while language could have helped more agents to survive, this population could have been less viable in the long term due to an inability to reproduce.

A separate form of evaluation is possible in which we examine the language as interesting in its own right, instead of viewing it as merely the way in which information is exchanged. By observing the internal language (set of rules) of each agent, one can measure the similarity between the languages used by each pair of agents as the ratio of shared rules and (hierarchically) cluster all agents accordingly. If agents are split into a set of mutually exclusive clusters, all agents within the same cluster can be seen as speakers of the same, shared language. In this case, language is seen as a social artefact that only exists in the community of its speakers.

In a second environment in which food and water were more plentiful and more evenly spread than in the previous experiment, simulations were run to examine the type of language which emerged. Reproduction was not used in these experiments, as the aim was to observe how the language spoken by agents changed over time, and the addition of new agents to the population could alter the way agents clustered. Our aim was not to study the advantage language gave in some task, hence only agents capable of language were studied without using non-speaking agents and no record of population dynamics was kept. Cluster diagrams resulting from a single run of this experiment are shown in figures 6–7. The diagrams show that the agents spontaneously arrange themselves into various distinct subsets, which manage to persist over time. Most of the agents which are closely related in the initial graph are still close in the latter graph, in most cases still clustering most tightly with the same agent, though some agents such as 50 and 21 do migrate to different groups. The difference between cluster diagrams for an environment with more resources which are more easily found, and that for an environment with lower numbers of resources is very apparent. When several food and water sources are available, the community splits into well-defined subsets each of which is linguistically distinct from the others. In contrast, when few resources are available, less well-defined groups are formed, with little distance between them. This is due to the fact that when many resources are available, agents find it possible to exist in a sub-region of the environment having little contact with the larger community.

In the sparser environment however, agents are forced into having similar grammars, as all rules gained will mention a small amount of significant places common to all individuals.

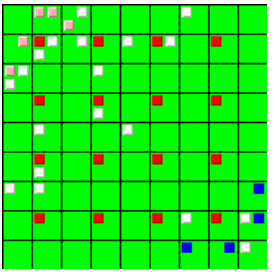


Fig. 1. The York Multi-Agent Environment. Agents are shown in white, water resources in light grey, food resources in black and landmarks in dark grey.

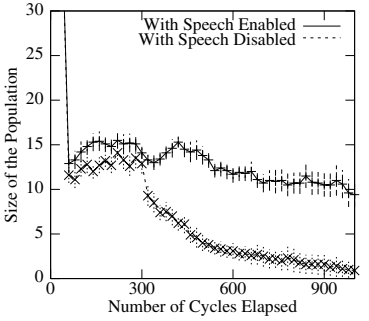


Fig. 2. Population size of communicating and non-communicating groups of agents.

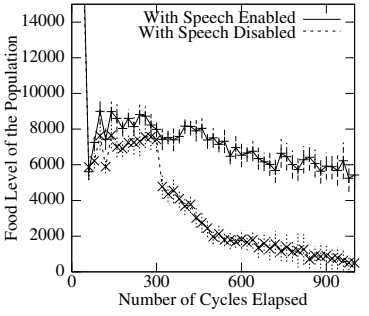


Fig. 3. Food levels of communicating and non-communicating groups of agents.

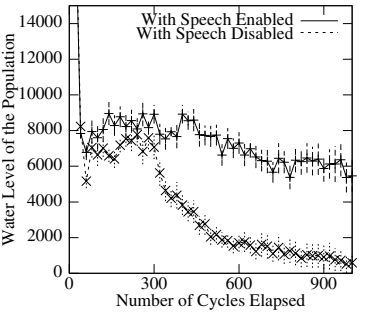


Fig. 4. Water levels of communicating and non-communicating groups of agents.

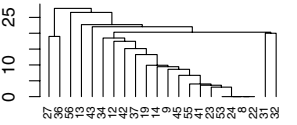


Fig. 5. Clustering by agent (after 40 cycles) for an environment with single food and water resources. Agents are referred to by their unique identifying numbers on the x-axis. Y-axis shows the number of different rules between clusters.

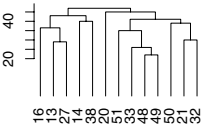


Fig. 6. Clustering by agent (after 80 cycles) for an environment with multiple resources.

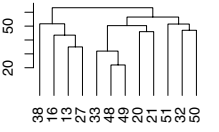


Fig. 7. Clustering by agent (after 100 cycles) for an environment with multiple resources.

Linking the properties of language to the environment in which it emerged has a precedent. Pagel [21] presents a detailed account of the variation of diversity of human languages in relation to environmental properties, such as latitude and number of habitats. He also demonstrates the correlation between linguistic diversity and genetic diversity (number of species) in a given ecosystem. One observation of particular interest is the reference to Birdsell's discovery that the density of Australian aboriginal languages is higher in wetter areas [22]. This is variously explained as a result of the higher potential of the environment to sustain more groups that are self-sufficient or due to migration towards richer areas.

8 Conclusions and Future Work

We have outlined in this paper one possible system whereby a useful simple compositional language is able to emerge from a purely lexical one. Furthermore, we have shown that the existence of such a language can improve the performance of agents able to use it. Our presented experiments have hinted at the possibility of language becoming severed into different mutually unintelligible communities under some conditions, a direction worth exploring further in the future.

Though currently, there is no explicit ranking of rules based on their usefulness in finding resources, a system does emerge whereby some rules are used more frequently and in preference to others. This occurs in situations where an agent has several rules for travelling between a single pair of points. In this situation, the route with the shorter length (i.e. the least number of landmarks) will always be chosen in preference to the other due to the metric used in the parsing process. Future improvements to the system would involve making such ranking explicit. Rules could then be pruned when found to be of little or no continuing use. This would prevent the apparent linguistic difference between agents speaking the same dialect increasing with time, even though they possess very similar knowledge. For similar reasons, it is important that duplicated information be removed from the grammar. This produces an unnecessary degree of separation between agents when they are clustered, and severely slows the speed of simulation. These changes would not lead to change in the behaviour of the system, however the grammar could also be acted on by some process of induction, with the aim of both reducing its size and producing new knowledge not previously held.

At present, we have assumed that landmarks can be distinguished between by globally recognised words. In our ongoing work, we intend to remove this condition and investigate a situation in which the same name is used for all landmarks of the same type, and these names are evolved through a process of negotiation as in Luc Steels' work [7] at the same time as the routes between resources are discovered.

References

1. Alonso, E., Kazakov, D., Kudenko, D., eds.: Adaptive Agents and Multi-Agent Systems. Springer-Verlag (2003).

2. Hamilton, W.D.: The genetic evolution of social behaviour II. Journal of Theoretical Biology (1964) 17–52.
3. Turner, H., Kazakov, D.: Stochastic simulation of inherited kinship-driven altruism. Journal of Artificial Intelligence and Simulation of Behaviour **1(2)** (2002) 183–196.
4. Kazakov, D., Bartlett, M.: A multi-agent simulation of the evolution of language. In: Proceedings of Information Society Conference IS'2002, Ljubljana, Slovenia, Morgan Kaufmann (2002) 39–41.
5. Hauser, M.D., Chomsky, N., Fitch, W.T.: The faculty of language: What is it, who has it, and how did it evolve? Science **298** (2002) 1569–1579.
6. O'Keefe, J., Nadel, L.: The Hippocampus as a Cognitive Map. Oxford University Press (1978).
7. Steels, L.: The spontaneous self-organization of an adaptive language. In: Machine Intelligence 15, St. Catherine's College, Oxford, Oxford University Press (1999) 205–224 Machine Intelligence Workshop: July 1995.
8. Batali, J.: The negotiation and acquisition of recursive grammars as a result of competition among exemplars. In: Linguistic Evolution through Language Acquisition: Formal and Computational Models. Cambridge University Press (2002).
9. Kirby, S.: Learning, Bottlenecks and the Evolution of Recursive Syntax. In: Linguistic Evolution through Language Acquisition: Formal and Computational Models. Cambridge University Press (2002).
10. Zuidema, W.H., Hogeweg, P.: Selective advantages of syntactic language - a model study. In: Proceedings of the Twenty-second Annual Conference of the Cognitive Science Society, Hillsdale, USA, Lawrence Erlbaum Associates (2000) 577–582.
11. Oliphant, M., Batali, J.: Learning and the emergence of coordinated communication. The newsletter of the Center for Research in Language **11** (1997).
12. Chomsky, N.: Aspects of the Theory of Syntax. MIT Press (1965).
13. Chomsky, N.: Language and Mind. Harcourt Brace & World (1968).
14. Chomsky, N.: Principles and Parameters in Syntactic Theory. In: Explanation in Linguistics. The Logical Problems of Language Acquisition. Longman (1981).
15. Oliphant, M.: Formal Approaches to Innate and Learned Communication: Laying the Foundation for Language. PhD thesis, Department of Cognitive Science, University of California, San Diego (1997).
16. Wagner, K.: Habitat, communication and cooperative strategies. In W. Banzhaf et al., ed.: Proceedings of the Genetic and Evolutionary Computation Conference, Morgan Kaufmann (1999) 694–701.
17. Hamilton, W.D.: The genetic evolution of social behaviour I. Journal of Theoretical Biology (1964) 1–16.
18. Kazakov, D., Kudenko, D.: Machine Learning and Inductive Logic Programming for Multi-agent Systems. In: Multi-Agent Systems and Applications. Springer (2001) 246–270.
19. Chatwin, B.: The Songlines. Picador (1987).
20. Barwick, L., Marett, A.: Aboriginal traditions. In: Currency Companion to Music and Dance in Australia. Currency Press, Sydney (2003) 26–28.
21. Pagel, M.: The history, rate and pattern of world linguistic evolution. In: The Evolutionary Emergence of Language: Social Function and the Origins of Linguistic Form. Cambridge University Press, Cambridge (2000).
22. Birdsell, J.B.: Some environmental and cultural factors influencing the structuring of Australian aboriginal populations. The American Naturalist **87** (1953) 171-207.

Author Index