Oleg B. Lupanov Oktay M. Kasim-Zade
Alexander V. Chaskin Kathleen Steinhöfel (Eds.)

Stochastic Algorithms: Foundations and Applications

Third International Symposium, SAGA 2005
Moscow, Russia, October 20-22, 2005
Proceedings

Springer

Volume Editors

Oleg B. Lupanov
Oktay M. Kasim-Zade
Alexander V. Chaskin
Moscow State University, Department of Discrete Mathematics
Leninskie Gory, Moscow 119992, Russia
E-mail: {lupanov, kasimz, chash}@mech.math.msu.su

Kathleen Steinhöfel
FIRST - Fraunhofer Institute for Computer Architecture and Software Technology
12489 Berlin, Germany
E-mail: kathleen.steinhoefel@first.fraunhofer.de

Library of Congress Control Number: 2005934273

CR Subject Classification (1998): F.2, F.1.2, G.1.2, G.1.6, G.2, G.3

ISSN 0302-9743
ISBN-10 3-540-29498-8 Springer Berlin Heidelberg New York
ISBN-13 978-3-540-29498-6 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 11571155 06/3142 5 4 3 2 1 0

Preface

This volume constitutes the proceedings of the 3rd Symposium on Stochastic Algorithms, Foundations and Applications (SAGA 2005), held in Moscow, Russia, at Moscow State University on October 20–22, 2005. The symposium was organized by the Department of Discrete Mathematics, Faculty of Mechanics and Mathematics of Moscow State University and was partially supported by the Russian Foundation for Basic Research under Project No. 05–01–10140–r. The SAGA symposium series is a biennial meeting which started in 2001 in Berlin, Germany (LNCS vol. 2264). The second symposium was held in September 2003 at the University of Hertfordshire, Hatfield, UK (LNCS vol. 2827).

Since the first symposium in Berlin in 2001, an increased interest in the SAGA series can be noticed. For SAGA 2005, we received submissions from China, the European Union, Iran, Japan, Korea, Russia, SAR Hong Kong, Taiwan, and USA, from which 14 papers were finally selected for publication after a thorough reviewing process.

The contributed papers included in this volume cover both theoretical as well as applied aspects of stochastic computations, which is one of the main aims of the SAGA series. Furthermore, five invited lectures were delivered at SAGA 2005: The talk by Alexander A. Sapozhenko (Moscow State University) summarizes results on the *container method*, a technique that is used to solve enumeration problems for various combinatorial structures and which has numerous applications in the design and analysis of stochastic algorithms. Christos D. Zaroliagis (University of Patras) presented recent advances in multiobjective optimization. Joachim Wegener (DaimlerChrysler AG, Research and Technology) introduced new search-based techniques for software testing, with particular emphasis on finding time-critical pathways in safety-relevant software components. Chrystopher L. Nehaniv (University of Hertfordshire) presented a comprehensive overview and the latest results on self-replication, evolvability and asynchronicity in stochastic worlds. The talk by Farid Ablayev (Kazan State University) analyzed from the communication point of view some proof techniques for obtaining lower complexity bounds in various classical models (deterministic, nondeterministic and randomized), and quantum models of branching programs.

We wish to thank all who supported SAGA 2005, all authors who submitted papers, all members of the Programme Committee and all reviewers for the great collective effort, all invited speakers, all members of the Organizing Committee, and the Russian Foundation for Basic Research for financial support.

October 2005

Oleg B. Lupanov, Oktay M. Kasim-Zade,
Alexander V. Chashkin, Kathleen Steinhöfel

Organization

SAGA 2005 was organized by the Department of Discrete Mathematics, Faculty of Mechanics and Mathematics of Moscow State University, Russia.

Organizing Committee

Andreas A. Albrecht	Manolis Christodoulakis	Kathleen Steinhöfel
Alexander V. Chashkin	Yoan Pinzon	

Programm Committee

Andreas A. Albrecht (University of Hertfordshire, UK)
Amihood Amir (Bar-Ilan University, Israel, and Georgia Tech, USA)
Michael A. Bender (State University New York, USA)
Alexander V. Chashkin (Moscow State University, Russia)
Walter Gutjahr (Vienna University, Austria)
Juraj Hromkovič (ETH Zürich, Switzerland)
Costas S. Iliopoulos (King's College London, UK)
Oktay M. Kasim-Zade (Co-chair, Moscow State University, Russia)
Irwin King (CUHK, Hong Kong, China)
Gad M. Landau (University of Haifa, Israel)
Oleg B. Lupanov (Co-chair, Moscow State University, Russia)
Kunsoo Park (Seoul National University, Korea)
Irina Perfilieva (Ostrava University, Czech Republic)
Tomasz Radzik (King's College London, UK)
Martin Sauerhoff (Dortmund University, Germany)
Christian Scheideler (Johns Hopkins University, USA)
Kathleen Steinhöfel (FhG FIRST, Germany)
Peter Widmayer (ETH Zürich, Switzerland)
Thomas Zeugmann (Hokkaido University, Japan)

Additional Referees

Gruia Calinescu	Jan Poland	Haixuan Yang
Lance Fortnow	Detlef Sieling	
André Gronemeier	Robert Spalek	

Sponsoring Institution

Russian Foundation for Basic Research

Table of Contents

Systems of Containers and Enumeration Problems

Alexander Sapozhenko[⋆]

Lomonosov University, Moscow

Abstract. We discuss a technique (named "the container method") for enumeration problems. It was applied for obtaining upper bounds and asymptotically sharp estimates for the number of independent sets, codes, antichains in posets, sum-free sets, monotone boolean functions and so on. The container method works even the appropriate recurrent equalities are absent and the traditional generating function method is not applicable. The idea of the method is to reduce a considered enumeration problem to evaluating the number of independent sets in the appropriate graph. We give some examples of such reduction and a survey of upper bounds for the number of independent sets in graphs. The method is usually successful if considered graphs are almost regular and expanders.

1 Introduction

We discuss a method for solving enumeration problems. The considered problems are not usually amenable to the traditional generating function method because of the absence of appropriate recurrent equations. The problems of estimating the number of independent sets in graphs, antichains in posets, sum-free sets in groups are among such problems. Our approach to solution of these problems based on the notion of covering system (or system of containers) for a family of sets. A family $\mathcal{B}$ of sets is called *covering* for a family $\mathcal{A}$, if for any $A \in \mathcal{A}$ there exists $B \in \mathcal{B}$ such that $A \subseteq B$. An element of the family $\mathcal{B}$ is called a *container* and $\mathcal{B}$ is called the *system of containers* for $\mathcal{A}$.

Suppose we want to know what is size of the family $\mathcal{A}$ of all sets with some given property Q. We can think of $\mathcal{A}$ as the family of subsets in the n-cube which are error correcting codes, sum-free sets in a finite group, vertex subsets of graph which are cliques and so on. The container method consists in finding a system of containers $\mathcal{B}$ with the special property. For example, it is good when there exists a subsystem $\mathcal{B}_1 \subseteq \mathcal{B}$ such that all (or almost all) $A \in \mathcal{B}_1$ possessing the property Q and simultaneously meeting the inequality

$$\sum_{A \in \mathcal{B} \setminus \mathcal{B}_1} 2^{|A|} \ll \sum_{B \in \mathcal{B}} 2^{|B|}. \tag{1}$$

⋆ Supported by the RFBR grant No 04-01-00359 (Russia).

Then we know that $|\mathcal{A}| \approx \sum_{B\in\mathcal{B}} 2^{|B|}$. The proof of (1) is usually reduced to upper bounds for independent sets in appropriate graph.

The aim of this paper to give examples where these simple considerations are successful. Except for the simplest cases, we does not present the complete proofs but only some sketches. We give references on the paper with the proofs. At the end, we consider the application of the container technique to estimating the complexity of some algorithms correctly working almost always.

2 Definitions

All graphs under consideration are finite, undirected, and simple. The vertices are considered as numbered. Denote the degree of a vertex v by $\sigma(v)$.

A subset of vertices of a graph G is called *independent* if the subgraph of G induced by A does not contain an edge. The family of all independent sets of G will be denoted by $\mathcal{I}(G)$. We put $I(G) = |\mathcal{I}(G)|$. Let $G = (V;E)$ be a graph with the vertex set V and the edge set E, and $v \in V$. We call the set $\partial v = \{u : (u,v) \in E\}$ the *boundary* of v. It is clear that $\sigma(v) = |\partial v|$. The *boundary* of $A \subseteq V$ in a graph $G = (V;E)$ is the set $\partial A = \left(\bigcup_{v\in A} \partial v\right) \setminus A$. Suppose $0 \le \delta < 1$. A graph G is called *δ-expander*, if $|A| \le |\partial A|(1-\delta)$ for any independent set A.

3 Reduction

In this section we discuss ways of reducing various enumeration problems to that of counting independent sets.

The simplest for reductions is the problem of enumerating cliques. We just use the fact that a clique in graph G corresponds to the independent set in the complement of G.

The enumeration of error-correcting codes with distance r is equivalent to that of independent sets in graph on the set of vertices of the n-cube, where an edge is an arbitrary pair (u,v) with Hamming distance less than r.

The number of sum-free sets in groups is also estimated above by the number of independent sets in the Cayley graph (see, for example, N.Alon [1]). Recall that a set A is sum-free in an additive group G if A does not contain triples a, b, c satisfying the equality $a + b = c$. For a group G and sets A, V the Cayley graph $\mathcal{C}_A(V)$ is defined as the graph on vertex set V s.t. (u, v) is an edge if $u+v$, $u - v$ or $v - u$ are contained in A. If $B \subseteq G$ is sum-free then B is independent in the Cayley graph $\mathcal{C}_A(G)$ for any $A \subseteq B$.

4 On the Number of Independent Sets

Here we obtain upper bounds for the number of independent sets in graphs by means of container method. We start with a simple lemma for regular graphs. A regular graph of degree k on n vertices will be called (n, k)-graph.

Lemma 1. *Let $G = (V; E)$ be a $(n, k)-$graph, $k > 1$, and $A \subseteq V$ be independent. Suppose $0 < \varphi < k$. Then there exists $T \subseteq A$ such that*

$$|T| \leq |\partial A|/\varphi \ , \tag{2}$$

$$A \subseteq D, \quad \textit{where} \quad D = D(T, \varphi) = \{v \in V \setminus \partial T : \ |\partial v \setminus \partial T| < \varphi\} \ , \tag{3}$$

and

$$|D| \leq |\partial T| \, \frac{k}{k - \varphi} \ . \tag{4}$$

Proof. The set T can be constructed by the following step-by-step procedure.

Step 1. Let u_1 be an arbitrary vertex from A. Set $T_1 = \{u_1\}$.
Suppose m steps have already been done and the set $T_m = \{u_1, \ldots, u_m\}$ has been constructed.

Step $m + 1$. If there exists $u_{m+1} \in A$ such that $|\partial u_{m+1} \setminus \partial T_m| \geq \varphi$, we put $T_{m+1} = T_m \cup \{u_{m+1}\}$. Otherwise, the procedure is complete and the result is $T = T_m$. The inequality (2) and the inclusion (3) evidently hold for the set T constructed as above. The inequality (4) follows from the facts that $|\partial v \cap \partial T| \geq k - \varphi$ for every $v \in D$ and $|\partial u| \leq k$ for every $u \in \partial T$. □

Denote by $\mathcal{I}(G)$ the family of independent sets of a graph G and set $I(G) = |\mathcal{I}(G)|$.

Corollary 1. *Let G be a (n, k)-graph, $1 \leq \varphi < k$. Then there exists the system of containers $\mathcal{F}$ for the family $\mathcal{I}(G)$ with the following properties:*

$$|\mathcal{F}| \leq \sum_{i \leq n/\varphi} \binom{n}{i}, \tag{5}$$

$$\textit{for any } D \in F \quad |D| \leq nk/(2k - \varphi). \tag{6}$$

Proof. By Lemma 1, every independent set A contains a subset T of size not exceeding $|\partial A|/\varphi$ provided that $A \subseteq D(T, \varphi)$. Note that $D(T, \varphi)$ is uniquely defined by T if φ is fixed. Thus the number of containers does not exceed the number of vertex subsets of size $|\partial A|/\varphi \leq n/\varphi$. From this (5) follows.

The inequality (6) follows from (4) in view of $|D| \leq n - \partial T$. □

The following theorem improves the error term in Alon's upper bound [1]. It also gives an example of application of the container method.

Theorem 1. *For any (n, k)-graph*

$$I(G) \leq 2^{\frac{n}{2}\left(1+O\left(\sqrt{(\log k)/k}\right)\right)} \ . \tag{7}$$

Proof. The inequality (7) immediately follows from the corollary. All independent sets can be enumerated in the following way. Fix $\varphi = \sqrt{k \log k}$. Choose some $T \subseteq V$ and construct $D = D(T, \varphi)$. Then choose $A \subseteq D$. Note that

$$|D| \leq nk/(2k - \varphi) \ , \tag{8}$$

in view of (6). Hence, under $\varphi = \sqrt{k \log k}$, we have

$$I(G) \le \sum_{T \subseteq V, |T| \le n/\varphi} 2^{|D(T,\varphi)|} \le \sum_{i \le n/\varphi} \binom{n}{i} 2^{nk/(2k-\varphi)}$$

$$\le 2^{\frac{n}{2}\left(1+O\left(\sqrt{(\log k)/k}\right)\right)} \ . \tag{9}$$

□

Theorem 1 was extended to the case of irregular graphs (see[17]). The need of extensions caused by applications to enumeration problems from algebra and number theory (see[3], [5], [11], [18], [19]).

G is called (n, k, θ)*–graph*, if it has n vertices and $k \le \sigma(v) \le k + \theta$ for any vertex v. By definition, $(n, k, 0)$-graph is regular of degree k.

Theorem 2. *For any (n, k, θ)-graph Γ*

$$I(\Gamma) \le 2^{\frac{n}{2}\left(1+O\left(\theta/k+\sqrt{(\log k)/k}\right)\right)} \ . \tag{10}$$

This theorem extends the previous one to "almost" regular graphs.
Denote by $I_\beta(\Gamma)$ the number of subsets $A \in \mathcal{I}(\Gamma)$ meeting the condition

$$||A| - n/4| \ge \beta n/4 \ .$$

Theorem 3. *Let $\Gamma = (V; E)$ be (n, k, θ)-graph and $0 < \beta < 1$. Then*

$$I_\beta(\Gamma) \le 2^{\frac{n}{2}\left(1-\frac{\beta^2}{2\ln 2}+O\left(\frac{\theta}{k}+\sqrt{\frac{\log k}{k}}\right)\right)} \ . \tag{11}$$

Theorem 4. *Let (n, k, θ)-graph $\Gamma = (V;\ E)$ be δ-expander for some $0 \le \delta < 1$. Then*

$$I(\Gamma) \le 2^{\frac{n}{2}\left(1-\delta/7+O\left(\theta/k+\sqrt{(\log k)/k}\right)\right)} \ . \tag{12}$$

Theorems 3 and 4 allow to obtain the upper bounds of the form $I(G) \le 2^{n(1/2-c)}$, where $c > 0$ is some constant. It is very helpful when proving assertions that some subsystem of containers gives small contribution (see, for example, [18]).

The further similar extensions were proved in [21]. Suppose l, k, θ, n meet the inequalities $l \le k - \theta \le k + \theta \le n$. A graph on n vertices is called a $(n, l, k, m, \delta, \varepsilon, \theta)$*-graph* if it meets the following condition: the minimal vertex degree is at least l, the maximal vertex degree is at most m, the fraction of vertices with a degree exceeding $k + \theta$ is not more than ε, and the fraction of vertices with a degree less than $k - \theta$ is not more than δ. In [19] the following statement is proved.

Theorem 5. *Let $G = (V, E)$ be a $(n, l, k, m, \delta, \varepsilon, \theta)$-graph. Then there exists a system $\mathcal{B}$ of containers for $\mathcal{I}(G)$ meeting the following two conditions:*

1. For any $B \in \mathcal{B}$

$$|B| \le n \frac{k + \delta(k-l) + \varepsilon(m-k) + \theta}{2k - \sqrt{k \log k}} \ ; \tag{13}$$

2. For $k > 3$ and large enough n

$$|\mathcal{B}| \le 2^{n\sqrt{\frac{\log k}{k}}} \ . \tag{14}$$

Furthermore, for large enough n

$$I(G) \le 2^{\frac{n}{2}\left(1+\delta\left(1-\frac{l}{k}\right)+\varepsilon\left(\frac{m}{k}-1\right)+O\left(\frac{\theta}{k}+\sqrt{\frac{\log k}{k}}\right)\right)} \ . \tag{15}$$

The following two theorems proved in ([21]) are extensions of theorems 3 and 4.

Theorem 6. *Let G be a $(n, l, k, m, \delta, \Delta, \theta)$-graph and*

$$\gamma = \delta\,(1 - l/k) + \Delta\,(m/k - 1) + O\left((\theta + \sqrt{k \log k})/k\right) \ .$$

Then for large enough n

$$I_\beta(G) \le 2^{\frac{n}{2}\left(1-(\beta-\gamma)^2/(2(1+\gamma)\ln 2)\right)} \ . \tag{16}$$

Theorem 7. *Let $(n, l, k, m, \delta, \Delta, \theta)$-graph G be ϵ-expander. Then for some absolute constant $c > 0$ and large enough n*

$$I(G) \le 2^{\frac{n}{2}\left(1-c\epsilon+\delta(1-l/k)+\Delta(m/k-1)+O\left(\left(\theta+\sqrt{k\log k}\right)/k\right)\right)} \ . \tag{17}$$

Theorems 5 – 7 sufficiently extend abilities of the container method. In particular, Theorem 5 is applied for proving the Cameron-Erdös conjecture in [19].

5 Bipartite Graphs

In the case of bipartite graphs the container method allows to get asymptotically ultimate result if the graph is an expander. A bipartite graph $G = (X, Z; E)$ with the vertex partition sets X, Z and the edge set E will be called an *two-sided (ϵ, δ)-expander* if $|A| \le |\partial(A)|(1-\delta)$ for any $A \subseteq X$ such that $|A| \le \epsilon\,|X|$ and for any $A \subseteq Z$ such that $|A| \le \epsilon\,|Z|$. The first asymptotical result for the number of the number of independent sets in regular bipartite graphs obtained with the help of container idea concerned the n-cube. Let B^n be the n-cube, which is n-regular graphs on $N = 2^n$ vertices. In [10] the following assertion was proved

Theorem 8.

$$I(B^n) \sim 2\sqrt{e}2^{2^{n-1}} = 2\sqrt{e}2^{N/2} \ . \tag{18}$$

In [16] the following theorem was proved.

Theorem 9. *Let a bipartite (n, k, θ)-graph $G = (X, Z; E)$ be a two-sided$(1/2, \delta)$-expander and z be maximum of solutions of the equation $x = \log_2 (2ex/c\delta)$. Then for large enough n and k*

$$2^{|X|} + 2^{|Z|} - 1 \le I(G) \le \left(2^{|X|} + 2^{|Z|}\right) \left(1 + 2^{-k\delta/z + O\left(\sqrt{k}\log k + \theta\right)}\right) \ . \qquad (19)$$

The proof of the upper bound in (19) is based on the idea that X and Z are "main" containers covering almost all independent sets of G. The number of remaining independent sets does not exceed

$$\left(2^{|X|} + 2^{|Z|}\right) 2^{-k\delta/z + O\left(\sqrt{k}\log k + \theta\right)}.$$

This follows from smallness of size and number of containers covering these sets.

Corollary 2. *Let $G_n = (X_n, Z_n; E_n)$ be a sequence of bipartite $(n, k(n), \theta_n)$-graphs which are simultaneously $(1/2, \delta_n)$-expanders. Suppose*

$$k(n)\delta_n \to \infty \quad \text{and} \quad \theta_n/k(n) + k(n)^{-1/2} \log k(n) \to 0 \qquad (20)$$

as $n \to \infty$. Then

$$I(G_n) \sim 2^{|X_n|} + 2^{|Z_n|} \ . \qquad (21)$$

Note that the corollary 2 means that the following statement holds: If a sequence $G_n = (X_n, Z_n; E_n)$ satisfies the condition of 2, then the portion of independent sets of G_n intersecting both X_n and Z_n tends to 0 as $n \to \infty$. The similar assertions plays a part in describing the phase transition processes (see, for example, citeBB and citeST).

A bipartite graph $G = (X, Z; E)$ is called *boundary* $(\epsilon, delta)$*-expander*, if

$$|A| \le |\partial A|(1 - \delta) \qquad (22)$$

for all $A \subseteq X$ with $|\partial A| \le \lceil \epsilon |Z| \rceil$.

Let $(\kappa, \nu, q, p, \lambda)$ be a tuple of numbers with κ, ν, λ integers, and q, p real. A bipartite graph $\Gamma = (X, Z; E)$ such that

$$(1) \quad \min_{v \in X} \sigma(v) = \kappa, \qquad (2) \quad \max_{v \in Z} \sigma(v) = \nu, \qquad (3) \quad \min_{v \in Z} \sigma(v) = \lambda,$$

will be called the $(\kappa, \nu, q, p, \lambda)$*–graph*, if the following inequalities hold

$$(4) \quad \max_{v \in X \cup Z} \sigma(v) \le \kappa^p; \qquad (5) \quad \max_{\substack{u, v \in X \\ v \ne u}} \left|\partial u \cap \partial v\right| \le q.$$

If some of these conditions fails or are not sufficient, the corresponding coordinates in the tuple will be replaced by "-". For example, if the properties 1, 5 hold and the remaining maybe are not, we say about a $(\kappa, -, q, -, -)$-graph.

A bipartite graph $G = (X, Z; E)$ is called *one-sided* (ε, δ)*– expander* if

$$|A| \le |\partial A|(1 - \delta)$$

for any subset $A \subseteq X$ such that

$$|A| \leq \lceil \varepsilon |X| \rceil.$$

A bipartite graph $G = (X, Z; E)$ is called *one-sided boundary* (ε, δ)– *expander* if

$$|A| \leq |\partial A|(1 - \delta)$$

for any $A \subseteq X$ such that

$$|\partial A| \leq \lceil \varepsilon |Z| \rceil.$$

Let $G = (X, Z; E)$ be a (Δ, κ, q, p)–graph. Set

$$r_0 = r_0(G) = \lceil \kappa/q \rceil, \qquad g_1 = g_1(G) = \min_{A \subseteq X : |A| > r_0} |\partial A|,$$

$$g_2 = g_2(G) = \kappa^3 \log^{-7} \kappa, \qquad \varepsilon_1 = \varepsilon_1(G) = g_2/|Z|,$$
$$\delta_1 = \delta_1(G) = \kappa^{-1} \log^2 \kappa, \qquad \delta_0 = \delta_0(G) = \kappa^{-2} \log^9 \kappa.$$

A $(\kappa, -, q, p, -)$–graph $G = (X, Z; E)$ will be called a (Δ, κ, q, p)–expander, if it is a boundary $(\varepsilon_1, \delta_1)$–expander, a one-sided $(1, \delta_0)$–expander and the following inequalities hold:

$$\kappa/\sqrt{\log \kappa} \leq \min_{v \in Z} \sigma(v) \leq \max_{v \in Z} \sigma(v) \leq \kappa, \tag{23}$$

$$|X| \leq 2^{(\Delta+1)\kappa - 2\log^2 \kappa}. \tag{24}$$

A sequence $(v_1, v_2, ..., v_k)$ of vertices in a graph $G = (V, E)$ is called a d–*chain* if the distance between v_i and v_{i+1} in G does not exceed d for $1 \leq i < k$. A set $A \subset V$ will be called *connected with distance* d, if any two vertices of A are connected by d-chain. A subset $B \subseteq A$ is called d-component of A, if B is d–connected but $B \bigcup \{v\}$ is not connected for any $v \in A \setminus B$. Denote by $I_\Delta(G)$ the number of independent sets A in a bipartite graph $G = (X, Z; E)$ with the property that the size of each 2-component of $A \bigcap X$ does not exceed Δ. In [15] the following statement is proved

Theorem 10. *Let $G = (X, Z; E)$ be total (Δ, κ, q, p)-expander. Then*

$$I_\Delta(G) \leq I(G) \leq \Big(1 + O\big(2^{-\frac{3}{2}\log^2 \kappa}\big)\Big)\ I_\Delta(G). \tag{25}$$

Theorem 11. *Let $G = (X, Z; E)$ be a total $(1, \kappa, q, p)$-expander. Then*

$$I(G) = \Big(1 + O\big(2^{-\frac{3}{2}\log^2 \kappa}\big)\Big)\ 2^{|Z|} \exp\Big\{\sum_{v \in X} 2^{-|\partial\{v\}|}\Big\}. \tag{26}$$

If $G = (X, Z; E)$ be a total $(2, \kappa, q, p)$-expander. Then

$$I(G) = 2^{|Z|} \exp\Big\{\big(1 + O\big(\kappa^2 2^{-\kappa}\big)\big) \sum_{v \in X} 2^{-|\partial\{v\}|}\Big\}. \tag{27}$$

Theorem 11 gives the asymptotically sharp estimate for many important cases.

6 Sum-Free Sets of Integers

Here we show how the containers are applied for solving enumeration problem in the number theory. A subset A of integers is said to be *sum-free* (abbreviation, SFS) if $a+b \notin A$ for any $a, b \in A$. For any real $p \leq q$ denote by $[p, q]$ the set of integers x such that $p \leq x \leq q$. The family of all SFS's of $[t, n]$ is denoted by $S(t, n)$. Put $s(t, n) = |S(t, n)|$, $S(n) = S(1, n)$ and $s(n) = |S(n)|$. In 1988 P. Cameron and Erdős conjectured in [3] that $s(n) = O(2^{n/2})$. This conjecture was proved independently by B.Green [5] and the author [19]. B.Green used the Fourier transformation technique. Our approach uses the idea of containers.

It is easy to see that the family $S_1(n)$ of all subsets of odd numbers consists of sum-free sets. The other "big" family of sum-free sets is the family $S_2(n)$ of all subsets of the interval $[\lfloor n/2 \rfloor + 1, n]$. We slightly correct the second family. Set $\hat{q} = n^{3/4} \log n$ and $t = \lfloor n/2 - \hat{q} \rfloor$. Denote by $S_3(n)$ the family of all subsets of the interval $[t, n]$. It is clear that

$$s(n) \geq |S_1 \cup S_3| \geq |S_1(n)| + |S_3(n)| - 2^{(n/4)+1} - \hat{q} \sim |S_1(n)| + |S_3(n)| \ . \quad (28)$$

Note that $|S_1| = 2^{\lceil n/2 \rceil}$. In [3] it was proved that $s(n/3, n) = O(2^{n/2})$. Since $|S_3| \leq s(n/3, n)$, we need only to prove that the size of the family $\widetilde{S}(n) = S(n) \setminus (S_1(n) \cup S_3(n))$ is of small enough size. In [19], we proved that

$$\widetilde{S}(n) = o(2^{n/2}). \quad (29)$$

This implies that

$$s(n) \sim |S_1(n)| + |S_3(n)| \ . \quad (30)$$

When proving (23), we do as follows. First, we prove the existence a so-called almost correct system of containers.

We use denotation N for interval of integers $[1, n]$. Set $\widetilde{q} = \hat{q} \log n$. Consider two families $\mathcal{A}$ and $\mathcal{B}$ of subsets of N. We say that $\mathcal{B}$ is a *system of containers* for $\mathcal{A}$ if for any $A \in \mathcal{A}$ there exists $B \in \mathcal{B}$ such that $A \subseteq B$. A set $B \in \mathcal{B}$ is called a *container*. A family $\mathcal{B}$ of subsets of the set N will be called *correct* if the following conditions hold:

1. For large enough n and any $B \in \mathcal{B}$
$$|B| \leq n/2 + O(\hat{q}) \ . \quad (31)$$
2. For large enough n
$$|\mathcal{B}| \leq 2^{o(\hat{q})} \ . \quad (32)$$
3. For any $i \in [\widetilde{q}, n - \widetilde{q}]$ and $p \in [\widetilde{q}, n - i]$
$$||B_{i,p}| - p/2| \leq \hat{q} \ . \quad (33)$$
4. For any $\sigma \in \{0, 1\}$, $i \in [\widetilde{q}, n - \widetilde{q}]$, and $p \in [\widetilde{q}, n - i]$
$$||B_{i,p} \cap N^{\sigma}| - p/4| \leq \hat{q} \ . \quad (34)$$

A family $\mathcal{B}$ is called an *almost correct* system of containers for $\mathcal{A}$ if it correct for some subfamily $\mathcal{A}' \subseteq \mathcal{A}$ such that $|\mathcal{A} \setminus \mathcal{A}'| = o(2^{n/2})$.

The existence of almost correct system of containers for $\widetilde{S}(n)$ is proved by reducing this problem to similar one for the family of independent sets in the appropriate Cayley graph and using theorem 5. The other fact used is the well-known Freiman theorem [4].

Theorem 12 (G. A. Freiman [4]). *Suppose a set K of integers meets the condition $|K+K| \leq 2|K| - 1 + b$, where $0 \leq b \leq |K| - 3$. Then K is contained in some arithmetic progression of length $|K| + b$.*

The properties 3 and 4 of the correct system of containers together with Freiman's theorem allow to obtain upper bound (29). Thus, in the considered case we use special properties of containers from auxiliary system to prove a smallness of the number of enumerated objects. At the same time subsets from the main system (which is $S_1(n) \bigcup S_3(n)$) are almost all sum-free sets. These circumstances lead to getting asymptotically sharp result.

7 Sum-Free Sets in Groups

In this section we consider application of the container idea to enumerating sum-free sets in abelian groups of even order. As in the case of sum-free sets in a segment of integers in this case the asymptotically sharp result is obtained (see [11] and [18]). The result is the following

Theorem 13. *For any sufficiently large even n and for any abelian group G of order n with t subgroups of index 2*

$$t \cdot 2^{n/2} - 2^{(n/4)(1+o(1))} \leq s(G) \leq t \cdot 2^{n/2} + 2^{n(1/2-c)} \ , \tag{35}$$

where $c > 0.01$.

The idea of the proof is to find the appropriate system of containers. Note that every coset of subgroup in a group is a sum-free set as well as any subset of a coset is also sum-free. In particular, the family of cosets of index two subgroups seems to be available to be system of containers for almost all sum-free sets in a group. In any case, this family gives a good lower bound for the number of sum-free sets.

Let G be an abelian group of order $n = 2m$. Denote by $\mathcal{S}(G)$ the family of sum-free sets in a group G. We call a set $C \in \mathcal{S}(G)$ *regular* if there exists an index two subgroup K such that $A \subseteq K'$ where K' is the coset of K. Otherwise, the set C is called *irregular.*

Denote by $\mathcal{S}_1(G)$ the set of regular sets $C \in \mathcal{S}(G)$, and by t the number subgroups of index two in G. Then

$$t\, 2^{n/2} - \binom{t}{2} 2^{n/4} \leq |\mathcal{S}_1(G)| \leq t\, 2^{n/2}. \tag{36}$$

The upper bound in (36) follows from the definition. The lower bound is the consequence of including–excluding formula and the fact that the intersection of two different subgroups of index two consists of $2^{n/4}$ elements. Besides, we use that $t \leq 2^{\log^2 n}$. So we have obtained the lower bound in (35).

Now we have to prove that the number of remaining (irregular) sum-free sets is small, i.e. this number is $o(t2^{n/2})$. We present the facts and considerations leading to the goal. First we use the fact that for any index two subgroup H each remaining sum-free set A both H and its coset $\overline{H}$. So we can consider the Cayley graph $\mathcal{C}_{H \cap A}(G)$ and reduce our problem to enumeration of independent sets in $\mathcal{C}_{H \cap A}(G)$. The following statement plays important role in the proof.

Theorem 14. (M. Kneser) *Let A and B be non-empty subsets of an abelian group G. Suppose that $|A+B| \leq |A|+|B|-1$. Then there exists subgroup H of G such that*

$$A+B+H = A+B \qquad \textit{and} \qquad |A+B| \geq |A+H|+|B+H|-|H| \ . \quad (37)$$

Kneser's theorem allows to prove that the Cayley graph $\mathcal{C}_{H \cap A}(G)$ is a δ-expander and hence to use theorem 4.

8 The Extended Dedekind Problem

Now we consider the question on the number of antichains in ranked posets. A subset A of poset is called *antichain* if any two elements are non-comparable. The particular case of the question when the poset is the n-cube is conventional to call the Dedekind problem. With the common case in mind, we will speak about the *extended* Dedekind problem. The problem was posed in 1897 when R.Dedekind counted the number of antichains in the 4-cube. A lot of papers was written on the topic. In 1980 A. D. Korshunov ([9]) obtained the asymptotically complete solution. The author in ([13], ([14]) found asymptotic solution of the extended Dedekind problem. Namely, asymptotics for the number of antichains in so-called unimodal posets were obtained. Roughly speaking, an unimodal poset is a cube-like ranked poset. The proof of the above mentioned results fall in the pattern of the container method. At first, the problem is reduced to that of 3-levelled poset $P = (X, Z, Y)$, consisting of the largest levels of the considered poset. The latter it turns out to be equivalent evaluating the number of independent sets in the bipartite graph $G = (X \bigcup Z, Y; E)$ with vertex partition sets $X \bigcup Z$ and Y, where the vertices $u \in X \bigcup Z$ and $v \in Y$ are joint by edge iff they are comparable in P.

9 Application to Algorithm Complexity in Typical Case

The information on the structure of main containers can be used for evaluation of the complexity of algorithms. We consider an example, connected with problem of *deciphering* monotone Boolean functions. The problem is to reconstruct a

Boolean functions with minimum number of queries to a black box. A black box knows some monotone Boolean function f and gives the answer of the form $f(\tilde{x})$ on request in the form of Boolean vector $\tilde{x} = (x_1, ..., x_n)$. In deterministic case the problem was investigated by V. K. Korobkov. Denote by $\varphi(n)$ the least number of queries to black box in the worst case. V. K. Korobkov [8] proved that $\varphi(n) = O\binom{n}{\lfloor n/2 \rfloor}$. The Hansel [6] obtained final result:

Theorem 15.

$$\varphi(n) = \binom{n}{\lfloor n/2 \rfloor} + \binom{n}{\lfloor n/2 \rfloor + 1}. \tag{38}$$

Now consider the probabilistic case. Denote by $\psi(n)$ the least number of queries sufficient for deciphering almost all functions on n variables. The following result follows from [15].

Theorem 16.

$$\psi(n) = \binom{n}{\lfloor n/2 \rfloor} + O\left(\binom{n}{\lfloor n/2 \rfloor} 2^{-n/2}\right). \tag{39}$$

Sketch of proof. The lower bound follows from the pigeon principle and the lower bound for the number of monotone Boolean functions. The algorithm giving the upper bound is the following. We ask the black box about function values on all vectors with $\lfloor n/2 \rfloor$ ones and then ask it on the values on those vectors with $\lfloor n/2 \rfloor \pm 1$ ones that do not contradict chosen values on vectors with $\lfloor n/2 \rfloor$ ones. By [9] and [15], this number is almost always $O\left(\binom{n}{\lfloor n/2 \rfloor} 2^{-n/2}\right)$.

The second algorithmic problem is finding an upper zero of monotone Boolean function. A binary vector $\tilde{x} = (x_1, ..., x_n)$ is called the upper zero of a monotone Boolean function f if $f(\tilde{x}) = 0$ and the weight of $\tilde{x}$, defined by the equality $\sum_{1 \le i \le n} x_i$, is maximum among all vectors making f vanish. The problem is to find an upper zero with the least queries to the black box. In the deterministic case the problem was investigated by N. A. Katerinochkina. Denote by $\mu(n)$ the least number of queries to the black box for finding an upper zero of a monotone Boolean function on n variables in the worst case. In [7] the following estimate are obtained:

$$mu(n) = \binom{n}{\lfloor n/2 \rfloor} + 1. \tag{40}$$

The typical case was investigated by the author[20]. Denote by $\nu(n)$ the least number of queries sufficient for finding an upper zero for almost all functions of n variables. It was shown that for any $\omega(n)$ tending to infinity as $n \to \infty$ for almost all monotone Boolean function

$$nu(n) \le \omega(n) 2^{n/2}. \tag{41}$$

The function $\omega(n)$ can not be replaced by a constant. The proof uses the fact that the upper zeroes of almost all functions are of weight $\lfloor n \rfloor$ or $\lceil n \rceil$. Besides, it uses the fact that the typical functions have a zero among the first $\omega(n)2^{n/2}$ vectors of the corresponding weight. This property follows from the structure almost all functions obtained with the help of the container method.

10 Some Unsolved Problems

Here we discuss some unsolved problems relevant to the topic of the paper. All these problems can likely be solved by container method after some improving the technique.

1. Error correcting codes.
   The number of binary codes with distance 2 was found in [10]. The question on the number of binary codes with distance more than 2 is open. The difficulty is that the expendability of the corresponding graph is not proved.
2. Antichaines in Cartesian degree of the chain on 3 vertices.
   The question on the number of antichains in Cartesian degree of the chain on 2 vertices is known as the Dedekind problem. It is solved. The difficulty of extending is the irregularity of the corresponding lattice.
3. Sum-free sets in groups.
   Asymptotic value of the number of SFS's in groups of prime order is known not for all such groups.
   There are little or no on the number of SFS's for noncommutative groups.

References

[1] Alon N., Independent sets in regular graphs and Sum-Free Subsets of Finite Groups. Israel Journal of Math., 73 (1991), No 2, 247-256.

[2] B. Bollobas, Random Graphs, Second eddition, Cambridge University Press, 2001, 495 p.

[3] Cameron P., Erdős P., On the number of integers with various properties. In R. A. Mollin (ed). Number Theory: Proc. First Conf. Can. Number Th. Ass., Banff, 1988, — de Gruyter. 1990 — P. 61-79.

[4] Freiman G.A., Finite set addition. Izv. Vyssh. Uchebn. Zaved., Matematika, **6 (13)**, (1959), 202-213.

[5] Green B., The Cameron-Erdos conjecture, Bull. Lond. Math. Soc. 36(2004), no. 6, 769-778.

[6] Hansel G., Sur le nombre des fonctions booleen monotonesde n variables, C.R. Acad. Sci. Paris, 262, (1966), 1088-1090.

[7] Katerinochkina N. n., Searcing of maximum upper zero of monotone Boolean functions. Doklady Acad. Nauk of URSS, v. 224, No. 3, 1975, 557-560, (in Russian).

[8] Korobkov V. K., On monotone Boolean functions. Problemy Kibernetiki, V.38, M. Nauka, 1981, 5-108. (in Russian)

[9] Korshunov A. D., On the number of monotone Boolean functions. Problemy Kibernetiki, V.13, M. Nauka, 1965, 5-28. (in Russian)

[10] Korshunov A. D., Sapozhenko A. A., On the number of binary codes with distance two. Problemy Kibernetiki, M. Nauka, Vol.40, 1993, 111-140 (in Russian).

[11] Lev V. F., Luczak T., Schoen T., Sum-free sets in Abelian groups. Israel Journ. Math. 125 (2001) 347, 347-367.

[12] Sapozhenko A. A., On the number of connected sets with given size of boundary in graphs. Metody diskretnoi matematiki v reshenii combinatornykh zadach, - Novosibirsk, (1987), v.45, 35-58.

[13] Sapozhenko A. A., On the number of antichains in posets. Discrete Mathematics and Applications - Utrecht, The Nethelands, Tokio, Japan, - v.1, No.1, - 35-59.
[14] Sapozhenko A. A., On the number of antichains in multylevelled posets. Discrete Mathematics and Applications - Utrecht, The Nethelands, Tokio, Japan, - v.1, No.2, - 149-171.
[15] Sapozhenko A. A., The Dedekind problem and boundary funcional method. Matematicheskie Voprosy Kibernetiki, M. Fizmatlit, 2000, No. 9, 161-220,
[16] Sapozhenko A. A., On the Number of Independent Sets in Bipartite Graphs with Large Minimum Degree. DIMACS Technical Report 2000-25, August 2000, 24-31,(in Russian).
[17] Sapozhenko A. A., On the Number of Independent Sets in expanders. Diskretnaya matematika, Moscow, v. 13, No. 1, 2001, 56-62. (in Russian)
[18] Sapozhenko A. A., On the number of sum-free sets in Abelian groups. Vestnik Moskovskogo Universiteta, ser. Math., Mech. 2002, No. 4, 14-18. (Russian).
[19] Sapozhenko A. A., The Cameron-Erdos conjecture. Doklady of Russian Academy of Sciences, 2003, v. 393, No. 6, P. 749-752. (English Translation).
[20] Sapozhenko A. A., On searcing upper zeroes of monotone functions on ranked posets. Journal of Mathematical Physics and Computational Mathematics, 1991, v.31, No 12, 1871-1884. (English translation)
[21] Sapozhenko A. A., On the Number of Independent Sets in Graphs. Problems of theoretical cybernetics. Proceedings of XIII International Conference, Kazan, 27-31 may 2002, 85-93.
[22] Stepanov V. E., Phase transition in random graphs. Theory Probab. Applcs 15, 187-203.

Some Heuristic Analysis of Local Search Algorithms for SAT Problems

Osamu Watanabe*

Department of Mathematical and Computing Science,
Tokyo Institute of Technology, Tokyo 152-8552, Japan
watanabe@is.titech.ac.jp

Abstract. By using heuristic analysis proposed in [WSA03], we investigate the dynamical behavior of greedy local search algorithms for satisfiability (SAT) problems. We observe that the difference between hard and easy instances is relatively small while there are enough places to be improved locally, and that the difference becomes crucial when all such places are processed. We also show that a tabu search type restriction could be useful for improving the efficiency of greedy local search algorithms.

1 Introduction

Dynamical behavior of algorithms are usually very difficult to analyze; but understanding such dynamical behavior is important to see why and how algorithms work and to find out a way of improving algorithms. For example, for some algorithms or heuristics, while their excellent performance has been shown experimentally, it has been open to give theoretical justifications to such experimental results. One approach for discussing average case performance of such algorithms is to study their dynamical behavior under given input distributions.

For bridging a gap between experimental observations and theoretical analyses, a *heuristic approach* for analyzing algorithms has been proposed by several researchers; see, e.g., Barhel, Hartmann, and M. Weigt [Betal03], Semerjian and Monasson [SM03], and Watanabe et al. [WSA03]. In such approach, an average execution of a given algorithm is approximated "heuristically" by a much simpler randomized model, and then, some mathematical analysis is made on this simpler model. In this paper, we take this approach and analyze some local search algorithms for two satisfiability problems following the heuristic analysis proposed in [WSA03].

Our target problems are the standard 3-SAT problem and the 3-XOR-SAT problem. The 3-XOR-SAT problem is a variant of the 3-SAT problem, where each clause is the exclusive-or of three literals. The problem is essentially the same as solving a system of linear equations over mod 2; hence, the problem of finding a satisfying assignment is indeed polynomial-time solvable. On the other

* This work is supported in part by a Grant-in-Aid for Scientific Research on Priority Areas "Statical-Mechanical Approach to Probabilistic Information Processing" 2002-2005.

hand, the problem of finding a satisfying assignment closest to a given initial assignment is NP-hard; we consider this problem for the 3-XOR-SAT problem. (See the next section for the precise definition of the problems and the average scenario.)

In this paper, we investigate local search algorithms. Local search algorithms are simple yet effective for solving many problems. In fact, many constraint satisfaction problems can be solved *efficiently to some extent* on average by local search algorithms, and our target problems are such examples. We would like to know why and when local search algorithms work. Note also that local search algorithms are not the same; there are several variations and their algorithmic features may not be the same. Again we would like to know why some local search algorithm does/does not work. We discuss this question on the 3-XOR-SAT and 3-SAT problems.

It is well known [SS96] that a simple and efficient *greedy local search* algorithm solves 3-XOR-SAT quite well to some extent. Here by "greedy local search" we mean to make a local change (to a current solution) on the most problematic point. More precisely speaking, we define the "penalty" of (a current assignment to) a variable to be the number of unsatisfied clauses that the variable appears and make a local change on a variable with highest penalty. The performance of this greedy local search algorithm is checked easily by computer experiments. Such experiments show that there exists some threshold, and the algorithm yields a correct answer with high success probability if the badness of an initial assignment (i.e., the Hamming distance between the assignment and the closest sat. assignment) is within this threshold; otherwise, the algorithm fails very badly, i.e., almost 0 success probability. We would like to know why the threshold exists. It should be noted here that Sipser and Spielman [SS96] indeed proved, for some upper bound of initial assignments' badness, the algorithm yields a correct answer with high probability if the badness of an initial assignment is within this bound. It is, nevertheless, still open to understand the performance of the algorithm near the threshold, and for this, we would like to know the average and dynamical behavior of the algorithm.

For investigating average behavior of the algorithm, we take the approach proposed in [WSA03]. That is, we first define a relatively simple Markov process and show *experimentally* that it simulates the behavior of the algorithm on randomly generated inputs. Secondly we approximate the average state of this Markov process by a recurrence formula, and again show *experimentally* that the approximation is reasonable. Then by analyzing the obtained formula, we observe that the difference occurs when the algorithm runs out variables with high penalty. The success of the algorithm depends on whether an intermediate solution is close enough to the target solution at this point of the execution.

Next we consider the 3-SAT problem. For the 3-SAT problem, *random walk local search* algorithms have been investigated extensively, see, e.g., [SAT00]; in particular, the *walk-SAT* algorithm has been analyzed in depth by Barhel et al. [Betal03] and Semerjian and Monasson [SM03], which are comparable to our approach. On the other hand, as far as the author knows, no theoretical investigation has been made on greedy local search algorithms for 3-SAT. Thus, we focus on a greedy local search algorithm similar to the one for 3-XOR-SAT. It turned out that our first approximation step, i.e., simulation by a relatively simple Markov process, does not work in this case; simulation almost always gives a better convergence than actual executions. We think that this difference occurs because some variables are changed their values for several times and the independence of local updates, which we assume for the simulation, no longer holds. Thus, we introduce "flip once" restriction; that is, we modify the algorithm so that the local update is made only on untouched variables so long as some untouched and unsatisfiable variable exists, and the execution goes back to the normal one from the point that there is no untouched and unsatisfiable variable. This may be regarded as a "tabu search" type heuristics. For this modified algorithm, we can again observe that algorithm's behavior is quite close to the Markov process simulation, and the algorithm runs more efficiently. Though our experiments are limited, this is an evidence that the "flip once" restriction could be useful when using greedy local search algorithms.

2 Preliminaries

We state problems and algorithms discussed in this paper formally. Our target problems are the following 3-SAT problem and 3-XOR-SAT problem. Below by, e.g., $(3,d)$-CNF formula, we mean a 3CNF-formula such that each variable appears exactly d times in the formula.

(3,d)-SAT

Input: (3,d)-CNF formula F over n Boolean variables.
Output: A sat. assignment.

Closest Solution Search for (3,d)-XOR-SAT

Input: (3,d)-Parity formula F (like the one below) over n Boolean variables,

$$F \;=\; (\neg x_3 \oplus x_7 \oplus x_2) \wedge (x_1 \oplus \neg x_{12} \oplus \neg x_{61}) \wedge \cdots$$

and an assignment $\boldsymbol{a}$ to $x_1, ..., x_n$.
Output: A sat. assignment that is closest to $\boldsymbol{a}$.

Since we consider only $(3,d)$-type formulas, the *clause/var. ratio*, the ratio between the number of clauses and that of variables, is $d/3$. For the 3-SAT problem, it has been well-known, see, e.g., [Ach01, SAT00], that formulas become hard when this ratio is larger than 4.2. We will, however, consider in this paper local search algorithms with "quite small" running time, and for such algorithms, even $(3,9)$-formulas (their clause/var. ratio is 3) are hard. Note, on the other hand, the clause/var. ratio does not seem so essential for the XOR-SAT problem.

Next we define an average scenario that we will assume for our analysis. Let n be the number of variables. A *formula index sequence* is a sequence of sets, each of which is a set of three elements from $[n] \overset{\text{def}}{=} \{1, ..., n\}$, such that each $i \in [n]$ appears exactly 6 times. A formula index sequence may contain some set more

```
program LocalSearch($F$);
    $x_1$, ..., $x_n$ ← randomly chosen $\boldsymbol{a}$ in $\{0,1\}^n$;
    repeat the following MAXSTEP steps
        if $F$ is satisfied with x then output the current assignment and halt;
        choose one variable; — (∗)
        flip the value of the selected variable;
    output "failure";
program end.
```

Fig. 1. General scheme for local search algorithms for 3-SAT

than once. For generating one $(3,d)$-SAT formula F, we first choose one $(3,d)$-formula index sequence uniformly at random from all possible ones. Secondly, for each set (i,j,k) in the sequence, we create a disjunctive clause consisting of variables x_i, x_j, and x_k, by determining the sign of the three literals randomly so that $(0,0,...,0)$ is one of its satisfying assignments. Then F is simply the conjunction of these clauses. We generate $(3,d)$-XOR-SAT formulas essentially in the same way. In our average case analysis, we assume that formulas are generated in this way. For the 3-XOR-SAT problem, we also assume that an initial assignment is generated uniformly at random from those with Hamming distance pn from the planted solution $(0,...,0)$, for some parameter p.

Finally, we state our local search algorithms. Figure 1 is a general scheme of local search algorithms that we consider in this paper. This is for the 3-SAT problem; for the 3-XOR-SAT, we simply use a given $\boldsymbol{a}$ for the initial assignment for x.

An actual algorithm is obtained by specifying the time bound MAXSTEP and a way to choose "flip variable" at (∗). Here we consider quite small time bound, MAXSTEP $= 2pn$ for 3-XOR-SAT and MAXSTEP $= n/2$ for 3-SAT. Recall that for our 3-XOR-SAT problem we assume that an initial assignment differs from the planted solution by pn bits; hence, pn steps (i.e., pn flippings) are at least necessary to get a solution. Our choice of time bound is quite tight; nevertheless, it will be shown that the algorithms work well to some extent even under such strong requirements.

For choosing a flip variable at (∗), we consider two ways: (1) choose one (randomly one of them if there are many) that appears most often in unsatisfied clauses, and (2) choose one randomly from those appearing in a randomly chosen unsatisfied clause. We call the former *greedy choice* and the latter *random-walk choice*. In particular, we call an algorithm of Figure 1 using the greedy choice (resp., the random-walk choice) a greedy (resp., a random-walk) local search algorithm for 3SAT, and denote them as GreedySAT and RandomWalkSAT.

Roughly speaking, greedy local search is more efficient than random-walk local search; that is, greedy gets close to the solution faster than random-walk. On the other hand, it is more likely that greedy local search gets trapped by some local minimum. In fact, GreedySAT almost always fails by being trapped by some local minimum. One standard way to avoid this problem is to introduce "soft decision" when choosing a flip variable. We explain below a soft decision type local search algorithm. It turns out that the random-walk local search algorithm is also one special case of this algorithm. That is, both (soft decision type) greedy local search and random-walk local search are uniformly expressed by this algorithm. Thus, we will use this algorithm in the following discussion.

A soft decision type local search algorithm also follows the outline of Figure 1. The difference is a way to select a flip variable at $(*)$. Consider any iteration in the execution of the algorithm, and let $\boldsymbol{a}$ be the current assignment to variables in $\mathtt{x}$. We say that a variable $\mathtt{x}_i$ has a *penalty* k (under the current assignment $\boldsymbol{a}$) if $\mathtt{x}_i$ appears in k unsatisfied clauses. Recall that we assume that each variable appears 6 times in the formula; hence, penalty k must be between 0 to 6. Let W be a *weight function*, a function assigning weight to each penalty $k \in \{0, 1, ..., 6\}$. Now our selection of a flip variable is as follows: (1) choose a penalty value k with probability $P(k)$, and then (2) choose one variable uniformly at random from all variables with penalty k. Here $P(k)$ is the proportion of the weight of all penalty k variables in total weight; formally,

$$P(k) \overset{\text{def}}{=} \frac{W(k) \cdot (\#\text{ of penalty } \mathtt{k} \text{ var.s})}{\text{total weight}}.$$

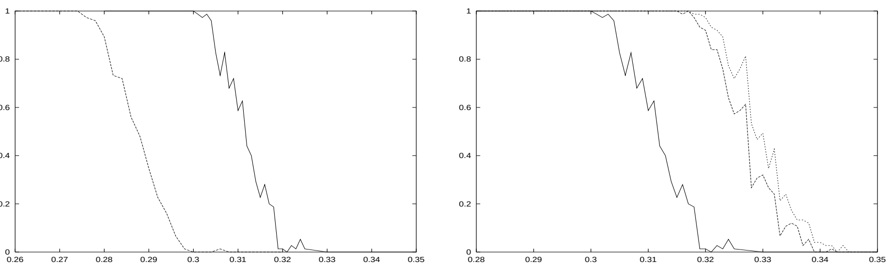
As one can easily expect, if $W(k) << W(k+1)$ holds for all k, $0 \le k \le 5$, then the execution of this algorithm is similar to that of the greedy local search algorithm `GreedySAT`. For example, we will use a weight function W defined by $W(0) = 0$ and $W(k) = 10^{k-1}$ for all k, $1 \le k \le 6$; we call the algorithm using this weight function a *soft decision greedy local search* algorithm and denote it as `SoftGreedySAT`. Also it is easy to see that with $W(k) = k$ for all k, $0 \le k \le 6$, this algorithm is exactly the same as the random-walk local search algorithm `RandomWalkSAT`. For the 3-XOR-SAT problem, algorithm `SoftGreedyXORSAT` is defined in the same way.

3 Greedy Local Search Algorithm for 3-XOR-SAT

We study a greedy local search algorithm for 3-XOR-SAT; more specifically, the soft decision greedy local search algorithm for 3-XOR-SAT — `SoftGreedyXORSAT` — defined in the previous section. (Unfortunately, random-walk local search does not work well; thus, we focus on the greedy algorithm.)

Let us first see that `SoftGreedyXORSAT` works well to some extent. Recall that p is the ratio of error assignments (w.r.t. the planted solution) in a given initial assignment. Figure 2 (a) shows the relation between p and the success probability of `SoftGreedyXORSAT` (left line) and `GreedyXORSAT` (right line); the success probability drops drastically at $p = 0.28 \sim 0.3$ for `SoftGreedyXORSAT` and $p = 0.3 \sim 0.32$ for `GreedyXORSAT`. We call this range as a *success threshold*, and we would like to know why the success prob. drops rapidly at this point.

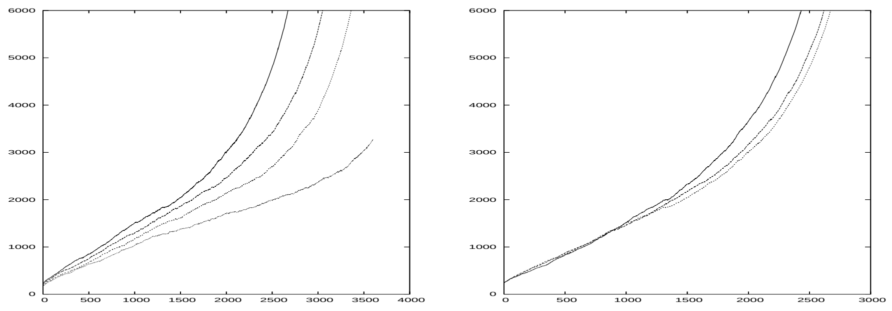
One might ask whether the success prob. improves by using larger time bound `MAXSTEP`. Figure 2 (b) shows this (for `GreedyXORSAT`); the success threshold moves to the right by using larger time bound `MAXSTEP`, but there seems to be some limit.



(a) `MAXSTEP` $= 2pn$ (b) `MAXSTEP` $= 2pn$, $5pn$, and $10pn$

The result of 5 runs, on 3 formulas $\times$ 5 initial assignments. Here and in the following experiments, the number n of variables is fixed to 6000.

Fig. 2. Success probability vs. p



(a) $p = 0.27$, 0.28, 0.29, 0.3 (b) three runs for the same input $p = 0.27$

Fig. 3. n_0 vs. step t

Recall that the penalty of a variable $\mathtt{x}$ is the number of clauses having $\mathtt{x}$ that is unsatisfied under a current assignment. Since each variable appears exactly 6 times in a formula, penalty must be between 0 to 6. For any k, $0 \leq k \leq 6$, let n_k denote the number of penalty k variables (under the assignment at a given point of the execution). We use $(n_0, ..., n_6)$ to express the status of the execution. In particular, when n_0 reaches to n, the total number of variables, then we know that the formula is satisfied with the assignment at this point[1] Figure 3 (a) shows how does n_0 get increased during the execution for four different choices of p before/after the success threshold. As shown in this figure, the execution for $p = 0.3$ fails to get a sat. assignment.

Remark on the Presentation of Experimental Results
In this paper, we will present our experimental results by showing some "typical" one instead of the average of several executions. This is because, in most

[1] A satisfying assignment may not be the desired solution, i.e., the closest sat. assignment. But it is quite unlikely that there is a sat. assignment other than the target solution $(0, ..., 0)$ within distance $2pn$ from a given initial assignment.

cases (more precisely, before or after the success threshold), difference among executions, initial assignments, and formulas is not so large. For example, Figure 3 (b) shows the result of three executions on the same input for $p = 0.27$; difference among initial assignments and/or formulas is more or less the same.

3.1 Heuristic Analysis (Step 1): Markov Process Simulation

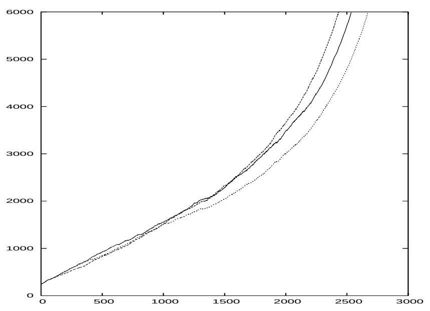
We would like to see, for example, how n_0 changes during the execution *on average*. The first step of our heuristic analysis is to simulate the algorithm's execution by a relatively simple Markov process. Clearly, the execution of `SoftGreedyXORSAT` is indeed a Markov process, where its state space is the set of assignments to n variables. But this state space with 2^n states is too large, and our first step is to simulate the algorithm by a Markov process with much smaller number of states, whose size is polynomially bounded by n.

As an example, we use a tuple $\boldsymbol{n} = (n_0, ..., n_6)$ to express the state of the algorithm during its execution, where n_k is, as defined above, the number of penalty k variables. Note that the information given by a tuple $\boldsymbol{n}$ is enough to simulate the random selection of the penalty k of a flip variable. In `SoftGreedyXORSAT`, k is randomly selected from $\{0, 1, ..., 6\}$ with probability $P(k)$, where

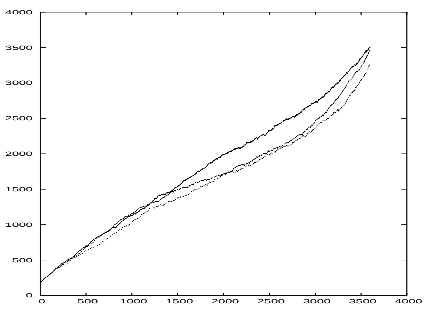
$$P(k) \; = \; \frac{W(k) \cdot n_k}{\sum_{i=0}^{6} W(i) \cdot n_i}. \qquad (1)$$

Hence, by using the current state $\boldsymbol{n} = (n_0, ..., n_6)$, the algorithm's execution can be simulated precisely; select k as the algorithm specifies, and then decrease n_k by one and increase n_{6-k} by one, thereby simulating the change of the flip variable.

For the simulation, however, we need to make further changes on $\boldsymbol{n}$ reflecting the change of the status of all *related* variables, variables appearing together with the flip variable in some clause. For example, in the real execution, suppose that a flip is made on $\mathtt{x}_3$, which makes a clause, say, C_{12} satisfiable, and let $\mathtt{x}_7$ be a variable in C_{12}; then the penalty of $\mathtt{x}_7$ gets decreased by one. Thus, if the



(a) $p = 0.27$



(b) $p = 0.3$

One simulation and two real executions

Fig. 4. n_0 vs. step t, simulation and execution penalty of $\mathtt{x}_7$ is 5, then n_5 gets decreased by one, and n_4 gets increased by one. Unfortunately, our information $\boldsymbol{n} = (n_0, ..., n_6)$ is insufficient for simulating such changes; we can specify penalty k, but we cannot specify the name of a flip variable, which is necessary to determine related variables.

Even though the names of related variables are unknown, we may still be able to guess them, in particular, their penalties. We may randomly choose k', the penalty of a variable appearing with a flip variable in each clause, with some appropriate probability. Unfortunately, $\boldsymbol{n}$ is too weak even for this purpose. But we found that a reasonable simulation is possible by using some similar but more complicated states, which we explain below.

For each variable $\mathtt{x}$, consider a tuple, e.g, $(+; --; -+; -+; ++; ++; ++)$ to express[2] its current configuration; the first $+$ means that $\mathtt{x}$ is not correctly assigned, and the next six $--$, $-+$, or $++$ express the status of related variables in six clauses containing $\mathtt{x}$. Note that there are $2\binom{6+2}{2} = 56$ possible configurations. That is, we classify variables into 56 groups and use $\boldsymbol{m} = (m_1, ..., m_{56})$ as a state, where m_i denotes now the number of variables with ith configuration (under a current assignment).

It is easy to see that the penalty of a variable is determined from its configuration; for example, a variable with configuration $(+; --; -+; -+; ++; ++; ++)$ has penalty 2. Thus, each n_k is computable from $\boldsymbol{m} = (m_1, ..., m_{56})$. Hence the selection of the penalty of a flip variable can be simulated as above. Furthermore, by a quite natural way, we can randomly choose the configuration (not only the penalty) of related variables. Then as shown in Figure 4, this Markov process can simulates the algorithm's execution quite well.

3.2 Heuristic Analysis (Step 2): Recurrence Formula

Assuming that our Markov process simulates the algorithm's execution well, we next derive a recurrence formula computing its "average state" at each step t. Since it is a Markov process, for each state $\boldsymbol{m}'$, the probability $\Pr[\boldsymbol{m}'|\boldsymbol{m}]$ is well-defined. Then the *average next state* of $\boldsymbol{m}$ is simply defined as $\widehat{\boldsymbol{m}} = \sum_{\boldsymbol{m}'} \boldsymbol{m}' \cdot \Pr[\boldsymbol{m}'|\boldsymbol{m}]$.

More specifically, for each configuration index i, $1 \le i \le 56$, a formula computing the ith element $\widehat{m}_i$ of $\widehat{\boldsymbol{m}}$ from $\boldsymbol{m}$ has the following form:

$$\widehat{m}_i \;=\; -P(i) + P(i') + \sum_j P(j)[\,-E(i|j) + \sum_{h \in N(i,j)} E(h|j)\,].$$

Here $P(i)$ denotes the probability that the ith configuration is chosen as a flip var.'s configuration, and i' denotes the configuration that moves to i by one flip. $E(i|j)$ is the expected number of var.s with the ith configuration that are related to some var. with the jth configuration, i.e., var.s with the ith configuration that share some clause with some var. with the jth configuration. $N(i,j)$ is the set of

[2] We ignore the order; for example, $+-$ and $-+$ are the same, and also $(+; --; --; --; ++; ++; ++)$ and $(+; --; ++; --; ++; --; ++)$ are considered as the same tuple.

configurations that are related to some var. x with the jth configuration and that change to the ith configuration when a flip is made on x. Note that $P(i)$ can be defined in the same way as (1); that is, $P(i) \overset{\text{def}}{=} W(k_i) \cdot m_i/\text{TotalWeight}$, where k_i is the penalty of the ith configuration. Though a slightly more complicated, $E(i|j)$ can be defined similarly. We use F to denote a function computing $\widehat{\boldsymbol{m}}$ from $\boldsymbol{m}$.

For a given parameter p and n, we can define the *average initial state* $\widehat{\boldsymbol{m}}^{(0)} = (\widehat{m}_1^{(0)}, ..., \widehat{m}_{56}^{(0)})$, where each $\widehat{m}_i^{(0)}$ is the average number of var.s with the ith configuration, when a formula and an initial assignment are generated randomly following our average scenario. Our second approximation step is to approximate $\widehat{\boldsymbol{m}}^{(t)}$, the *average state* after the tth step, by the following recurrence formula.

$$\widehat{\boldsymbol{m}}^{(t)} \approx F^t(\widehat{\boldsymbol{m}}^{(0)}),$$

where F^t means to t time applications of F. Again our experiments show[3] that this is a quite good approximation.

3.3 Observations and Claims

Now let us assume that our two step approximations are reasonable, as our experiments suggest. Then we can discuss our question by analyzing $F^t(\widehat{\boldsymbol{m}}_{(0)})$, which we call a *pseudo average state* at step t. We denote this pseudo average state $F^t(\widehat{\boldsymbol{m}}_{(0)})$ by $\widetilde{\boldsymbol{m}}^{(t)} = (\widetilde{\boldsymbol{m}}_1^{(t)}, ..., \widetilde{\boldsymbol{m}}_{56}^{(t)})$, and let $\widetilde{\boldsymbol{n}}^{(t)} = (\widetilde{\boldsymbol{n}}_0^{(t)}, ..., \widetilde{\boldsymbol{n}}_6^{(t)})$ be a tuple of average n_k computed from $\widetilde{\boldsymbol{m}}^{(t)}$.

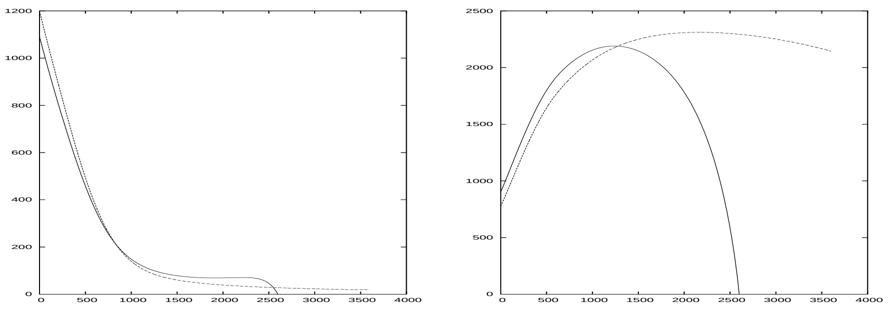
By comparing $\widetilde{\boldsymbol{n}}^{(t)}$ for $p = 0.27$ (before the threshold) and $p = 0.3$ (after the threshold), we noticed the following points:

1. For any $k \geq 4$, $\widetilde{\boldsymbol{n}}_k^{(t)}$ are almost the same between these p values (Figure 5 (a)).
2. Difference appears on $\widetilde{\boldsymbol{n}}_1^{(t)}$ at $t = 1500$, which is about the time when $\widetilde{\boldsymbol{n}}_4^{(t)}$ becomes small and its decreasing speed gets slower (e.g., Figure 5 (b)).

Notice that making a flip on any variable of penalty $k \geq 4$ improves the situation by reducing the total amount of penalties. So long as there are many such variables, the algorithm behaves more or less in the same way; on the other hand, the fate of the execution is determined when the most of such variables get corrected. It seems important that the state at this point is close enough to the solution or at least it starts getting closer to the solution.

We obtained a formula for pseudo average states. But, unfortunately, our formula is not so simple for us to handle it symbolically. There is, however, still some advantage of having such formula. One important merit is that we can now easily change (some of the) parameters, which may not be easy in the real execution or even in the simulation. For example, we can easily compute average

[3] For this approximation, we have some error bound [Wat05] and it is good for the simplest case; unfortunately, however, a good bound for the general case is still open, and in this paper we will rely on experimental justification.



(a) $\widetilde{\boldsymbol{n}}_4^{(t)}$ (for $p = 0.27$ and $p = 0.3$) (b) $\widetilde{\boldsymbol{n}}_1^{(t)}$ (for $p = 0.27$ and $p = 0.3$)

Fig. 5. Average number of var.s of each penalty

states for much larger n; this computation time is essentially independent from n. In fact, it is not so hard to see the following relation holds:

3. For any $c \geq 1$, let $\widetilde{\boldsymbol{M}}^{(t)}$ be a pseudo average computed for our Markov process with size parameter cn. Then we have $\widetilde{\boldsymbol{m}}^{(t)} \approx \widetilde{\boldsymbol{M}}^{(ct)}/c$.

Thus, if we accept our approximations, we can expect that the above observations hold not only on $n = 6000$ but also on any sufficiently large n. Furthermore, we may claim that the choice of weights $W(k)$ can be independent from n.

4 Greedy Local Search Algorithm for 3-SAT

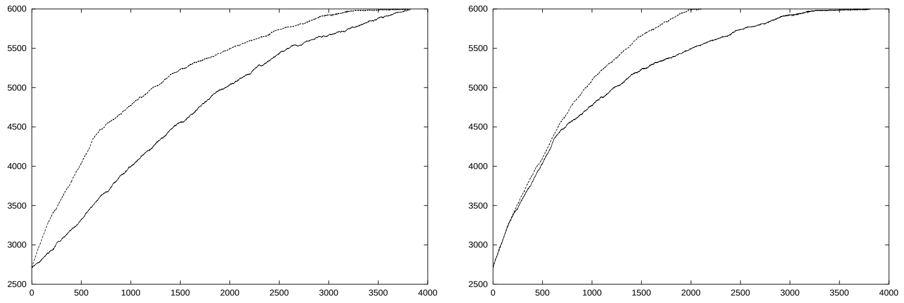
Here we study a greedy local search algorithm for 3-SAT. As mentioned in Introduction, the standard greedy algorithm usually gets trapped by a local minimum; but we can avoid this problem by the soft decision version — `SoftGreedySAT`, which we will study in this section.

First let us compare the executions of `SoftGreedySAT` and `RandomWalkSAT`. Figure 6 (a) shows typical executions of these algorithms. Here again we fix $n = 6000$ in our experiments. The graph shows how n_0, the number of sat. clauses grows[4]. The greedy seems a bit faster during earlier steps, but then it gets slower and it obtains the satisfying assignment almost at the same step.

Now we apply again our first approximation step, i.e., simulation by a Markov process. Though slightly different, we can take essentially the same approach and use a tuple of numbers, each of which denotes the number of var.s of a certain configuration. Interestingly, however, this same approach does not work. Figure 6 (b) shows the comparison between a real execution and a simulation by our relatively simple Markov process. In the simulation, the value of n_0 is improved slightly faster; it is also noticed that the improvement is smoother.

Note that in 3-SAT there are not so many variables with high penalty; initially, most variables have penalty ≤ 3 and there are less than 200 var.s with

[4] Here again some sat. assignment should be obtained when n_0 reaches to n; but for 3-SAT, such sat. assignments are usually different from the planted solution.



(a) `SoftGreedySAT` (left line) vs. `RandomWalkSAT` (right line)

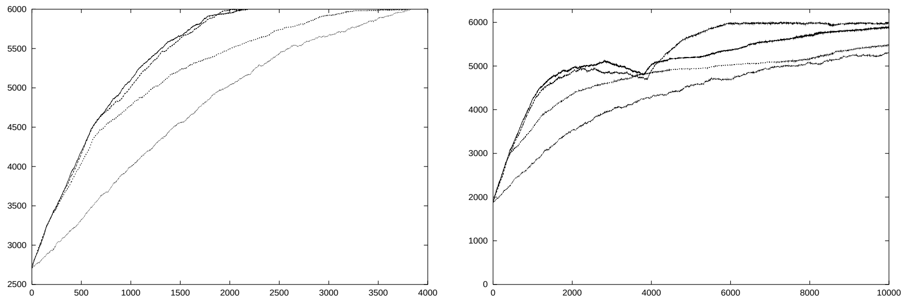
(b) `SoftGreedySAT` (right line) vs. simulation (left line)

Fig. 6. n_0 vs. step t

penalty ≥ 4, where $n = 6000$. (*Cf.* A typical 3-XOR-SAT instance has about 1500 var.s with penalty ≥ 4.) Thus, we can expect that the algorithm fixes these high penalty variables soon. Then some variables may be chosen as a flip variable for several times. Due to this, the independence of flip variables is destroyed, which prevents the execution follows the simulation where complete independence is assumed.

From this consideration, we introduce the following additional restriction.

Flip Once Rule:
When choosing a flip variable, give a priority to variables that have not been chosen before. More specifically, so long as there are some untouched variable with penalty > 0, such variable is chosen first. Then at the point when no such variable exists, the execution is switched back to the standard greedy.



(a) (3,6)-SAT

(b) (3,9)-SAT

`RandomWalkSAT`, `SoftGreedySAT`, Flip Once `SoftGreedySAT`, and a simulation by our simple Markov process

Fig. 7. n_0 vs. step t

Then as shown in Figure 7 (a), the execution of the algorithm becomes quite close to the one defined by the Markov process. Furthermore, this improves the efficiency of `SoftGreedySAT`; the running time is reduced almost half of the original. Intuitively, a Markov process gives an ideal execution; thus, by removing an obstacle that prevents actual executions to follow the execution of an appropriately defined Markov process, we may be able to improve the efficiency of greedy algorithms.

Note, however, this improvement has some limitation. The same trick does not work for the (3,9)-SAT problem (Figure 7 (b)). Although the execution gets close to the one by the Markov process, this does not lead to any improvement. This is because the greedy search cannot reach to an appropriate assignment for getting a solution when it runs out untouched and high penalty variables.

References

[Ach01] D. Achlioptas, Lower bounds for random 3-SAT via differential equations, *Theoret. Comput. Sci.* 265, 159–185, 2001.

[Betal03] W. Barthel, A. Hartmann, and M. Weigt, Solving satisfiability by fluctuations: The dynamics of stochastic local search algorithms, *Physical Review E*, 67, 066104, 2003.

[SAT00] Special Issue “SAT-2000”, *J. of Automated Reasoning* 24(1-2), 2000.

[SM03] G. Semerjian and R. Monasson, Relaxation and metastability in a local search procedure for the random satisfiability problem, *Physical Review E*, 67, 066103, 2003.

[SS96] M. Sipser and D. Spielman, Expander codes, *IEEE Trans. on Information Theory*, 42(6), 1710–1719, 1996.

[WSA03] O. Watanabe, T. Sawai, and H. Takahashi, Analysis of a randomized local search algorithm for LDPCC decoding problem, in *Proc. SAGA'03*, Lecture Notes in Comp. Sci. 2827, 50–60, 2003.

[Wat05] O. Watanabe, Pseudo expectation: A tool for analyzing local search algorithms, *Progress of Theoret. Physics Supplement*, No.157, 338–344, 2005. (A detail version is available: Research Report C-198, Dept. of Mathematical and Comput. Sci., Tokyo Inst. of Tech.)

Clustering in Stochastic Asynchronous Algorithms for Distributed Simulations

Anatoli Manita[1] and François Simonot[2]

[1] Faculty of Mathematics and Mechanics, Moscow State University, 119992 Moscow, Russia
manita@mech.math.msu.su
[2] IECN, Université Henri Poincaré Nancy I, Esstin, 2, Rue J. Lamour, 54500 Vandoeuvre, France
francois.simonot@esstin.uhp-nancy.fr

Abstract. We consider a cascade model of N different processors performing a distributed parallel simulation. The main goal of the study is to show that the long-time dynamics of the system have a cluster behaviour. To attack this problem we combine two methods: stochastic comparison and Foster–Lyapunov functions.

1 On Probabilistic Models of Parallel Simulations

The present paper contains the probabilistic analysis of some mathematical model of asynchronous algorithm for parallel simulation. For the detailed discussion of synchronization issues in parallel and distributed algorithms we refer to [1,11]. Here we give only a brief description of the problem. In large-scale parallel computation it is necessary to coordinate the activities of different processors which are working together on some common task. Usually such coordination is implemented by using a so-called message-passing system. This means that a processor shares data with other processors by sending timestamped messages. Between sending or receiving the messages the processors work independently. It can happen that till the moment of receiving of a message some processor can proceed farther in performing its program than the value of timestamp indicated in this newly received message; in this case the state of the processor should be *rolled back* to the indicated value. It is clear that due to possible rollbacks the mean speed of a given processor in the computing network will be lower than its proper speed. One of the most important performance characteristics of the system is the progress of the computing network on large time intervals.

Probabilistic models for such system are studied for already twenty years. From the probabilistic point of view these models consist of many relatively independent components which synchronize from time to time their states according to some special algorithm. The detailed review of all existing publications is out of range of this paper. We would like to mention only that the bibliography on this subject consists mostly of two groups of papers. The first group of publication [2,6,9,10,12,13] is devoted to the case of two processors. The paper [2] is of special interest since it contains an exhaustive basic analysis of the two-dimensional model and had a big influence on further researches. The case of many processors is studied in [3,4,5,8,9,14,15]. The important difference of such models from the two-dimensional case is that in realistic models with more than two processors one message can provoke a multiple rollback of a chain of processors. Since the multi-dimensional model is much more complicated for a rigorous study, in the above papers the authors dealt with the set of identical processors and their mathematical results are contained in preparatory sections before large numerical simulations.

It should be noted also that probabilistic models with synchronization mechanism are interesting also for modeling database systems (see for example, [1]). Moreover, now the synchronization-like interactions are considered as well in the framework of interaction particle systems [16,17,18].

The model considered in the present paper is of special interest for the following reasons. We deal with a nonhomogeneous model which consists of several different processors. We consider the case of message-passing topology different from the topology of complete graph which was considered in all previous papers. Our main interest is the cascade model which pressuposes a subordination between processors. We put forward a conjecture on the cluster behavior of the system: processors can be divided into separated groups which are asymptotically independent and have their own proper performance characteristics. Our main goal is to justify this conjecture. It should be pointed out that in the case of the complete graph topology the cluster decomposition into groups is degenerated and, thus, is not interesting.

We describe our model in terms of multi-dimensional continuous time Markov process. To get asymptotical performance characteristics of the model we combine two probabilistic methods: stochastic comparison and Foster–Lyapunov functions.

The paper is organized as follows. In Sect. 2 we introduce a general continuous time Markov model and define a cascade model as a special subclass of the general model. In Sect. 3 we pass to the embedded Markov chain. The main problem is now to study a long-time behavior of Markov chain with highly nonhomogeneous transition probabilities. To do this we consider relative coordinates and find groups of processors whose evolution is ergodic (it convergences to a steady state) in these relative coordinates. In our opinion the method of Foster-Lyapunov functions seems to be the only one to prove the stability in the relative coordinates for the Markov chain under consideration. First of all in Sect. 5 we begin from the case of two processors ($N = 2$) and the analysis here is rather simple and similar to [2]. In the study of the three-dimensional case (Sect. 7) the main point is the proof of ergodicity. We propose an explicit construction of some nonsmooth Foster-Lyapunov function. Our construction is rather nontrivial as it can be seen by comparing it with already existing explicit examples of Lyapunov functions (see [7]). All this analysis brings us to some conclusions presented in Sect. 8. This section contains the decomposition into groups (clusters) in the case of the cascade model with any number of processors N and our main Conjecture 5. We show that the proof of this conjecture could be related with progress in explicit construction of multi-dimensional Foster-Lyapunov functions. Analysis of random walks in $\mathbf{Z}_+^n$ (which was done in [7]) shows that, in general, this technical problem may be very difficult. In the next papers we hope to overcome these difficulties by using specific features of our concrete Markov processes.

2 Description of Continuous Time Model

2.1 General Model

We present here some mathematical model for parallel computations. There are N computing units (processors) working together on some common task. The state of a processor k is described by an integer variable $x_k \in \mathbf{Z}$ which is called a local (or inner) time of the processor k and has a meaning of amount of job done by the processor k up to the given time moment.

Assume that the state $(x_1, x_2, \ldots, x_N)$ of the system evolves in the continuous time $t \in \mathbf{R}_+$. Any change of the state is possible only at some special random time instants. Namely, with any processor k we associate a Poissonian flow $\Pi^k = \left\{0 = \sigma_0^k < \sigma_1^k < \cdots < \sigma_n^k < \cdots\right\}$ with intensity λ_k and with a pair (k, l) of processors we associate a Poissonian flow $\Pi^{kl} = \left\{0 = \sigma_0^{kl} < \sigma_1^{kl} < \cdots < \sigma_n^{kl} < \cdots\right\}$ with intensity β_{kl}. This means, for example, that $\left\{\sigma_n^k - \sigma_{n-1}^k\right\}_{n=1}^{\infty}$ is a sequence of independent exponentially distributed random variables with mean λ_k^{-1}: $\forall n = 1, 2, \ldots \quad \mathsf{P}\left\{\sigma_n^k - \sigma_{n-1}^k > s\right\} = \exp\left(-\lambda_k s\right)$, and similarly for the flows Π^{kl}. We also assume that all these flows Π^k and Π^{kl} are mutually independent.

Let us now define a stochastic process $\left(X(t) = (x_1(t), \ldots, x_N(t)), t \in \mathbf{R}_+\right)$ on the state space $\mathbf{Z}^N$ according to the following rules:

1) At time instants σ_n^k the processor k increases its local time x_k by 1:

$$x_k(\sigma_n^k + 0) = x_k(\sigma_n^k) + 1 \ .$$

2) There is an exchange of information between different processors. At time instant σ_i^{kl} the processor k sends a message $m_{kl}^{(x_k)}$ to the processor l. We assume that the messages reach their destination immediately. A message $m_{kl}^{(x_k)}$ coming to node l from node k contains an information about local time $x_k(\sigma_i^{kl}) = x_k$ of the sender k. *If* at the time instant σ_i^{kl} (when the message $m_{kl}^{(x_k)}$ arrives to the node l) we have $x_l(\sigma_i^{kl}) > x_k(\sigma_i^{kl})$ *then* the local time x_l rolls back to the value x_k: $x_l(\sigma_i^{kl} + 0) = x_k(\sigma_i^{kl})$. Moreover, if the processor l rolls back, then all messages sent by the processor l during the time interval $\mathcal{I} = (\theta_l(x_k, \sigma_i^{kl}), \sigma_i^{kl})$, where $\theta_l(x, u) := \max\left\{s \le u : \, x_l(s) = x, \, x_l(s+0) = x+1\right\}, \sigma_i^{kl})$,should be eliminated This may generate a cascading rollback of local times for some subset of processors. For example, assume that there is a processor q which received a message $m_{lq}^{(x_l')}$ at some time instant $s' \in \mathcal{I}$ and $x_q(\sigma_i^{kl}) > x_l(s') = x_l'$. Then the local clock of q should be rolled back to the value $x_l(s')$: $x_q(\sigma_i^{kl} + 0) = x_l(s')$ and, moreover, all messages sent by q during the interval $\mathcal{I} = (\theta_q(x_l(s'), \sigma_i^{kl}), \sigma_i^{kl})$

should be deleted, and so on. Hence, at time instant σ_i^{kl} a message from k to l can provoke a multiple rollback of processor $l, q, \ldots$ in the system.

2.2 Cascade Model

From now we shall consider the following special subclass of the above general model. A chain of processors $1, 2, \ldots, N$ is called a *cascade* if any processor j can send a message only to its right neighbour $j+1$. Hence, the processor N does not send any message and the processor 1 does not receive any message. In other words, $\beta_{ij} \neq 0 \Leftrightarrow (j = i+1)$. A message sent from j to $j+1$ can provoke a cascading rollback of processors $j+2, \ldots$. Recall that all above time intervals are exponentially distributed and assumed to be independent. Obviously, the stochastic process $X_c^{(N)}(t) = (\, x_1(t), \ldots, x_N(t)\,)$ is Markovian. A very important property is that any "truncated" marginal process $X_c^{(N_1)}(t) = (\, x_1(t), \ldots, x_{N_1}(t)\,)$, $N_1 \le N$, is also Markovian.

Assume that for any j the following limit

$$v_j^* = \lim_{t \to +\infty} \frac{x_j(t)}{t} \qquad \text{(in probability)} \tag{1}$$

exists. Then the numbers v_j^*, $j = 1, \ldots, N$, characterize *performance* of the model. In this paper we propose an approach for proving the existence of these limits. We present here not only rigorous results in the final form (Theorems 3, 4 and 7) but also some conjectures (Sect. 8) which should be considered as starting points for future studies.

Note that if we uniformly transform the absolute time scale $t = cs$, where $c > 0$ is a constant and s is a new absolute time scale, the performance characteristics (1) will not change.

3 Definition of the Discrete Time Cascade Model

Consider a sequence $0 = \tau_0 < \tau_1 < \tau_2 < \cdots < \cdots$ of time moments when changes of local time at nodes may happen (we mean local time updates and moments of sending of messages). It is clear that $\{\tau_{r+1} - \tau_r\}_{r=0}^{\infty}$ is a sequence of independent identically distributed r.v. having exponential distribution with parameter

$$Z = \sum_{i=1}^{N} \lambda_i + \sum_{i=1}^{N-1} \beta_{i,i+1} \ .$$

Observing the continuous time Markov process $(x_1(t), \ldots, x_N(t))$ at epochs τ_n we get the so-called embedded discrete time Markov chain $\{X(n), n = 0, 1, \ldots\}$ with state space $\mathbf{Z}_+^N$. In the sequel we will be interested in the long-time behaviour of the chain $\{X(n), n = 0, 1, \ldots\}$.

Transition Probabilities. In the MC $\{X(n), n = 0, 1, \ldots\}$ there are transitions produced by the free dynamics and transitions generated by rollbacks. By the free dynamics we mean updating of local times

$$P\{X(n+1) = x + \mathbf{e}_j \,|\, X(n) = x\} = \lambda_j Z^{-1}, \quad j = 1, \dots, N\,,$$

where $\mathbf{e}_j = (0, \dots, 0, \underset{j}{1}, 0, \dots, 0)$. It is easy to see that if a state $x = (x_1, \dots, x_N)$ is such that for some j $x_j < x_{j+1}$ then a message sent from j to $j+1$ produces a transition of the following form

$$(x_1, \dots, x_j, x_{j+1}, \dots, x_l, x_{l+1}, \dots, x_N) \to (x_1, \dots, x_j, w_{j+1}, \dots, w_l, x_{l+1}, \dots, x_N) \qquad (2)$$

with probability

$$Z^{-1}\beta_{j,j+1} \prod_{q=j+1}^{l-1} p(w_q, x_q; w_{q+1}, x_{q+1}) \times (1 - b_l)^{\min(x_l, x_{l+1}-1) - w_l + 1}\,, \qquad (3)$$

where

– sequence $(w_{j+1}, \dots, w_l)$ is admissible in the following sense:

$$j < l \le N, \qquad w_{j+1} = x_j \qquad w_q \le w_{q+1} \le \min\left(x_q, x_{q+1} - 1\right), (j < q < l)$$

– $p(w_q, x_q; w_{q+1}, x_{q+1}) = b_q \left(1 - b_q\right)^{w_{q+1} - w_q}$
– $b_q = \dfrac{\lambda_q}{\lambda_q + \beta_{q,q+1}}$, $q < N$.

Here b_q is the probability of an event that processor q in state x_q sends at least one message to $q+1$ before updating its state $x_q \to x_q + 1$. For $q = N$ we put $b_N = 0$. So in the case $l = N$ the probability (3) takes the form

$$Z^{-1}\beta_{j,j+1} \prod_{q=j+1}^{N-1} p(w_q, x_q; w_{q+1}, x_{q+1}) \;.$$

Relative Coordinates. Note that the first processor $x_1(t)$ evolves independently of other processors. It is useful to introduce new process $Y_c(t) = (y_2(t), \dots, y_N(t)) \in \mathbf{Z}^{N-1}$ in relative coordinates as viewing by an observer sitting at the point $x_1(t)$:

$$y_j(t) := x_j(t) - x_1(t), \quad j = 2, \dots, N \;.$$

In a similar way we define $Y(n) = Y_c(\tau_n)$, $n = 0, 1, \dots$. The free dynamics produce the following transitions of $Y(n)$:

$$P\{Y(n+1) = y + \mathbf{e}_j \,|\, Y(n) = y\} = \lambda_j Z^{-1}, \quad j = 2, \dots, N\,, \qquad (4)$$

$$P\left\{Y(n+1) = y - \textstyle\sum_{j=2}^{N} \mathbf{e}_j \,|\, Y(n) = y\right\} = \lambda_1 Z^{-1} \;. \qquad (5)$$

Since rollback does not affect on the first processor the corresponding transitions have the same form and the same probabilities as (2) and (3).

4 Stochastic Monotonicity

All statements of this section are valid for the both Markov processes $X_c^{(N)}(t)$, $t \in \mathbf{R}_+$, and $X(n)$, $n \in \mathbf{Z}_+$. For the sake of brevity we give here results only for the continuous time model $X_c^{(N)}(t)$.

Theorem 1. *Consider two cascade models (say $X_{c,1}^{(n)}(t)$ and $X_{c,2}^{(n)}(t)$) with processors $1, 2, \ldots, n$ and parameters $\lambda_1, \ldots, \lambda_n$ and $\beta_{12}^{(1)}, \beta_{23}^{(1)}, \ldots, \beta_{n-1,n}^{(1)}$ for the first model $X_{c,1}^{(n)}(t)$ and parameters $\lambda_1, \ldots, \lambda_n$ and $\beta_{12}^{(2)}, \beta_{23}^{(2)}, \ldots, \beta_{n-1,n}^{(2)}$ for the second model $X_{c,2}^{(n)}(t)$. Assume that $\beta_{i,i+1}^{(1)} \le \beta_{i,i+1}^{(2)} \quad \forall i$. Then $X_{c,1}^{(n)}$ is stochastically larger than $X_{c,2}^{(n)}$, that is: if $X_{c,1}^{(n)}(0) = X_{c,2}^{(n)}(0)$ then $X_{c,1}^{(n)}(t) \ge_{\mathrm{st}} X_{c,2}^{(n)}(t)$ for any t.* [1]

Proof may be given by a standard explicit coupling construction of the processes $X_{c,1}^{(n)}(t)$ and $X_{c,2}^{(n)}(t)$ on the same probability space. The following fact should be used: a Poisson flow with intensity $\beta_{12}^{(1)}$ can be obtained from a Poisson flow with intensity $\beta_{12}^{(2)}$ in which any point (independently from other) is killed with probability $1 - \beta_{12}^{(1)}/\beta_{12}^{(2)}$.

Corollary 2 (Solid barriers). *Fix some $1 \le r_1 < r_2 < \cdots < r_b < n$ and consider two cascade models: $X_{c,1}^{(n)}(t)$ with parameters $\Big(\lambda_1, \ldots, \lambda_n\,;\, \beta_{12}^{(1)}, \ldots, \beta_{n-1,n}^{(1)}\Big)$ and $X_{c,2}^{(n)}(t)$ with parameters $\Big(\lambda_1, \ldots, \lambda_n\,;\, \beta_{12}^{(2)}, \ldots, \beta_{n-1,n}^{(2)}\Big)$, where*

$$\beta_{i,i+1}^{(2)} = \beta_{i,i+1}^{(1)} \quad \forall i \notin \{r_1, \ldots, r_b\}\,, \qquad \beta_{i,i+1}^{(2)} = 0 \quad \forall i \in \{r_1, \ldots, r_b\} \quad .$$

We can say that the model $X_{c,2}^{(n)}(t)$ differs from the model $X_{c,1}^{(n)}(t)$ by the presence of b solid barriers between processors r_1 and $r_1 + 1$, ..., r_b and $r_b + 1$. Then by Theorem 1 we have that $X_{c,1}^{(n)}(t) \le_{\mathrm{st}} X_{c,2}^{(n)}(t)$.

5 Case $N = 2$

We start with the Markov chain $X_c^{(2)}(t)$. Since processor 1 works independently, it is enough to consider the Markov chain $Y_c^{(2)}(t) = x_2(t) - x_1(t)$.

Taking in mind the remark at the end of Subsect. 2.2, for brevity of notation let us rescale absolute time in such a way that $Z = 1$. Then the Markov chain $Y(n)$ has the following transition probabilities

$$p_{i,i+1} = \lambda_2, \quad p_{i,i-1} = \lambda_1, \quad p_{i,0} = \beta_{12}\ (i \ge 0), \quad p_{i,i} = \beta_{12}\ (i < 0)$$

and $p_{i,j} = 0$ for any another pair i, j .

[1] It means that there exists a *coupling* $\Big(\widetilde{X}_1^{(n)}(t,\omega), \widetilde{X}_2^{(n)}(t,\omega)\Big)$of stochastic processes $X_1^{(n)}(t)$ and $X_2^{(n)}(t)$ such that $P\Big\{\omega : \widetilde{X}_1^{(n)}(t,\omega) \ge \widetilde{X}_2^{(n)}(t,\omega)\ \forall t\Big\} = 1$. If $w, z \in \mathbf{R}^n$ we say $w \ge z$ if $w_i \ge z_i$ for all $i = 1, \ldots, n$ (partial order).

Theorem 3. *If $\lambda_1 < \lambda_2$ then the Markov chain $Y(n)$ is ergodic and we have $v_1^* = v_2^* = \lambda_1$. If $\lambda_1 > \lambda_2$ then the Markov chain $Y(n)$ is transient and we have $v_1^* = \lambda_1$, $v_2^* = \lambda_2$.*

Proof. The Markov chain $Y(n)$ is one-dimensional and its analysis is quite easy. To establish ergodicity under assumption $\lambda_1 < \lambda_2$ we use the Foster-Lyapunov criterion (Theorem 8, see Appendix) with test function $f(y) = |y|$, $y \in \mathbf{Z}$. This implies that $x_2(t) - x_1(t)$ has a limit in distribution as $t \to \infty$. Recall that $x_1(t)$ is a Poissonian process hence the limit $v_1^* = \lim_t t^{-1} x_1(t) = \lambda_1$ exists (in probability). It follows from this that $v_2^* = \lim_t t^{-1} x_2(t) = \lambda_1$.

Under assumption $\lambda_1 > \lambda_2$ we get transience by choosing the function $f(y) = \min(e^{\delta y}, 1)$, $y \in \mathbf{Z}$, where we fix sufficiently small $\delta > 0$, and applying Theorem 9 from Appendix. Therefore any trajectory of $Y(n)$ spends a finite time in any prefixed domain $\{y \geq C\}$ entailing $\lim_{t\to\infty} x_2(t) - x_1(t) = -\infty$ (a.s.). It means that after some time, the messages from 1 to 2 can not produce roll-backs anymore, so $x_1(t)$ and $x_2(t)$ become asymptotically independent and hence $v_2^* = \lim_t t^{-1} x_2(t) = \lambda_2$.

6 Case $N = 3$

Theorem 4. *Four situations are possible.*

1. *If $\lambda_1 < \min(\lambda_2, \lambda_3)$ then $v_1^* = v_2^* = v_3^* = \lambda_1$.*
2. *If $\lambda_2 > \lambda_1 > \lambda_3$ then $v_1^* = v_2^* = \lambda_1$, $v_3^* = \lambda_3$.*
3. *If $\lambda_2 < \min(\lambda_1, \lambda_3)$ then $v_1^* = \lambda_1$, $v_2^* = v_3^* = \lambda_2$.*
4. *If $\lambda_1 > \lambda_2 > \lambda_3$ then $v_1^* = \lambda_1$, $v_2^* = \lambda_2$, $v_3^* = \lambda_3$.*

Items 2, 3 and 4 can be reduced in some sense to the results of the case $N = 2$ (see Theorem 3). We prove them in the current section. Proof of the item 1 is much more intricate and relies heavily on the construction of an adequate Lyapunov function needing lengthy developments deferred to Sect. 7.

Proof of Theorem 4 (items 2–4). We start from the item 2: $\lambda_2 > \lambda_1 > \lambda_3$. Since the first two processors are governed by the Markov chain $X_c^{(2)}(t)$ and do not depend on the state of processor 3 we apply Theorem 3 and conclude that $X_c^{(2)}(t)$ is ergodic and $v_1^* = v_2^* = \lambda_1$.

Let us compare the following two cascade models

$$X_c^{(3)}(t): \quad 1 \xrightarrow{\beta_{1,2}} 2 \xrightarrow{\beta_{2,3}} 3 \qquad \text{and} \qquad X_{c,2}^{(3)}(t): \quad 1 \xrightarrow{\beta_{1,2}} 2 \xrightarrow{0} 3$$

(parameters λ_1, λ_2 and λ_3 are the same for the both models $X_c^{(3)}(t)$ and $X_{c,2}^{(3)}(t)$). In the model $X_{c,2}^{(3)}$ the groups of processors $\{1,2\}$ and $\{3\}$ evolve independently. Evidently, an asymptotic speed of processor 3 in the model $X_{c,2}^{(3)}$ exists and is equal to λ_3. By Corollary 2 $X_c^{(3)}(t) \leq_{\text{st}} X_{c,2}^{(3)}(t)$. Hence in the model $X_c^{(3)}$ an asymptotic speed of the processor 3 is *not greater* than λ_3. Let us show now that the inferior limit $\liminf_{t\to+\infty} t^{-1}x_3(t)$ *can not be less* than λ_3. Indeed, since $\lambda_3 < \lambda_1$ we conclude that there exists a random time moment τ_0 depending on $X_c^{(3)}(0)$ such that for all $t \geq \tau_0$ the both coordinates $x_1(t)$ and $x_2(t)$ of the process $X_c^{(3)}(t)$ will be greater than its third coordinate $x_3(t)$. So after the time τ_0 a message from 2 to 3 can not roll back anymore the processor 3. We see that with probability 1 the processor 3 gets only a finite number of rollbacks from the processor 2 on the time interval $[0, +\infty)$. Hence the asymptotic speed of the third processor in the cascade system $X_c^{(3)}$ coincides with its proper asymptotic speed which is equal to λ_3.

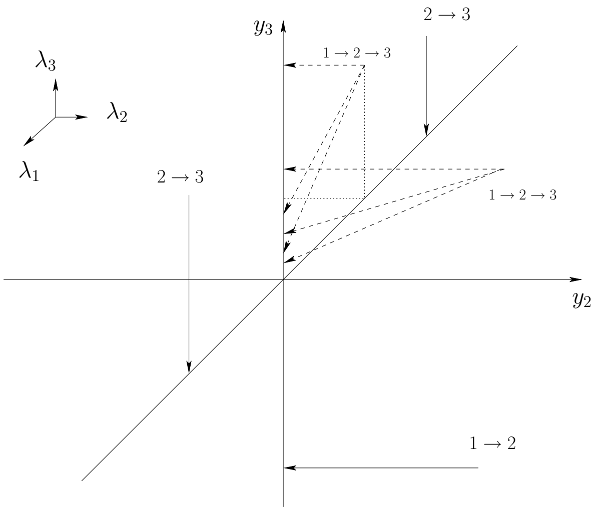
Items 3 and 4 can be considered in a similar way. Note that the item 3 consists of two subcases: $\lambda_1 > \lambda_3 > \lambda_2$ and $\lambda_3 > \lambda_1 > \lambda_2$. We omit details.

7 Explicit Construction of Lyapunov Function

In this section we prove the item 1 of Theorem 4. Recall that our key assumption here is $\lambda_1 < \lambda_2$, $\lambda_1 < \lambda_3$. The main idea is to prove that the Markov chain $Y(n)$ is ergodic. To do this we apply the Foster-Lyapunov criterion (see Theorem 8 in Appendix). As in the case of Theorem 3 ergodicity of $Y(n)$ implies that $v_j^* = \lambda_1$, $j = 1, 2, 3$.

7.1 Transition Probabilities

Consider the embedded Markov chain $Y(n)$. A stochastic dynamics produced by this Markov chain consists of two components: transitions generated by the free dynamics and transitions generated by rollbacks. For each transition probability $p_{\alpha\beta}$, $\alpha \neq \beta$, we have the following representation: $p_{\alpha\beta} = s_{\alpha\beta} + r_{\alpha\beta}$, where $s_{\alpha\beta} \geq 0$ corresponds to a transition $\alpha \to \beta$ which occurs due to the free dynamics and $r_{\alpha\beta} \geq 0$ corresponds to a rollback transition $\alpha \to \beta$.



Taking into account the remark at the end of Subsect. 2.2, without loss of generality we assume that the time is rescaled in such way that $Z = 1$. This slightly simplifies notation for transition probabilities. For example, the free dynamics transitions (4)–(5) are equal to λ_2, λ_3 and λ_1 correspondingly. On the above figure we show all non-zero transitions $\alpha \to \beta$, $(\alpha \neq \beta)$. It is true, of course, that $p_{\alpha\alpha} = 1 - \sum_{\beta\neq\alpha} p_{\alpha\beta}$, but it is useless to put this information on the picture. Below we give the explicit form of rollback transition probabilities:

$$1 \to 2: \qquad r_{yz} = \beta_{12} \qquad \text{for } 0 < y_2$$

$$2 \to 3: \qquad r_{yz} = \beta_{23} \qquad \text{for } y_2 < y_3$$

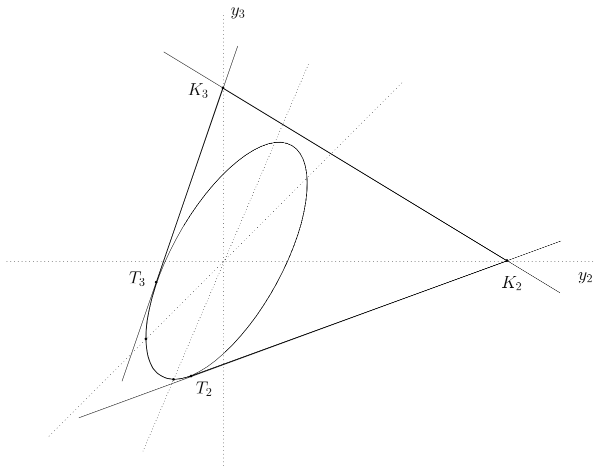
$$1 \to 2 \to 3: \qquad r_{yz} = \begin{cases} \beta_{12}\,(1-b_2)^{z_3}\, b_2, & z_3 < y_3 \\ \beta_{12}\,(1-b_2)^{y_3}, & z_3 = y_3 \end{cases} \qquad \text{for } 0 < y_3 \leq y_2$$

$$1 \to 2 \to 3: \qquad r_{yz} = \begin{cases} \beta_{12}\,(1-b_2)^{z_3}\, b_2, & z_3 \leq y_2 \\ \beta_{12}\,(1-b_2)^{y_2+1}, & z_3 = y_3 \end{cases} \qquad \text{for } 0 < y_2 < y_3$$

where $y = (y_2, y_3)$, $z = (z_2, z_3)$.

7.2 Contour of Level 1

Fix some $a > 0$ and $b > 0$. In the plane Oy_2y_3 consider the ellipse $e(y_2, y_3) = ay_2^2 + b\,(y_2 - y_3)^2 = 1$ and draw a tangent line to it with normal vector $(-\Delta, 1)$. Evidently, there exist two tangent lines with the same normal vector $(-\Delta, 1)$. If $\Delta > 0$ is sufficiently large then one of this tangent line touches the ellipse at some point T_3 of the domain $y_2 < 0$, $y_3 < 0$. Take a segment on this line from the point T_3 to a point $K_3 = (0, u_3)$ of intersection with coordinate axis Oy_3. Now let us draw tangent lines to the ellipse corresponding to a normal vector $(1, -\Delta)$. If $\Delta > 0$ is sufficiently large, then one of these lines touches the ellipse at some point T_2 of the domain $y_3 < 0$. Let us take this tangent line and fix a segment on it from the point T_2 to a point $K_2 = (u_2, 0)$ of intersection with coordinate axis Oy_2. It is evident that $[K_2K_3] = \mathbf{R}_+^2 \cap \{(y_2, y_3) : y_2/u_2 + y_3/u_3 = 1\}$.



Let us consider now a closed contour L, consisting of subsequently joined segment K_3K_2, segment K_2T_2, arc T_2T_3 of the ellipse and segment T_3K_3. This contour has the following property: any ray of the form$\{cv,\ c>0\}$, where $v \in \mathbf{R}^2$, $v \neq 0$, has exactly one common point with the contour L.

We denote by $n(y)$ the outer normal unitary vector of the contour L corresponding to the point $y \in L$, $n(y)$ is well defined at all points of L except the points K_2 and K_3 and, moreover, this function is continuous on L except the points K_2 and K_3. The behaviour of $n(y)$ on the arc T_2T_3 is of prime interest:

$$n(y) = \frac{\nabla e(y)}{\|\nabla e(y)\|}, \qquad \nabla e(y) = 2(\,ay_2 + b(y_2 - y_3), -b(y_2 - y_3)\,), \quad y \in T_2T_3 \subset L.$$

It is easy to see that $n(y) = n(T_2)$ for $y \in (K_2T_2]$, $n(y) = n(T_3)$ for $y \in [T_3K_3)$ and $n(y) = \left(u_2^{-1}, u_3^{-1}\right)$ for $y \in (K_3K_2)$. For the sequel it is important to point out the following points of the arc T_2T_3: $y^{(3)} = (-a^{-1/2}, -a^{-1/2})$ and $y^{(2)}$, $\left\{y^{(2)}\right\} = T_2T_3 \cap \left\{y_3^{(2)} = \frac{a+b}{b} y_2^{(2)}\right\}$. Obviously, both points belong to the domain $\{y_2 < 0,\ y_3 < 0\}$. It is easy to check that $n(y^{(2)}) \| Oy_3$, $n(y^{(3)}) \| Oy_2$.

7.3 Definition of Function φ and the Foster Conditions

For any point $(y_2, y_3) \in \mathbf{R}^2 \backslash \{0\}$ define $\varphi(y_2, y_3) > 0$ such that $\dfrac{(y_2, y_3)}{\varphi(y_2, y_3)} \in L$. For $(y_2, y_3) = 0$ we put $\varphi(0, 0) = 0$. The function $\varphi(y_2, y_3)$ is well-defined and has the following properties:

– $\varphi : \mathbf{R}^2 \to \mathbf{R}_+$ (positivity)
– $\varphi(ry_2, ry_3) = r\varphi(y_2, y_3)$, $r > 0$, (homogeneity)
– $L = \{y : \varphi(y) = 1\}$.

To finish the proof it is sufficient to check that the above defined function $\varphi(x)$ satisfies to the conditions of the Foster criterion (see Appendix). From the point of view of transition probabilities there are four different domains to consider:

$$\begin{aligned} E_- &:= \{y = (y_2, y_3) : \min(y_2, y_3) < 0\}, \quad E_{1,2} := \{y_2 > 0,\ y_3 = 0\}, \\ E_1 &:= \{y = (y_2, y_3) : y_2 > 0, y_3 > 0\}, \quad E_{1,3} := \{y_2 = 0,\ y_3 > 0\}\ . \end{aligned}$$

This is a technical job and we omit details.

8 Conclusions, Conjectures and Perspectives

We shall always assume that all $\lambda_1, \ldots, \lambda_N$ are different. Define a function $\ell(m) := \min\limits_{i \le m} \lambda_i$. Level sets of the function ℓ generate a partition of the set $\{1, \ldots, N\}$. Namely, there exists a sequence $j_1 = 1 < j_2 < \cdots < j_K < j_{K+1} = N + 1$ such that the set of all processors can be divided into several nonintersecting groups

$$\{1, \ldots, N\} = \bigcup_{k=\overline{1,K}} G_k\,, \tag{6}$$

$$G_k := \{j_k, j_k + 1, \ldots, j_{k+1} - 1\}\,, \quad \ell(j_k - 1) > \ell(j_k) = \cdots = \ell(j_{k+1} - 1) > \lambda_{j_{k+1}}.$$

Taking into account Theorems 3 and 4 and the above notion of groups of processors we put forward the following conjecture.

Conjecture 5. *Assume that all* $\lambda_1, \dots, \lambda_N$ *are different. Then for any* j *the following limit* $v_j^* = \lim\limits_{t\to+\infty} \dfrac{x_j(t)}{t}$ *exists and* $v_j^* = \ell(j)$.

Therefore this conjecture entails $v_j^* = \ell(j_k)$ for $j \in G_k$. If for some k the group G_k consists of *more than one* processor we may say that the processors of the group G_k are *synchronized.*

Remark 6 (On monotone cases). If $\lambda_1 < \cdots < \lambda_N$ then $v_j^* = \lambda_1$ for any j. If $\lambda_1 > \cdots > \lambda_N$ then for all j we have $v_j^* = \lambda_j$.

Let us discuss briefly perspectives of rigorous proof of the above conjecture for large values of N. In fact, we have already proved this conjectures for a wide class of cascade models.

Theorem 7. *Assume that all* $\lambda_1, \dots, \lambda_N$ *are different and a partition (6) of the set of processors* $\{1, \dots, N\}$ *is such that* $|G_k| \le 3$ *for all* k. *Then the limits* $v_j^* = \lim\limits_{t\to+\infty} \dfrac{x_j(t)}{t}$ *exist and* $v_j^* = \ell(j)$.

The proof of this statement is just a combination of the result of Theorem 4 (item 1) and arguments of the proof of items 2-4 of Theorem 4. We will not pursue further.

So the key to the proof of Conjecture 5 consists in generalization of item 1 of Theorem 4. As it was seen in Sect. 7, a possible way of such generalization is an explicit construction of Foster-Lyapunov function in high dimensions. This seems to be a difficult technical problem which is out of scope of this paper.

References

1. D. Jefferson, A. Witkowski, An Approach to Performance Analysis of Time stamp-driven Synchronization Mechanisms. 1984 ACM0-89791-143-1 84, 008/0243
2. D. Mitra, I. Mitrani, Analysis and Optimum performance of two message-passing parallel processors synchronized by rollback. Performance Evaluation 7 (1987), 111-124
3. V.K. Madisetti, J.C. Walrand and D.G. Messerschmitt, Asynchronous Algorithms for the ParaSimulation of Event-Driven Dynamical Systems, ACM Transactions on Modelling and Computer Simulation, Vol. 1, No 3, July 1991, Pages 244-274
4. A. Gupta, I.F. Akyildiz, Fujimoto, Performance Analysis of Time Warp With Multiple Homogeneous Processors. IEEE Transactions On Software Engineering, Vol. 17, No. 10, October 1991, 1013.
5. I.F. Akyildiz, L. Chen, S.R. Dast, R.M. Fujimoto, R.F. Serfozo, Performance Analysis of Time Warp with Limited Memory. Performance Evaluation Review, Vol. 20, No. 1, June 1992
6. A. Kumar and R. Shorey, Stability of Event Synchronisation in Distributed Discrete Event Simulation. Proc. of the eighth workshop on parallel and distributed simulation. Edinburgh, Scotland, United Kingdom. 65–72 (1994).

7. Fayolle G., Malyshev V., Menshikov M., Topics on constructive countable Markov chains. Cambridge University Press, 1995.
8. S.Yu. Popov, A.G. Greenberg, V.A. Malyshev, Stochastic models of massively parallel computation. Markov Processes and Related Fields, V.1, N4 (1995), 473-490.
9. A.G. Greenberg, S. Shenker, A.L. Stolyar, Asynchronous Updates in Large Parallel Systems. SIGMETRICS 96 5/96 PA, USA
10. M. Gupta, A. Kumar, R. Shorey, Queueing Models and Stability of Message Flows in Distributed Simulators of Open Queueing Networks. Proc. of the tenth workshop on parallel and distributed simulation. Philadelphia, Pennsylvania, United States. 162–169 (1996).
11. D.P. Bertsekas, J.N. Tsitsiklis, Parallel and Distributed Computation: Numerical Methods. Athena Scientific, Belmont, Mass. 1997.
12. R. Shorey, A. Kumar, and K.M. Rege, Instability and Performance Limits of Distributed Simulators of Feedforward Queueing Networks. ACM Transactions on Modeling and Computer Simulation, Vol. 7, No. 2, April 1997, Pages 210–238.
13. M. Gupta and A. Kumar, A Nonblocking Algorithm for the Distributed Simulation of FCFS Queueing Networks with Irreducible Markovian Routing. Proc. of the twelfth workshop on parallel and distributed simulation. Banff, Alberta, Canada. 20–27 (1998).
14. T.V. Voznesenskaya, Analysis of algorithms of time synchronisation for distributed simulation. Artificial intelligence (Donetsk), N2, 24-30 (2000) (in Russian).
15. T.V. Voznesenskaya, Mathematical model of algorithms of synchronization of time for the distributed simulation,in L.N. Korolev (Eds.), "Program systems and tools": the Thematic collection of faculty VMiK of the Moscow State University N1: MAX Press, 56-66 (2000).
16. Malyshev V., Manita A. Time synchronization problem. Rapport de recherche INRIA, N. 5204, 2004.
17. A. Manita, V. Shcherbakov. Asymptotic analysis of particle system with mean-field interaction, arXiv:math.PR/0408372 (http://arxiv.org), 2004.
18. Malyshev V.A., Manita A.D. Phase transitions in the time synchronization model. Probability Theory and Applications, Vol. 50, 150–158 (2005).

A Appendix

Let $(\xi_n,\, n = 0, 1, \ldots)$ be a countable irreducible aperiodic Markov chain with the state space $\mathcal{A}$. We use the following Foster criterion.

Theorem 8 ([7]). *The Markov chain ξ_n is ergodic if and only if there exists a positive function $f(\alpha)$, $\alpha \in \mathcal{A}$, a number $\varepsilon > 0$ and a finite set $A \in \mathcal{A}$ such that*

1) $\mathsf{E}\,(f(\xi_{n+1})\,|\,\xi_n = y) - f(y) < -\varepsilon$ *for all $y \notin A$,*

2) $\mathsf{E}\,(f(\xi_{n+1})\,|\,\xi_n = y) < +\infty$ *for all $y \in A$.*

The following theorem give a criterion of transience.

Theorem 9 ([7]). *The Markov chain is transient, if and only if there exists a positive function $f(\alpha)$ and a set A such that the following inequalities are fulfilled*

$$\mathsf{E}\,(f(\xi_{m+1})\,|\,\xi_m = \alpha_i) - f(\alpha_i) \le 0, \quad \forall \alpha_i \notin A,$$

$$f(\alpha_k) < \inf_{\alpha_j \in A} f(\alpha_j), \quad \textit{for at least one } \alpha_k \notin A.$$

On Construction of the Set of Irreducible Partial Covers

Mikhail Ju. Moshkov[1,2]

[1] Foundation for Support of Agrarian Reform and Rural Development, 21-1, B. Kharitonyevsky, Moscow 105064, Russia
[2] Institute of Computer Science, University of Silesia, 39, Będzińska St., Sosnowiec 41-200, Poland
moshkov@us.edu.pl

Abstract. In the paper a totally polynomial algorithm for construction of the set of irreducible partial covers for the major part of set cover problems is considered.

Keywords: Irreducible partial cover, totally polynomial algorithm.

1 Introduction

Set cover problem arises often in areas of computer science connected with analysis of data sets. In particular, problems of construction of minimal tests [2], decision rules and reducts [5] can be represented as set cover problems.

The paper is devoted to investigation of partial covers. If a data set contains noise then exact covers can be "over-learned" i.e. depend essentially on noise. If we see on constructed covers as on a way for knowledge representation [7] then instead of large exact covers it is more appropriate to work with small partial covers which cover the major part of elements.

Let A be a set with n elements and S be a family of m subsets of A. We consider so-called t-covers for the set cover problem (A, S). A t-cover is a subfamily of S, subsets from which cover at least $n - t$ elements from A. A t-cover is called irreducible if each proper subset of this cover is not a t-cover. We study the problem of construction of all irreducible t-covers for a given set cover problem. This problem is connected, in particular, with the evaluation of importance of attributes in decision tables.

We prove that if $t = 3\lceil \log_2 m \rceil$ and $n = \lfloor m^\alpha \rfloor$, where α is a positive real number, then there is no polynomial algorithm which for the major part of set cover problems constructs all irreducible t-covers, but there exists a totally polynomial algorithm which constructs all irreducible t-covers for the major part of set cover problems. Totally polynomial means that the algorithm has polynomial time complexity depending on total length of input and output. Note that this algorithm not only finds all irreducible t-covers but also identifies the characteristic function for the set of t-covers which is a monotone Boolean function.

The problem of identification of monotone Boolean function is the following: for a monotone Boolean function $f(x_1, \dots, x_m)$ (usually given by oracle) it is required to find the set $\min T(f)$ of minimal true vectors and the set $\max F(f)$ of maximal false vectors [1]. Totally polynomial algorithms for this problem solving (algorithms with polynomial time complexity depending on m, $|\min T(f)|$ and $|\max F(f)|$) are unknown. In [6] it was shown that there exists a totally polynomial algorithm which identifies almost all monotone Boolean functions. This algorithm is based on a totally polynomial algorithm for so-called k-tight monotone Boolean functions [4], and on the fact discovered in [3]: for almost all monotone Boolean functions all minimal true vectors and all maximal false vectors are located on only three layers of Boolean m-cube which are closed to central layers (with near to $\frac{m}{2}$ units in each vector).

It must be noted that for almost all of the considered set cover problems in characteristics functions for the set of t-covers all minimal true vectors and all maximal false vectors are located on lower layers of Boolean m-cube (with at most $2\lceil \alpha \log_2 m \rceil$ units in each vector).

2 Main Notions

Let $A = \{a_1, \dots, a_n\}$ be a nonempty finite set in which elements are enumerated by numbers $1, \dots, n$, and $S = \{B_1, \dots, B_m\}$ be a family of subsets of A in which subsets are enumerated by numbers $1, \dots, m$. It is possible that $B_1 \cup \ldots \cup B_m \neq A$, and subsets from S with different numbers are equal. The pair (A, S) will be called a *set cover problem.*

There is one-to-one correspondence between such set cover problems and tables with n rows and m columns filled by numbers from $\{0, 1\}$. The problem (A, S) corresponds to the table which for $i = 1, \dots, n$ and $j = 1, \dots, m$ has 1 at the intersection of i-th row and j-th column if and only if $a_i \in B_j$. So the number of different set cover problems (A, S) such that A contains n elements and S contains m subsets is equal to 2^{mn}.

Let t be a natural number. A subfamily $Q = \{B_{i_1}, \dots, B_{i_k}\}$ of the family S will be called a *t-cover for* (A, S) if

$$|B_{i_1} \cup \ldots \cup B_{i_k}| \geq |A| - t \ .$$

The number k will be called the *cardinality* of the considered t-cover. A t-cover will be called *irreducible* if each proper subset of this t-cover is not a t-cover.

3 On the Set of t-Covers

In this section we prove that if $t = 3\lceil \log_2 m \rceil$ then for the major part of set cover problems (A, S) any k subsets from S form a t-cover and any r subsets from S do not form a t-cover where k is near to $2\log_2 n$ and r is near to $\log_2 n$.

Theorem 1. *Consider set cover problems (A, S) such that $A = \{a_1, \ldots, a_n\}$ and $S = \{B_1, \ldots, B_m\}$. Let $t = 3\lceil \log_2 m \rceil$ and $k = 2\lceil \log_2 n \rceil$. Then the relative number of set cover problems for which any k subsets from S form a t-cover is at least*

$$1 - \frac{1}{2^{\lceil \log_2 m \rceil \lceil \log_2 n \rceil + \lceil \log_2 n \rceil}} \ .$$

Proof. Consider tables with n rows and m columns filled by numbers from $\{0, 1\}$. Fix k columns and $t + 1$ rows. The number of tables which have only 0 at the intersection of considered rows and columns is equal to $2^{mn-k(t+1)}$. There are at most m^k variants for the choice of k columns. There are at most n^{t+1} variants for the choice of $t + 1$ rows. Therefore the number of tables which have only 0 at the intersection of some k columns and some $t + 1$ rows is at most

$$2^{mn+k\log_2 m+(t+1)\log_2 n-k(t+1)} \leq 2^{mn-\lceil \log_2 m \rceil \lceil \log_2 n \rceil - \lceil \log_2 n \rceil} \ .$$

Last number is an upper bound on the number of set cover problems (A, S) for which there exist k subsets from S that do not form a t-cover. Therefore the relative number of set cover problems for which any k subsets from S form a t-cover is at least

$$\frac{2^{mn} - 2^{mn-\lceil \log_2 m \rceil \lceil \log_2 n \rceil - \lceil \log_2 n \rceil}}{2^{mn}} = 1 - \frac{1}{2^{\lceil \log_2 m \rceil \lceil \log_2 n \rceil + \lceil \log_2 n \rceil}} \ . \qquad \square$$

Theorem 2. *Consider set cover problems (A, S) such that $A = \{a_1, \ldots, a_n\}$ and $S = \{B_1, \ldots, B_m\}$. Let $t = 3\lceil \log_2 m \rceil$, $m \geq 2$, $n > t$,*

$$r = \lfloor \log_2(n - t) - \log_2 5 - \log_2 \lceil \log_2 m \rceil - \log_2 \lceil \log_2 n \rceil \rfloor$$

and $r > 0$. Then the relative number of set cover problems for which any r subsets from S do not form a t-cover is at least

$$1 - \frac{1}{2^{\lceil \log_2 m \rceil \lceil \log_2 n \rceil + \lceil \log_2 m \rceil}} \ .$$

Proof. Consider tables with n rows and m columns filled by numbers from $\{0, 1\}$. Fix r columns and t rows. Let us evaluate the number of tables in which there are no rows (with the exception, maybe, of the fixed t rows) that have only 0 at the intersection with the considered r columns. The number of such tables is equal to

$$2^{mn-r(n-t)}(2^r - 1)^{n-t} = 2^{mn}\left(\frac{2^r - 1}{2^r}\right)^{n-t} = 2^{mn}\left(\frac{2^r - 1}{2^r}\right)^{2^r \frac{n-t}{2^r}} \ .$$

Using well known inequality $\left(\frac{c-1}{c}\right)^c \leq \frac{1}{e}$, which holds for any natural c, obtain

$$2^{mn}\left(\frac{2^r - 1}{2^r}\right)^{2^r \frac{n-t}{2^r}} \leq 2^{mn-\frac{n-t}{2^r}} \ .$$

There are at most m^r variants for the choice of r columns. There are at most n^t variants for the choice of t rows. Therefore the number of tables which have r columns and $n - t$ rows such that among the considered rows there is no row with only 0 at the intersection with the considered r columns is at most $2^{mn+r\log_2 m+t\log_2 n-\frac{n-t}{2^r}}$. It is clear that

$$\frac{n-t}{2^r} \geq \frac{(n-t)5\lceil\log_2 m\rceil\lceil\log_2 n\rceil}{(n-t)} = 5\lceil\log_2 m\rceil\lceil\log_2 n\rceil \quad ,$$

$r \leq \lceil\log_2 n\rceil - 1$ and $r\log_2 m + t\log_2 n \leq 4\lceil\log_2 m\rceil\lceil\log_2 n\rceil - \lceil\log_2 m\rceil$. Hence

$$2^{mn+r\log_2 m+t\log_2 n-\frac{n-t}{2^r}} \leq 2^{mn-\lceil\log_2 m\rceil\lceil\log_2 n\rceil-\lceil\log_2 m\rceil} \quad .$$

Last number is an upper bound on the number of set cover problems (A, S) for each of which there exist r subsets from S that form a t-cover. Therefore the relative number of set cover problems for which any r subsets from S do not form a t-cover is at least

$$\frac{2^{mn} - 2^{mn-\lceil\log_2 m\rceil\lceil\log_2 n\rceil-\lceil\log_2 m\rceil}}{2^{mn}} = 1 - \frac{1}{2^{\lceil\log_2 m\rceil\lceil\log_2 n\rceil+\lceil\log_2 m\rceil}} \quad . \qquad \square$$

Corollary 1. *Consider set cover problems (A, S) such that $A = \{a_1, \ldots, a_n\}$ and $S = \{B_1, \ldots, B_m\}$. Let $t = 3\lceil\log_2 m\rceil$, $k = 2\lceil\log_2 n\rceil$,*

$$r = \lfloor\log_2(n-t) - \log_2 5 - \log_2\lceil\log_2 m\rceil - \log_2\lceil\log_2 n\rceil\rfloor \quad ,$$

$m \geq 2$, $n > t$ and $r > 0$. Then the relative number of set cover problems for which any k subsets from S form a t-cover, and any r subsets from S do not form a t-cover is at least

$$1 - \frac{1}{2^{\lceil\log_2 m\rceil\lceil\log_2 n\rceil}} \quad .$$

4 On the Set of Irreducible *t*-Covers

Let (A, S) be a set cover problem. Denote by $I(A, S, t)$ the number of irreducible t-covers for (A, S). In this section we obtain lower and upper bounds on the value $I(A, S, t)$ for $t = 3\lceil\log_2 m\rceil$ and for the major part of set cover problems such that $n = \lfloor m^\alpha\rfloor$, where α is a positive real number, and m is large enough.

Theorem 3. *Consider set cover problems (A, S) such that $A = \{a_1, \ldots, a_n\}$, $S = \{B_1, \ldots, B_m\}$ and $n = \lfloor m^\alpha\rfloor$ where α is a real number and $\alpha > 0$. Let $t = 3\lceil\log_2 m\rceil$ and $k = 2\lceil\log_2 n\rceil$. Then for large enough m the relative number of set cover problems (A, S) for which any k subsets from S form a t-cover, and*

$$m^{\frac{\alpha}{4}\log_2 m} \leq I(A, S, t) \leq m^{5\alpha\log_2 m}$$

is at least

$$1 - \frac{1}{2^{\lceil\log_2 m\rceil\lceil\alpha\log_2 m\rceil}} \quad .$$

Proof. Denote $k = 2\lceil \log_2 \lfloor m^\alpha \rfloor \rceil$ and

$$r = \lfloor \log_2(\lfloor m^\alpha \rfloor - t) - \log_2 5 - \log_2 \lceil \log_2 m \rceil - \log_2 \lceil \log_2 \lfloor m^\alpha \rfloor \rceil \rfloor \ .$$

From Corollary 1 it follows that for large enough m the relative number of set cover problems for which any k subsets from S form a t-cover, and any r subsets from S do not form a t-cover is at least

$$1 - \frac{1}{2^{\lceil \log_2 m \rceil \lceil \log_2 \lfloor m^\alpha \rfloor \rceil}} \geq 1 - \frac{1}{2^{\lceil \log_2 m \rceil \lceil \alpha \log_2 m \rceil}} \ .$$

Consider an arbitrary set cover problem (A, S) for which any k subsets from S form a t-cover, and any r subsets from S do not form a t-cover. Let us show that if m is large enough then

$$m^{\frac{\alpha}{4} \log_2 m} \leq I(A, S, t) \leq m^{5\alpha \log_2 m} \ .$$

It is clear that each t-cover has an irreducible t-cover as a subset. Let Q be an irreducible t-cover. Let us evaluate the number of t-covers of cardinality k which have Q as a subset. Let $|Q| = p$. One can show that $r + 1 \leq p \leq k$. There are C_{m-p}^{k-p} ways to obtain a t-cover of cardinality k from Q by adding subsets from S. It it clear that $C_{m-p}^{k-p} \leq C_m^{k-p}$. If $k < \frac{m}{2}$ then $C_m^{k-p} \leq C_m^{k-r}$. Thus, for large enough m the number of t-covers of cardinality k which have Q as a subset is at most C_m^{k-r}.

The number of t-covers of cardinality k is equal to C_m^k. Hence

$$I(A, S, t) \geq \frac{C_m^k}{C_m^{k-r}} = \frac{(m-k+1)\dots(m-k+r)}{(k-r+1)\dots k} \geq \left(\frac{m-k}{k}\right)^r \ .$$

For large enough m

$$\frac{m-k}{k} = \frac{m - 2\lceil \log_2 \lfloor m^\alpha \rfloor \rceil}{2\lceil \log_2 \lfloor m^\alpha \rfloor \rceil} \geq m^{\frac{1}{2}} \ .$$

Therefore $I(A, S, t) \geq m^{\frac{r}{2}}$. It is clear that for large enough m the inequality $r \geq \frac{1}{2}\alpha \log_2 m$ holds. Thus, for large enough m

$$I(A, S, t) \geq m^{\frac{\alpha}{4} \log_2 m} \ .$$

It is clear that the cardinality of each irreducible t-cover is at most k. Therefore $I(A, S, t) \leq km^k$ if $k < \frac{m}{2}$. One can show that for large enough m the inequality $km^k \leq m^{5\alpha \log_2 m}$ holds. Thus, for large enough m

$$I(A, S, t) \leq m^{5\alpha \log_2 m} \ .$$ □

5 On Algorithms for Construction of All Irreducible *t*-Covers

Let us consider set cover problems (A, S) such that $A = \{a_1, \dots, a_n\}$, $S = \{B_1, \dots, B_m\}$ and $n = \lfloor m^\alpha \rfloor$ where α is a real number and $\alpha > 0$. Let

$t = 3\lceil \log_2 m \rceil$ and $k = 2\lceil \log_2 n \rceil$. For a given set cover problem (A, S) it is required to find all irreducible t-covers for (A, S). The length of input for this problem is equal to $mn \le m^{1+\alpha}$.

From Theorem 3 it follows that for large enough m the relative number of set cover problems (A, S) for which any k subsets from S form a t-cover, and

$$m^{\frac{\alpha}{4}\log_2 m} \le I(A, S, t) \le m^{5\alpha \log_2 m}$$

is at least

$$1 - \frac{1}{2^{\lceil \log_2 m \rceil \lceil \alpha \log_2 m \rceil}} \ .$$

Thus, there is no polynomial algorithm which for large enough m for the major part of set cover problems constructs the set of irreducible t-covers.

Let us consider an algorithm which finds all nonempty subfamilies of the family S with at most $k = 2\lceil \log_2 n \rceil$ subsets, and for each such subfamily recognizes is this subfamily a t-cover or not. It is clear that this recognition problem can be solved (for one subfamily) in polynomial time depending on mn. After that among the considered subfamilies the algorithm chooses all minimal subfamilies which are t-covers (such subfamilies are irreducible t-covers and correspond to minimal true vectors of the characteristic function for the set of t-covers) and all maximal subfamilies which are not t-covers (such subfamilies correspond to maximal false vectors of the characteristic function for the set of t-covers).

From Theorem 3 it follows that for large enough m for the major part of set cover problems the algorithm finds all minimal true vectors of the characteristic function for the set of t-covers (all irreducible t-covers) and all maximal false vectors of the characteristic function for the set of t-covers.

The considered algorithm for large enough m works with at most km^k subfamilies of S. One can show that $km^k \le m^{5\alpha \log_2 m}$ for large enough m. Using Theorem 3 we conclude that for large enough m

$$km^k \le (I(A, S, t))^{20} \ .$$

Thus, there exists a totally polynomial algorithm which for large enough m for the major part of set cover problems constructs the set of irreducible t-covers and finds all minimal true and all maximal false vectors of the characteristic function for the set of t-covers.

6 Conclusions

In the paper it is shown that under some assumptions on m, n and t there is no polynomial algorithm which for the major part of set cover problems (A, S) with A containing n elements and S containing m subsets constructs all irreducible t-covers, but there exists a totally polynomial algorithm which for the major part of considered set cover problems constructs all irreducible t-covers.

Similar results can be obtained for more wide classes of set cover problems and thresholds t.

Acknowledgments

The author is greatly indebted to Andrzej Skowron for stimulating discussions, and to anonymous reviewer for helpful remarks.

References

1. Boros, E., Hammer, P.L., Ibaraki, T., Kawakami, K.: Polynomial time recognition of 2-monotonic positive Boolean functions given by an oracle. SIAM J. Computing **26**(1) (1997) 93–109
2. Chegis, I.A., Yablonskii, S.V.: Logical methods of electric circuit control. Trudy MIAN SSSR **51** (1958) 270–360 (in Russian)
3. Korshunov, A.D.: On the number of monotone Boolean functions. Problemy Kibernetiki **38** (1981) 5–108 (in Russian)
4. Makino, K., Ibaraki, T.: The maximum latency and identification of positive Boolean functions. SIAM J. Computing **26**(5) (1997) 1363–1383
5. Pawlak, Z.: Rough Sets – Theoretical Aspects of Reasoning about Data. Kluwer Academic Publishers, Dordrecht, Boston, London, 1991
6. Shmulevich, I., Korshunov, A.D., Astola, J.: Almost all monotone Boolean functions are polynomially lernable using membership queries. Information Processing Letters **79** (2001) 211–213
7. Skowron, A.: Rough sets in KDD. Proceedings of the 16-th World Computer Congress (IFIP'2000). Beijing, China (2000) 1–14

Recent Advances in Multiobjective Optimization*

Christos Zaroliagis[1,2]

[1] Computer Technology Institute, P.O. Box 1122, 26110 Patras, Greece
[2] Department of Computer Engineering and Informatics,
University of Patras, 26500 Patras, Greece
zaro@ceid.upatras.gr

Multiobjective (or multicriteria) optimization is a research area with rich history and under heavy investigation within Operations Research and Economics in the last 60 years [1,2]. Its object of study is to investigate solutions to combinatorial optimization problems that are evaluated under several objective functions – typically defined on multidimensional attribute (cost) vectors. In multiobjective optimization, we are interested not in finding a single optimal solution, but in computing the *trade-off* among the different objective functions, called the *Pareto set (or curve)* $\mathcal{P}$, which is the set of all feasible solutions whose vector of the various objectives is *not* dominated by any other solution.

Multiobjective optimization problems are usually NP-hard due to the fact that the Pareto set is typically exponential in size (even in the case of two objectives). On the other hand, even if a decision maker is armed with the entire Pareto set, s/he is still left with the problem of which is the "best" solution for the application at hand. Consequently, three natural approaches to deal with multiobjective optimization problems are to:

(i) Study approximate versions of the Pareto curve that result in (guaranteed) near optimal but smaller Pareto sets.
(ii) Optimize one objective while bounding the rest (*constrained approach*).
(iii) Proceed in a normative way and choose the "best" solution by introducing a utility (often non-linear) function on the objectives (*normalization approach*).

Until quite recently, the vast majority of research in multiobjective optimization [1,2] had focussed either on exact methods (i.e., to compute the entire Pareto set), or approximation methods through heuristic and metaheuristic approaches (that do not provide guarantees on the quality of the returned solution). An important outcome of the existing literature is that the two objectives case has been extensively studied, while there is a certain lack of efficient (generic) methods for the case of more than two objectives. Most importantly, there has been

* This work was partially supported by the IST Programme (6th FP) of EC under contract No. IST-2002-001907 (integrated project DELIS), and the Action PYTHAGORAS of the Operational Programme for Educational & Vocational Training II, with matching funds from the European Social Fund and the Greek Ministry of Education.

a lack of a systematic study of the complexity issues regarding approximate versions of the Pareto set (in a way analogous to the well-established approximation theory for single objective optimization problems).

The normalization approach has been quite investigated, especially within Operations Research. In this approach, a utility function is introduced that translates (in a linear or non-linear way) the different criteria into a common cost measure. For instance, when travelling in a traffic network one typically wishes to minimize travel distance and time; both criteria can be translated into a common cost measure (e.g., money), where the former is linearly translated, while the latter non-linearly (small amounts of time have relatively low value, while large amounts of time are very valuable). Under the normalization approach, we seek for a single optimum in the Pareto set (a feasible solution that optimizes the utility function). Due to the exponential size of the Pareto set, a fair portion of research had focussed on solving relaxed versions of the optimization problem, which corresponds to finding the best solution in the convex hull of the Pareto set. This turns out to be a good starting point to locate the exact solution by applying heuristic methods. However, the approaches used so far employ exhaustive algorithms for solving the relaxations of the problem at hand with complexities bounded by some polynomial in the size of the convex hull (which can be subexponentially large). In a very recent study [9], the first efficient (polynomial time) algorithm for solving the relaxation of the normalized version of the bicriteria shortest path is given, when the utility function is non-linear and convex.

The constrained approach had been (almost exclusively) the method adopted within Computer Science to deal with multiobjective optimization problems; see e.g., [3,4,6,8]. Classical examples concern the restricted (or constrained) shortest path and the restricted (or constrained) spanning tree problems.

Very recently, a systematic study (within Computer Science) has been initiated regarding the complexity issues of approximate Pareto curves [7]. Informally, an $(1+\varepsilon)$-*Pareto curve* $\mathcal{P}_\varepsilon$ is a subset of feasible solutions such that for any Pareto optimal solution and any $\varepsilon > 0$, there exists a solution in $\mathcal{P}_\varepsilon$ that is no more than $(1+\varepsilon)$ away in all objectives. Although this concept is not new (it has been previously used in the context of bicriteria and multiobjective shortest paths [5,12]), Papadimitriou and Yannakakis in a seminal work [7] show that for *any* multiobjective optimization problem there exists a $(1+\varepsilon)$-Pareto curve $\mathcal{P}_\varepsilon$ of (polynomial) size $|\mathcal{P}_\varepsilon| = O((4B/\varepsilon)^{d-1})$, where B is the number of bits required to represent the values in the objective functions (bounded by some polynomial in the size of the input). They also provide a necessary and sufficient condition for its efficient (polynomial in the size of the input and $1/\varepsilon$) construction. In particular, $\mathcal{P}_\varepsilon$ can be constructed by $O((4B/\varepsilon)^d)$ calls to a GAP routine that solves (in time polynomial in the size of the input and $1/\varepsilon$) the following problem: given a vector of values $\mathbf{a}$, either compute a solution that dominates $\mathbf{a}$, or report that there is no solution better than $\mathbf{a}$ by at least a factor of $1+\varepsilon$ in all objectives. Extensions to that method to produce a constant approximation to the smallest possible $(1+\varepsilon)$-Pareto curve for the cases of 2 and 3 objectives are presented in [11], while for $d > 3$ objectives inapproximability results are shown for such a constant approximation.

Apart from the above general results, there has been very recent work on improved approximation algorithms (FPTAS) for multiobjective shortest paths [10]. In that paper, a new and remarkably simple algorithm is given that constructs $(1 + \varepsilon)$-Pareto sets for the single-source multiobjective shortest path problem, which improves considerably upon previous approaches. In the same paper, it is also shown how this algorithm can provide better approximation schemes for both the constrained and the normalized versions of the problem for any number of objectives. An additional byproduct is a generic method for constructing FPTAS for any multiobjective optimization problem with non-linear objectives of a rather general form (that includes any polynomial of bounded degree with non-negative coefficients). This method does not require the existence of a GAP routine for such non-linear objectives.

All these very recent algorithmic and complexity issues will be discussed and elaborated in the talk.

References

1. M. Ehrgott, *Multicriteria Optimization*, Springer, 2000.
2. M. Ehrgott and X. Gandibleux (Eds), *Multiple Criteria Optimization – state of the art annotated bibliographic surveys*, Kluwer Academic Publishers, Boston, 2002.
3. O. Etzioni, S. Hanks, T. Jiang, R. Karp, O. Madari, and O. Waarts, "Efficient Information Gathering on the Internet", in *Proc. 37th IEEE Symp. on Foundations of Computer Science* – FOCS 1996, pp. 234-243.
4. G. Handler and I. Zang, "A Dual Algorithm for the Constrained Shortest Path Problem", *Networks* 10(1980), pp. 293-310.
5. P. Hansen, "Bicriterion Path Problems", *Proc. 3rd Conf. Multiple Criteria Decision Making – Theory and Applications*, LNEMS Vol. 117 (Springer, 1979), pp. 109-127.
6. M. Marathe, R. Ravi, R. Sundaram, S.S. Ravi, D. Rosenkrantz, and H.B. Hunt, "Bicriteria Network Design Problems", *Journal of Algorithms* 28 (1998), pp. 142-171.
7. C. Papadimitriou and M. Yannakakis, "On the Approximability of Trade-offs and Optimal Access of Web Sources", in *Proc. 41st IEEE Symp. on Foundations of Computer Science* – FOCS 2000, pp. 86-92.
8. R. Ravi, M. Marathe, S.S. Ravi, D. Rosenkrantz, and H.B. Hunt, "Many Birds with One Stone: Multi-objective Approximation Algorithms", in *Proc. 25th ACM Symp. on Theory of Computing* – STOC 1993, pp. 438-447.
9. G. Tsaggouris and C. Zaroliagis, "Non-Additive Shortest Paths", in *Algorithms* – ESA 2004, LNCS Vol. 3221 (Springer-Verlag, 2004), pp. 822-834.
10. G. Tsaggouris and C. Zaroliagis, "Improved FPTAS for Multiobjective Shortest Paths with Applications", CTI Technical Report TR 2005/07/03, July 2005.
11. S. Vassilvitskii and M. Yannakakis, "Efficiently Computing Succinct Trade-off Curves", in *Automata, Languages, and Programming* – ICALP 2004, LNCS Vol. 3142 (Springer, 2004), pp. 1201-1213.
12. A. Warburton, "Approximation of Pareto Optima in Multiple-Objective Shortest Path Problems", *Operations Research* 35(1987), pp. 70-79.

Polynomial Time Checking for Generation of Finite Distributions of Rational Probabilities[⋆]

Roman Kolpakov

Moscow State University, Moscow 119992, Russia
foroman@mail.ru

Abstract. We study the generation of finite probabilistic distributions by discrete transformations. By discrete transformation of distributions we mean the distribution of a random variable which is a function of the values of independent random variables with initial distributions. We propose an algorithm which allows to determine in polynomial time whether a given distribution is generated by a given set of finite distributions of rational probabilities. Moreover, we describe all sets of finite distributions of rational probabilities which are closed under the considered generation. Among these sets we find all finitely generated sets. We also determine the structure of the lattice formed of these sets.

Keywords: Stochastic automata, probabilistic transformations, generation of randomness.

1 Introduction

We consider discrete transformations of independent finite distributions of rational probabilities. By transformation of distributions we mean the distribution of a random variable which is a function of random variables with initial distributions. Such transformations are of great importance for the generation of randomness which plays a vital role in many areas of computer science: structural theory of stochastic automata, simulations, network constructions, cryptography, and so on (see [1, 12]). More precisely, the generation of randomness is based actually on construction of some random variable ζ_0 with a wanted distribution proceeding from initial disposable random variables $\zeta_1, \ldots, \zeta_k$ with distributions different from the wanted distribution. The variable ζ_0 is defined by some function $f : \Omega_1 \times \ldots \times \Omega_k \longrightarrow \Omega_0$ where Ω_i is the set of values of the random variable ζ_i, $i = 0, 1, \ldots, k$. In the paper we consider random variables which have a finite number of values. In this case without loss of generality we can assume that for any $i = 0, 1, \ldots, k$ the set Ω_i is $\{0, 1, \ldots, h_i - 1\}$, and the probabilistic distribution of ζ_i is defined by a stochastic vector[1] $\mathcal{D}_i$ of dimension

⋆ This work is supported by the Russian Foundation for Fundamental Research (Grant 05-01-00994), the program for supporting of leading scientific schools (Grant NSh-1807.2003.1), and the program "Universities of Russia" (Grant UR.04.02.528).

[1] A *stochastic* vector is a vector with nonnegative components such that the sum of all components of the vector is equal to 1.

h_i such that jth component of $\mathcal{D}_i$ is the probability of ζ_i to be equal to $j-1$. We will denote the set $\Omega_1 \times \ldots \times \Omega_k = \{0,1,\ldots,h_1-1\} \times \ldots \times \{0,1,\ldots,h_k-1\}$ by $\Omega(\mathcal{D}_1,\ldots,\mathcal{D}_k)$. For $i = 0,1,\ldots,h_0-1$ denote by $\mathcal{N}_i(f)$ the set of all tuples from $\Omega(\mathcal{D}_1,\ldots,\mathcal{D}_k)$ on which the function f is equal to i. By $\mathcal{D}[j]$ we denote jth component of a stochastic vector $\mathcal{D}$. Note that if all variables $\zeta_1,\ldots,\zeta_k$ are independent, the vector $\mathcal{D}_0$ is determined uniquely by the vectors $\mathcal{D}_1,\ldots,\mathcal{D}_k$ and the function f. In particular, we can define components of the vector $\mathcal{D}_0$ in the following way:

$$\mathcal{D}_0[j] = \sum_{(\sigma_1;\ldots;\sigma_k)\in\mathcal{N}_{j-1}(f)} \mathcal{D}_1[\sigma_1+1]\cdot\ldots\cdot\mathcal{D}_k[\sigma_k+1]. \tag{1}$$

Further we denote the vector $\mathcal{D}_0$ by $\mathbf{P}\{f(\mathcal{D}_1,\ldots,\mathcal{D}_k)\}$.

Let H be a set of distinct stochastic vectors. We say that a stochastic vector $\mathcal{D}$ is *generated* by the set H if there exists a function $f(x_1,\ldots,x_k)$ such that $\mathcal{D} = \mathbf{P}\{f(\mathcal{D}_1,\ldots,\mathcal{D}_k)\}$ for some $\mathcal{D}_1,\ldots,\mathcal{D}_k \in H$. We denote by $\langle H\rangle$ the set of all stochastic vectors generated by H and call this set the *closure* of the set H. The set H is called *closed* if $H = \langle H\rangle$. Note that $H \subseteq \langle H\rangle$ and $\langle H' \cap H''\rangle \subseteq \langle H'\rangle \cap \langle H''\rangle$ for any sets H' and H'' of stochastic vectors. Below (Corollary 1) we show that $\langle\langle H\rangle\rangle = \langle H\rangle$, i.e. the considered operation of closure of sets of stochastic vectors satisfies also the third standard property of a operation of closure. We say also that a set A of stochastic vectors is *generated* by the set H if $A \subseteq \langle H\rangle$. For a set T of natural numbers and a natural k we denote by $T^{>k}$ the set of all numbers of T which are greater than k. By $\mathcal{I}(n)$ we denote the set of all prime divisors for a natural number n, and by $(x_1,\ldots,x_k)$ we denote the greatest common divisor of numbers $x_1,\ldots,x_k$.

Investigating the introduced generation of stochastic vectors, first of all we face the problem of determining whether a given stochastic vector is generated by a given set of stochastic vectors. The basic difficulty of this problem consists, obviously, in impossibility of the direct description of arbitrary sets of stochastic vectors, since the set of all stochastic vectors has the continuum cardinality. So one of the natural approaches for solving this problem is to consider it on closed subsets of stochastic vectors everywhere dense in the set of all stochastic vectors. The set of all stochastic vectors with rational components is the most appropriate example of such subsets. We denote this set by **SQ**. Then for the set **SQ** we have the following problem.

Problem 1 (Generation Problem on **SQ***).* For any stochastic vector $\mathcal{D}$ and any finite set M of stochastic vectors from **SQ** determine whether $\mathcal{D}$ is generated by M.

Another important problem which is closely related to the Generation Problem on **SQ** is the problem of description of all closed subsets of the set **SQ**. For any set Π of different prime numbers we consider in **SQ** the subset $SG[\Pi]$ of all stochastic vectors such that any component of these vectors can be represented by a fraction of which the denominator is a production of powers of numbers from Π:

$$SG[\Pi] = \left\{ (d_1; \ldots ; d_h) \;\middle|\; \begin{array}{l} \sum_{i=1}^{h} d_i = 1,\ d_i = \frac{m_i}{n},\ m_i \in \mathbb{Z}^+,\ i = 1, \ldots , h \\ n \in \mathbb{N},\ \mathcal{I}[n] \subseteq \Pi \end{array} \right\}.$$

It is easy to see from (1) that subsets of a set $SG[\Pi]$ can generate only stochastic vectors from the same set. Thus sets $SG[\Pi]$ present an example of closed subsets of **SQ**. Further we show that in **SQ** there exist closed subsets different from sets $SG[\Pi]$.

Most of investigations in this field concerned the case of generation of two-dimensional stochastic vectors from **SQ** (since a two-dimensional stochastic vector is determined uniquely by any one of its components, instead of two-dimensional stochastic vectors the numbers of the segment $[0, 1]$ which are the second components of these vectors are usually considered in the literature). In [10, 11] it is shown that the sets of all two-dimensional stochastic vectors from $SG[\{2\}]$ and $SG[\{3\}]$ are generated by the vectors $\left(\frac{1}{2}; \frac{1}{2}\right)$ and $\left(\frac{1}{3}, \frac{2}{3}\right)$ respectively. Thus these sets are *finitely generated*, i.e., generated by some their own finite subsets. In [6, 9] these results are generalized to the case of the set of all two-dimensional stochastic vectors from $G[\Pi]$ for an arbitrary Π. Moreover, the lattice formed of these sets is described. Analogous results for the case of stochastic vectors of arbitrary dimensions are obtained in [7, 8]. A number of questions on the approximate generation of two-dimensional stochastic vectors are considered in [5, 11]. In [2] an explicit description for the closures of arbitrary sets in the class of all two-dimensional stochastic vectors from $G[\Pi]$ is obtained, and, basing on this description, all closed subsets of this class are found. In [3] the closures of all finite sets of arbitrary vectors from **SQ** are explicitly described. Using this result, we propose a polynomial time algorithm for solving the Generation Problem on **SQ**. Moreover, we present all closed and all finitely generated closed subsets of the set **SQ**. We also determine the structure of the lattice formed of these subsets.

2 Auxiliary Definitions and Results

First of all we note the following property of stochastic vectors which can be verified directly.

Proposition 1. *Let $\mathcal{D}_1^{(1)}, \ldots , \mathcal{D}_{k(1)}^{(1)}, \ldots , \mathcal{D}_1^{(n)}, \ldots , \mathcal{D}_{k(n)}^{(n)}$ be stochastic vectors, $f_1(x_1, \ldots , x_{k(1)})$,..., $f_n(x_1, \ldots , x_{k(n)})$ be discrete functions defined respectively on the sets $\Omega(\mathcal{D}_1^{(1)}, \ldots , \mathcal{D}_{k(1)}^{(1)})$,..., $\Omega(\mathcal{D}_1^{(n)}, \ldots , \mathcal{D}_{k(n)}^{(n)})$, and $f(x_1, \ldots , x_n)$ be a discrete functions defined on the set $\Omega(\mathcal{D}_1, \ldots , \mathcal{D}_n)$ where $\mathcal{D}_i = \mathbf{P}\{f_i(\mathcal{D}_1^{(i)}, \ldots , \mathcal{D}_{k(i)}^{(i)})\}$, $i = 1, \ldots , n$. Then*

$$\mathbf{P}\{f(\mathcal{D}_1, \ldots , \mathcal{D}_n)\} = \mathbf{P}\{\hat{f}(\mathcal{D}_1^{(1)}, \ldots , \mathcal{D}_{k(1)}^{(1)}, \ldots , \mathcal{D}_1^{(n)}, \ldots , \mathcal{D}_{k(n)}^{(n)})\}$$

where $\hat{f}(x_1^{(1)}, \ldots , x_{k(1)}^{(1)}, \ldots , x_1^{(n)}, \ldots , x_{k(n)}^{(n)}) = f\big(f_1(x_1^{(1)}, \ldots , x_{k(1)}^{(1)}), \ldots , f_n(x_1^{(n)}, \ldots , x_{k(n)}^{(n)})\big)$.

Corollary 1. *The closure of any set of stochastic vectors is a closed set.*

Thus, the considered operation of closure of sets of stochastic vectors satisfies all standard properties of a operation of closure.

Stochastic vector is called *degenerated* if it has a component which is equal to 1. By natural way we assume that all degenerated stochastic vectors are generated by the empty set and, therefore, are contained in any closed set of stochastic vectors. For any nondegenerated stochastic vector $\mathcal{D}$ we denote by $\mathcal{D}^+$ the vector which is obtained from $\mathcal{D}$ by removing of all zero components. For any set M of stochastic vectors we denote by M^+ the set of all vectors $\mathcal{D}^+$ such that $\mathcal{D} \in M$. The following obvious fact takes place.

Proposition 2. *For any set M of stochastic vectors the relation $\langle M \rangle = \langle M^+ \rangle$ is valid.*

Nondegenerated stochastic vectors with nonzero components are called *positive*. A set M of stochastic vectors is called *closed positively* if for any nondegenerated vector $\mathcal{D}$ of H the vector $\mathcal{D}^+$ is also contained in H.

Natural numbers $a_1, a_2, \ldots, a_k$ are called *pairwise relatively prime* if each of these numbers is relatively prime to any other of them. We call a set of numbers of $\mathbf{N}^{>1}$ *divisible* if it contains less than two numbers or all its numbers are pairwise relatively prime. We call also a set of natural numbers *relatively prime* to a natural number n if each of the numbers of this set is relatively prime to n. An empty set is supposed to be relatively prime to any natural number. If a set A of natural numbers is finite we denote by $|A|$ the number of elements of A and by $\|A\|$ the product of all numbers of A. For an empty set we assume $\|\emptyset\| = 1$.

Let A, B be non-empty divisible sets of natural numbers. We call the set B a *divisor* of the set A if for any number b of B the set A has a number divisible by b. An empty set is supposed to be a divisor of any divisible set. Let $A_1, \ldots, A_s$ be finite divisible sets. The *greatest common divisor* $(A_1, \ldots, A_s)$ of these sets is the set

$$\{\, a \mid a = (a_1, a_2, \ldots, a_s) > 1,\ a_i \in A_i,\ i = 1, 2, \ldots, s \,\},$$

which consists of the numbers of $\mathbf{N}^{>1}$ that are greatest common divisors of all possible s-tuples formed by picking out one number of each of the sets $A_1, \ldots, A_s$. If at least one of the sets $A_1, \ldots, A_s$ is empty we assume $(A_1, \ldots, A_s) = \emptyset$. Note the following obvious properties of the set $(A_1, \ldots, A_s)$.

Proposition 3. *The greatest common divisor of divisible sets is a divisible set which is relatively prime to any number relatively prime to at least one of these sets.*

Proposition 4. *For any finite divisible sets $A_1, \ldots, A_s$ the equality $\|(A_1, \ldots, A_s)\| = (\|A_1\|, \ldots, \|A_s\|)$ is valid.*

We can also introduce the notion of greatest common divisor for an infinite number of finite divisible sets. Let $A_1, A_2, \ldots$ be finite divisible sets. We define the *greatest common divisor* $(A_1, A_2, \ldots)$ of these sets as the set

$$\{\, a \mid a > 1,\ a = (a_1, a_2, \ldots),\ a_i \in A_i,\ i = 1, 2, \ldots \,\},$$

which consists of all the numbers of $\mathbf{N}^{>1}$ that are greatest common divisors of infinite samples of numbers of $A_1, A_2, \ldots$ formed by picking out one number of each of the sets. If at least one of the sets $A_1, A_2, \ldots$ is empty we assume $(A_1, A_2, \ldots) = \emptyset$. Analogously to Proposition 3, we have

Proposition 5. *The greatest common divisor of an infinite number of divisible sets is a divisible set which is relatively prime to any number relatively prime to at least one of these sets.*

3 Closed Subsets of SQ

Let Π be a non-empty set of different prime numbers, and T be a finite divisible set of numbers relatively prime to the set Π. Denote by $SG[\Pi; T]$ the following subset of the set $SG[\Pi]$:

$$\left\{ (d_1; \ldots; d_h) \;\middle|\; \begin{array}{l} d_i = \frac{m_i}{n},\ m_i \in \mathbb{Z}^+,\ i = 1, \ldots, h,\ n \in \mathbb{N},\ \mathcal{I}(n) \subseteq \Pi, \\ \sum_{i=1}^{h} d_i = 1,\ \exists T_1, \ldots, T_{h-1},\ T \supseteq T_1 \supseteq T_2 \supseteq \ldots \supseteq T_{h-1}, \\ \sum_{j=1}^{i} m_j \equiv 0 \pmod{\|T_i\|},\ i = 1, \ldots, h-1, \\ \sum_{j=1}^{i} m_j \equiv n \pmod{\|T \setminus T_i\|},\ i = 1, \ldots, h-1 \end{array} \right\}$$

(the sets $T_1 \ldots, T_{h-1}$ can coincide T or $\emptyset$). In case of $T = \emptyset$ we assume that $SG[\Pi; \emptyset] = SG[\Pi]$. One can note that the validity of the relation $(d_1; \ldots; d_h) \in SG[\Pi; T]$ depends formally on choosing the common denominator n of components $d_1, \ldots, d_h$. However, it is easy to verify that this dependence is actually fictitious, i.e. our definition of $SG[\Pi; T]$ is invariant with respect to multiplication or division of nominators and denominators of fractions by the same number. Note that for any vector $(d_1; \ldots; d_h)$ of $SG[\Pi; T]$ the corresponding sets $T_1 \ldots, T_{h-1}$ are determined uniquely by this vector. Note also that any set $SG[\Pi; T]$ is infinite and closed positively and contains all degenerated vectors. In [4] the following fact is proved.

Lemma 1. *For any sets Π and T the set $SG[\Pi; T]$ is closed.*

According to Lemma 1, all sets $SG[\Pi; T]$ form a collection of closed subsets of the set **SQ**. We denote this collection by **SG**. The following proposition gives for any two sets $SG[\Pi'; T']$, $SG[\Pi''; T'']$ of **SG** the relationship between the inclusion $SG[\Pi'; T'] \subseteq SG[\Pi''; T'']$ and the parameters Π', Π'', T', T''.

Proposition 6. *Let $G[\Pi'; T']$, $G[\Pi''; T'']$ be two sets of* **SG**. *Then*

1. *$SG[\Pi'; T'] \subseteq SG[\Pi''; T'']$ if and only if $\Pi' \subseteq \Pi''$ and T'' is a divisor of T';*
2. *$SG[\Pi'; T'] = SG[\Pi''; T'']$ if and only if $\Pi' = \Pi''$ and $T' = T''$.*

Proposition 6 allows to establish the inclusion relation between any sets of **SG** and to describe in this way the structure of the lattice formed of all sets from **SG**.

4 Closures of Finite Subsets of SQ

Our solution of the Generation Problem on **SQ** is based on an explicit description of closures of all finite subsets of the set **SQ**. In order to give this description, by Proposition 2 it is enough to consider only subsets consisting of positive vectors. First we consider the case of a subset consisting of only one positive vector $\mathcal{D} = (d_1; \dots; d_t)$ of **SQ**. Without loss of generality we can assume that all components of $\mathcal{D}$ are represented by fractions reduced to their least common denominator n, i.e. $d_i = \frac{m_i}{n}$ where $i = 1, \dots, t$ and $(m_1, \dots, m_t) = 1$. Denote $\Pi(\mathcal{D}) = \mathcal{I}(n)$. Let l_j be the greatest common divisor of all numbers $m_1, \dots, m_t$ except the number m_j for $j = 1, \dots, t$. Then we denote by $T(\mathcal{D})$ the set $\{l_1, \dots, l_t\}^{>1}$. It is easy to verify the following proposition.

Proposition 7. *For any $\mathcal{D}$ of* **SQ** *the set $T(\mathcal{D})$ is divisible and relatively prime to n.*

Thus we can consider the set $SG[\Pi(\mathcal{D}); T(\mathcal{D})]$. It is proved in [3] that this set coincides $\langle\{\mathcal{D}\}\rangle$.

Theorem 1. $\langle\{\mathcal{D}\}\rangle = SG[\Pi(\mathcal{D}); T(\mathcal{D})]$.

Now consider the case of an arbitrary finite subset of the set **SQ**. Let $M = \{\mathcal{D}_1, \dots, \mathcal{D}_s\}$ be a finite set of positive stochastic vectors of **SQ**, and $h_1, \dots, h_s$ be respective dimensions of these vectors. As in the case of a set of a single vector, without loss of generality each vector $\mathcal{D}_i$ can be represented in the form

$$\mathcal{D}_i = \left(\frac{m_1^{(i)}}{n_i}, \dots, \frac{m_{h_i}^{(i)}}{n_i} \right), \tag{2}$$

where $(m_1^{(i)}, \dots, m_{h_i}^{(i)}) = 1$. Denote $T(M) = (T(\mathcal{D}_1), \dots, T(\mathcal{D}_s))$ in case of $s \geq 2$ and $T(M) = T(\mathcal{D}_1)$ in case of $s = 1$. By Proposition 7 each set $T(\mathcal{D}_i)$ is divisible and relatively prime to n_i. So by Propositions 7 and 3 we have

Proposition 8. *The set $T(M)$ is divisible and relatively prime to numbers $n_1, \dots, n_s$.*

Denote $\Pi(M) = \bigcup_{i=1}^{s} \mathcal{I}(n_i)$. By Proposition 8 we can consider the set $SG[\Pi(M); T(M)]$. In [3] the following result is obtained.

Theorem 2. *Let M be a finite set of positive stochastic vectors of* **SQ**. *Then $\langle M \rangle = SG[\Pi(M); T(M)]$.*

5 Checking of Generation of Stochastic Vectors by Finite Sets

Using Theorem 2, we can propose an effective algorithm for solving the Generation Problem on **SQ**. According to Theorem 2, for this purpose it is enough to verify if the vector $\mathcal{D}$ is contained in the set $SG[\Pi(M);T(M)]$. First we compute the set $T(M)$. Let M consist of positive vectors $\mathcal{D}_1,\dots,\mathcal{D}_s$ which have dimensions $h_1,\dots,h_s$ respectively. We assume that all components of these vectors are represented initially by fractions in their lowest terms. Let components of the vector $\mathcal{D}_i$, $i=1,\dots,s$, be represented by fractions with denominators $n_1^{(i)},\dots,n_{h_i}^{(i)}$. Denote $\eta_M=\max_{i,j} n_j^{(i)}$, $h_M=\max_i h_i$ and $\Sigma_M=\sum_{i=1}^s h_i$. First of all each vector $\mathcal{D}_i$ is converted to form (2). Note that the denominator n_i of components of $\mathcal{D}_i$ in form (2) is the least common multiple of $n_1^{(i)},\dots,n_{h_i}^{(i)}$. So we can obtain n_i by computing consecutively the numbers $\hat{n}_1^{(i)},\dots,\hat{n}_{h_i-1}^{(i)},\hat{n}_{h_i}^{(i)}=n_i$ where $\hat{n}_1^{(i)}=n_1^{(i)}$ and $\hat{n}_j^{(i)}$ is the least common multiple of $\hat{n}_{j-1}^{(i)}$ and $n_j^{(i)}$, i.e. $\hat{n}_j^{(i)}=\frac{\hat{n}_{j-1}^{(i)}n_j^{(i)}}{(\hat{n}_{j-1}^{(i)},n_j^{(i)})}$, for $j=2,\dots,h_i$. Note that at each step of computing the greatest common divisor of two numbers $a,b\in\mathbb{N}^{>1}$ by means of Euclid's algorithm these numbers are reduced one after another at least half. Thus, for computation of (a,b) it is required to perform no more than $O\Big(1+\log\big(\min(a,b)/(a,b)\big)\Big)$ operations of division of integer numbers. Hence, proceeding from the numbers $\hat{n}_{j-1}^{(i)}$ and $n_j^{(i)}$, we can compute the numbers $(\hat{n}_{j-1}^{(i)},n_j^{(i)})$ and $\hat{n}_j^{(i)}$ by using $O\left(1+\log\frac{n_j^{(i)}}{(\hat{n}_{j-1}^{(i)},n_j^{(i)})}\right)=O\left(1+\log\frac{\hat{n}_j^{(i)}}{\hat{n}_{j-1}^{(i)}}\right)$ arithmetic operations. Thus we need $O(h_i+\log n_i)$ arithmetic operations for computing the number n_i and converting the vector $\mathcal{D}_i$ to form (2). Note that $n_i\le n_1^{(i)}\cdot\ldots\cdot n_{h_i}^{(i)}\le\eta_M^{h_i}$. So $\log n_i\le h_i\log\eta_M$. Taking into account this inequality, we conclude that $O(\Sigma_M\log\Lambda_M)$ arithmetic operations are required in total for converting all vectors $\mathcal{D}_1,\dots,\mathcal{D}_s$ to form (2). Then for any $i=1,\dots,s$ we have to construct the set $T(\mathcal{D}_i)$, i.e. for any $j=1,\dots,h_i$ we have to compute the greatest common divisor of all numbers $m_1^{(i)},\dots,m_{h_i}^{(i)}$ except the number $m_j^{(i)}$. To obtain $(m_2^{(i)},\dots,m_{h_i}^{(i)})$, we compute consecutively the numbers $d_j=(m_2^{(i)},\dots,m_j^{(i)})$ for $j=3,\dots,h_i$. As noted above, proceeding from the numbers $(m_2^{(i)},\dots,m_{j-1}^{(i)})$ and $m_j^{(i)}$, the greatest common divisor d_j of these numbers can be computed by means of Euclid's algorithm, performing no more than $O\left(1+\log\frac{(m_2^{(i)},\dots,m_{j-1}^{(i)})}{(m_2^{(i)},\dots,m_j^{(i)})}\right)$ operations of division of integer numbers. So for computing $d_{h_i}=(m_2^{(i)},\dots,m_{h_i}^{(i)})$ we need in total $O(h_i+\log m_2^{(i)})$ arithmetic operations. Note that every number $m_j^{(i)}$ is less than n_i. Hence the proved inequality implies that every number $m_j^{(i)}$ satisfies the inequality

$$\log m_j^{(i)}<h_i\log\eta_M. \tag{3}$$

Thus $(m_2^{(i)},\dots,m_{h_i}^{(i)})$ can be computed by performing $O(h_i\log\eta_M)=O(h_M\log\eta_M)$ arithmetic operations. In an analogous way we can compute all other numbers of the set $T(\mathcal{D}_i)$. Therefore, the computation of $T(\mathcal{D}_i)$ requires $O(h_ih_M\log\eta_M)$ arithmetic operations. Thus for computing all sets $T(\mathcal{D}_1),\dots,T(\mathcal{D}_s)$ we need $O(\Sigma_Mh_M\log\eta_M)$ arithmetic operations. Proceeding from these sets, we construct the set $T(M)$ by computing consecutively the sets $A_i = \big(T(\mathcal{D}_1), \dots, T(\mathcal{D}_i)\big)$ for $i = 2, \dots, s$. Each set A_i is computed as $\big(A_{i-1}, T(\mathcal{D}_i)\big)$ (where $A_1 = T(\mathcal{D}_1)$). Since $T(\mathcal{D}_i)$ contains no more than h_i numbers, in order to compute $\big(T_{i-1}, T(\mathcal{D}_i)\big)$, for any number of A_{i-1} we have to perform no more than h_i operations of obtaining the greatest common divisor of this number and a number of $T(\mathcal{D}_i)$. Recall that for computing the greatest common divisor (a,b) we need $O\Big(1 + \log\big(\min(a,b)/(a,b)\big)\Big) = O\big(\log\min(a,b)\big)$ arithmetic operations. Therefore, proceeding from the sets A_{i-1} and $T(\mathcal{D}_i)$, we can compute the set A_i by performing $O\left(h_i \sum_{a\in A_{i-1}} \log a\right) = O\left(h_i \log \|A_{i-1}\|\right)$ arithmetic operations. It follows from Proposition 4 that $\|A_{i-1}\| \le \|T(\mathcal{D}_1)\|$. Note that all numbers of the set $\|T(\mathcal{D}_1)\|$ are pairwise relatively prime divisors of either $m_1^{(1)}$ or $m_2^{(1)}$. So $\|T(\mathcal{D}_1)\| \le m_1^{(1)} m_2^{(1)}$. Therefore, taking into account inequality (3), we have

$$\log\|A_{i-1}\| \le \log\|T(\mathcal{D}_1)\| \le \log(m_1^{(1)} m_2^{(1)}) < 2h_1 \log \eta_M \le 2h_M \log \eta_M.$$

Thus, $O(h_i h_M \log \eta_M)$ arithmetic operations are required for computing the set A_i from the sets A_{i-1} and $T(\mathcal{D}_i)$. Therefore, $O(\varSigma_M h_M \log \eta_M)$ arithmetic operations are required for computing the set $T(M) = A_s$ from the sets $T(\mathcal{D}_1), \dots, T(\mathcal{D}_s)$. Thus, we can compute the set $T(M)$ by performing in total $O(\varSigma_M h_M \log \eta_M)$ arithmetic operations.

Now consider the vector $\mathcal{D}$. Let h be the dimension of this vector. If $\mathcal{D} \notin \mathbf{SQ}$, then, obviously, $\mathcal{D} \notin \langle M \rangle$. Assume that $\mathcal{D} \in \mathbf{SQ}$ and all components of $\mathcal{D}$ are represented by fractions in their lowest terms. Let numbers $n_1^{(0)}, \dots, n_h^{(0)}$ be the denominators of these fractions. Denote $\eta_{\mathcal{D}} = \max_j n_j^{(0)}$ and $n = n_1^{(0)} \cdot \ldots \cdot n_h^{(0)}$. If any of numbers $n_1^{(0)}, \dots, n_h^{(0)}$ has a prime divisor which is not contained in $\varPi(M)$, then $\mathcal{D} \notin SG[\varPi(M)]$, so $\mathcal{D} \notin \langle M \rangle$ by Theorem 2. In order to verify that all prime divisors of $n_1^{(0)}, \dots, n_h^{(0)}$ are contained in $\varPi(M)$, it is enough to check the relation

$$\mathcal{I}(n) \subseteq \mathcal{I}(n_1 \cdot \ldots \cdot n_s). \tag{4}$$

This relation is valid if and only if $\eta(\mathcal{D})$ is a divisor of some great enough power of the number $n_1 \cdot \ldots \cdot n_s$. For such a power we can, obviously, take any power with an exponent no less than $\log_2 \eta(\mathcal{D})$. In particular, we can take a power obtained by applying the operation of raising a number to square power $\lceil \log_2 \log_2 \eta(\mathcal{D}) \rceil$ times consecutively to the number $n_1 \cdot \ldots \cdot n_s$. Thus, relation (4) can be checked by performing $O(h + s + \log\log n)$ arithmetic operations. From $n \le \eta_{\mathcal{D}}^h$ we have $\log\log n \le \log h + \log\log \eta_{\mathcal{D}}$. Thus, the number of arithmetic operations required for checking if all prime divisors of $n_1^{(0)}, \dots, n_h^{(0)}$ are contained in $\varPi(M)$ can be estimated by $O(h + s + \log\log \eta_{\mathcal{D}})$. If this checking is positive we verify that $\mathcal{D}$ is contained in $SG[\varPi(M); T(M)]$. As noted in Section 3, we can take the number n as a common denominator of components of $\mathcal{D}$. In this case, let μ_i be the sum of numerators of the first i components of $\mathcal{D}$, $i = 1, \dots, h-1$. It is enough, obviously, to perform $O(h)$ arithmetic operations for computing $\mu_1, \dots, \mu_{h-1}, n - \mu_1, \dots, n - \mu_{h-1}$. Then, in order to construct the subsets $T_1 \ldots, T_{h-1}$ corresponding to the vector $\mathcal{D}$, we have to check for any number t from $T(M)$ the divisibility of the numbers μ_i and $n-\mu_i$, where $i = 1, \ldots, h-1$, by t (if μ_i is divisible by t, then $t \in T_i$; if $n - \mu_i$ is divisible by t, then $t \in T(M) \setminus T_i$; if neither μ_i, nor $n - \mu_i$ is divisible by t, then we conclude that $\mathcal{D} \notin SG[\Pi(M); T(M)]$). It requires $O(h|T(M)|)$ arithmetic operations. No more than $O(h|T(M)|)$ operations are additionally required for checking the relations $T_1 \supseteq T_2 \supseteq \ldots \supseteq T_{h-1}$. Summing all stages of the proposed algorithm of checking the relation $\mathcal{D} \in SG[\Pi(M); T(M)]$ and taking into account that $s < \Sigma_M$, we conclude that this algorithm requires to perform $O\big(\Sigma_M h_M \log \eta_M + h(1 + |T(M)|) + \log\log \eta_{\mathcal{D}}\big)$ operations. Note that $|T(M)| \le \log_2 \|T(M)\|$ and the proved estimation $\log \|A_i\| < 2h_M \log \eta_M$ is valid for the set $T(M) = A_s$, i.e. $\log \|T(M)\| < 2h_M \log \eta_M$. Thus $|T(M)| = O(h_M \log \eta_M)$. Using this estimation for $|T(M)|$, we finally obtain

Theorem 3. *If all components of the vector $\mathcal{D}$ and vectors of the set M are represented by fractions in their lowest terms, then for checking the relation $\mathcal{D} \in \langle M \rangle$ it is enough to perform $O\big((\Sigma_M + h)h_M \log \eta_M + \log\log \eta_{\mathcal{D}}\big)$ arithmetic operations.*

Note that the value $(\Sigma_M + h)h_M \log \eta_M + \log\log \eta_{\mathcal{D}}$ is polynomially bounded by the size of input for the Generation Problem on **SQ**. So Theorem 3 implies

Corollary 2. *The Generation Problem on* **SQ** *can be solved in polynomial time.*

6 Closures of Infinite Subsets of SQ

The results of Section 4 can be generalized to the case of infinite subsets of the set **SQ**. Let $M = \{\mathcal{D}_1, \mathcal{D}_2, \ldots\}$ be an infinite set of positive stochastic vectors of **SQ**. In the same way as in Section 4, for each vector $\mathcal{D}_i$ we assume that all components of this vector are represented by fractions reduced to their least common denominator n_i and define the set $T(\mathcal{D}_i)$. Denote $T(M) = (T(\mathcal{D}_1), T(\mathcal{D}_2), \ldots)$. Propositions 7 and 5 imply

Proposition 9. *$T(M)$ is a finite divisible set which is relatively prime to numbers $n_1, n_2, \ldots$.*

Denote by $\Pi(M)$ the set $\bigcup_{i=1}^{\infty} \mathcal{I}(n_i)$. By Proposition 9 we can again consider the set $SG[\Pi(M); T(M)]$. The following generalization of Theorem 2 to the case of infinite subsets of the set **SQ** is proved in [4].

Theorem 4. *Let $M = \{\mathcal{D}_1, \mathcal{D}_2, \ldots\}$ be an infinite set of positive stochastic vectors of* **SQ**. *Then $\langle M \rangle = SG[\Pi(M); T(M)]$.*

This theorem allows to give a description of all closed subsets of the set **SQ**. Since each of these subsets is a closure of some set of vectors of **SQ** (for example, of the subset itself), it is follows from Theorems 2 and 4 and Proposition 2 that each of these subsets is an element of the set **SG**. Hence, taking into account Lemma 1, we obtain the following statement.

Theorem 5. **SG** *is the set of all closed subsets of the set* **SQ**.

7 Finitely Generated Closed Subsets of SQ

Among closed classes of stochastic vectors, the classes generated by finite subsets are most important in practice. So the problem of a description of all finitely generated closed subsets of the set $\mathbf{SQ}$ is of great interest. Denote by $\mathbf{SG_{fin}}$ the subset of $\mathbf{SG}$ which consists of all sets $SG[\Pi; T]$ such that Π is finite. It follows from Theorem 2 and Proposition 2 that all finitely generated closed subsets of the set $\mathbf{SQ}$ are elements of the set $\mathbf{SG_{fin}}$. On the other hand, it is proved in [4] that any set belonging to $\mathbf{SG_{fin}}$ is finitely generated. Thus we have

Theorem 6. $\mathbf{SG_{fin}}$ *is the set of all finitely generated closed subsets of the set* $\mathbf{SQ}$.

References

[1] R. Bukharaev. *Foundations of the Theory of Probabilistic Automata.* Moscow: Nauka, 1985 (in Russian).

[2] R. Kolpakov. Classes of Binary Rational Distributions Closed under Discrete Transformations. *Lecture Notes in Computer Science*, Springer Verlag, 2003, 2827:157–166.

[3] R. Kolpakov. On discrete transformations of finite distributions with rational probabilities. *Matematicheskie voprosy kibernetiki*, Moscow: Nauka, 2003, 12:109–146 (in Russian).

[4] R. Kolpakov. Closed classes of finite distributions of rational probabilities. *Diskretnyj analiz i issledovanie operacij*, Ser. 1, Novosibirsk, 2004, 11(3):16–31 (in Russian).

[5] N. Nurmeev. On Boolean functions with variables having random values. *Proceedings of VIII USSR Conference "Problems of Theoretical Cybernetics"*, Gorky, 1988, 2:59–60 (in Russian).

[6] F. Salimov. To the question of modelling Boolean random variables by functions of logic algebra. *Verojatnostnye metody i kibernetika*, Kazan: Kazan State University, 1979, 15:68–89 (in Russian).

[7] F. Salimov. Finite generativeness of some algebras over random values. *Voprosy kibernetiki*, Moscow, 1982, 86:122–130 (in Russian).

[8] F. Salimov. On maximal subalgebras of algebras of distributions. *Izvestija vuzov*, Ser. Matematika, 1985, 7:14–20 (in Russian).

[9] F. Salimov. On one family of distribution algebras. *Izvestija vuzov*, Ser. Matematika, 1988, 7:64–72 (in Russian).

[10] R. Skhirtladze. On the synthesis of p-circuits of contacts with random discrete states. *Soobschenija AN GrSSR*, 1961, 26(2):181–186 (in Russian).

[11] R. Skhirtladze. On a method of constructing Boolean random variable with a given probability distribution. *Diskretnyj analiz*, Novosibirsk, 1966, 7:71–80 (in Russian).

[12] A. Srinivasan and D. Zuckerman. Computing with very weak random sources. *SIAM J. on Computing*, 1999, 28(4):1433–1459.

FPL Analysis for Adaptive Bandits

Jan Poland[*]

Grad. School of Inf. Sci. and Tech.,
Hokkaido University, Japan
jan@ist.hokudai.ac.jp
http://www-alg.ist.hokudai.ac.jp/~jan

Abstract. A main problem of "Follow the Perturbed Leader" strategies for online decision problems is that regret bounds are typically proven against oblivious adversary. In partial observation cases, it was not clear how to obtain performance guarantees against adaptive adversary, without worsening the bounds. We propose a conceptually simple argument to resolve this problem. Using this, a regret bound of $O(t^{\frac{2}{3}})$ for FPL in the adversarial multi-armed bandit problem is shown. This bound holds for the common FPL variant using only the observations from designated exploration rounds. Using all observations allows for the stronger bound of $O(\sqrt{t})$, matching the best bound known so far (and essentially the known lower bound) for adversarial bandits. Surprisingly, this variant does not even need explicit exploration, it is self-stabilizing. However the sampling probabilities have to be either externally provided or approximated to sufficient accuracy, using $O(t^2 \log t)$ samples in each step.

1 Introduction

"Expert Advice" stands for an active research area which studies online algorithms. In each time step $t = 1, 2, 3, \ldots$ the master algorithm, henceforth called *master* for brevity, is required to commit to a decision, which results in some cost. The master has access to a class of *experts*, each of which suggests a decision at each time step. The goal is to design master algorithms such that the *cumulative regret* (which is just the cumulative excess cost) with respect to any expert is guaranteed to be small. Bounds on the regret are typically proven *in the worst case*, i.e. without any statistical assumption on the process assigning the experts' costs. In particular, this might be an *adaptive adversary* which aims at maximizing the master's regret and also knows the master's internal algorithm. This implies that (unless the decision space is continuous and the cost function is convex) the master must *randomize* in order to protect against this danger.

In the recent past, a growing number of different but related online problems have been considered. Prediction of a binary sequence with expert advice has been popular since the work of Littlestone and Warmuth in the early 1990's. Freund and Schapire [1] removed the structural assumption on the decision space

[*] This work was supported by JSPS 21st century COE program C01.

and gave a very general algorithm called Hedge which in each time step randomly picks one expert and follows its recommendation. We will refer to this setup as the *online decision problem.* Auer et al. [2,3] considered the first *partial observation* case, namely the bandit setup, where in each time step the master algorithm only learns its own cost, i.e. the cost of the selected expert. All these and many other papers are based on *weighted forecasting* algorithms.

A different approach, *Follow the Perturbed Leader* (FPL), was pioneered as early as 1957 by Hannan [4] and rediscovered recently by Kalai and Vempala [5]. Compared to weighted forecasters, FPL has two main advantages and one major drawback. First, it applies to the online decision problem and admits a much more elegant analysis for *adaptive learning rate* [6]. Even infinite expert classes do not cause much complication. (However, the leading constant of the regret bound is generically a factor of $\sqrt{2}$ worse than that for weighted forcasters.) Adaptive learning rate is necessary unless the total number of time steps to be played is known in advance.

As a second advantage, FPL also admits efficient treatment of cases where the expert class is potentially huge but has a linear structure [7,8]. We will refer to such problems as *geometric online optimization.* An example is the online shortest path problem on a graph, where the set of admissible paths = experts is exponential in the number of vertices, but the cost of each path is just the sum of the costs of the vertices.

FPL's main drawback is that its general analysis only applies against an *oblivious* adversary, that is an adversary that has to decide on *all* cost vectors before the game starts – as opposed to an adaptive one that before each time step t just needs to commit to the current cost vector. For the full information game, one can show that a regret bound against oblivious adversary implies the *same* bound against an adaptive one [6]. The intuition is that FPL's current decision at time t does not depend on its past decisions. Therefore, the adversary may well decide on the current cost vector before knowing FPL's previous decisions. This argument does not apply in partial observation cases, as there FPL's behavior does depend on its past decisions (because the observations do so). As a consequence, authors started to explicitly distinguish between oblivious and adaptive adversary, sometimes restricting to the former, sometimes obtaining bounds of lower quality for the latter. E.g. McMahan and Blum [7] suggest a workaround, proving sublinear regret bounds against an adaptive bandit, however of worse order ($t^{\frac{3}{4}}\sqrt{\log t}$ instead of $t^{\frac{2}{3}}$, for both, geometric online optimization and online decision problem). This is not satisfactory, since in case of the bandit online decision problem for a suitable weighted forecaster, even a $O(\sqrt{t})$ bound against adaptive adversary is known [3].

In this work, we remove FPL's major drawback. We give a simple argument (Section 2) which shows that also in case of partial observation, a bound for FPL against an oblivious adversary implies the same bound for adaptive adversary. This will allow in particular to prove a $O\big((tn\sqrt{\log n})^{\frac{2}{3}}\big)$ bound for the bandit online decision problem (Section 3). This bound is shown for the common construction where only the observations of designated exploration rounds are used. As this master algorithm is label efficient, the bound is essentially sharp. In contrast, using all informations will enable us to prove a stronger $O(\sqrt{tn\log n})$ bound (Section 4). This matches the best bound known so far for the adversarial bandit problem [3], which is sharp within $\sqrt{\log n}$. The downside of this algorithm is that either the sampling probabilities have to be given by an oracle, or they have to be approximated with to sufficient accuracy, using $O(t^2\log t)$ samples.

2 FPL: Oblivious $\Rightarrow$ Adaptive

Assume that $c_1, c_2, \ldots \in [0,1]^n$ is a sequence of cost vectors. There are $n \geq 1$ experts. That is, c_t^i is expert i's cost at time t, and the costs are bounded (w.l.o.g. in $[0,1]$). In the full observation game, at time t the master would know the past cumulative costs $c_{<t} = c_{1:t-1} = \sum_{s=1}^{t-1} c_s$ (observe that we have introduced some notation here). However, our focus are *partial observations* where this is not the case. Hence, assume that there are *estimates* $\hat{c}_t$ (to be specified later) for the cost vectors c_t. Then at time t, FPL(t) samples a perturbation vector $q_t \in [0,\infty)^n$ the components of which are independently exponentially distributed, that is, $\mathbf{P}(q_t^i \geq x) = e^{-x}$. Afterwards, the expert with the best (minimum) score $\hat{c}_{<t} - \frac{q_t}{\eta_t}$ is selected, where $\eta_t > 0$ is the *learning rate*:

$$\mathrm{FPL}(t, \hat{c}_{<t}) = \arg\min_{1\leq i\leq n}\left\{\hat{c}_{<t}^i - \tfrac{q_t^i}{\eta_t}\right\} \text{ where } q_t^i \overset{d.}{\sim} \mathrm{Exp} \text{ independently.} \tag{1}$$

Denote the expert FPL chooses at time t by $I_t = \mathrm{FPL}(t, \hat{c}_{<t})$. Then an adaptive adversary is a function $A : [0,1]^{n\times t-1} \times \{1\ldots n\}^{t-1} \to [0,1]^n$. (We assume A to be deterministic but remark that all our results and proofs hold for randomized A without major modification.) The complete game between FPL and A is specified by $c_t = A(c_1c_2\ldots c_{t-1}, I_1I_2\ldots I_{t-1})$ and $I_t = \mathrm{FPL}(t, \hat{c}_{<t})$ for $t = 1, 2, \ldots$ The estimated cost vector $\hat{c}_t$ is revealed to FPL after time t and specified by a mechanism "outside" this game which is defined later (this is the exploration).

After the game has proceeded for a number of time steps T, we want to evaluate FPL's performance. Actually, the *expected* performance is the right quantity to address. If we are rather interested in high probability bounds on the actual performance, then they are easily obtained by observing that the difference of actual to expected performance is a martingale with bounded differences (all instantaneous costs c_t^i are in $[0,1]$). Thus, high probability bounds follow by Azuma's inequality, as we will demonstrate in Proposition 2.

How can we compute FPL's expected costs $\mathbf{E}c_{1:T}^{\mathrm{FPL}} = \mathbf{E}\sum_{t=1}^{T} c_t^{I_t}$? The key observation is that – on the cost vectors generated by FPL and A and with the given estimated costs $\hat{c}_t$ – FPL's expected costs at time t are the same as another algorithm $\widetilde{\mathrm{FPL}}$'s expected costs. $\widetilde{\mathrm{FPL}}$ is defined by

$$\widetilde{\mathrm{FPL}}(t, \hat{c}_{<t}) = \arg\min_{1\leq i\leq n}\left\{\hat{c}_{<t}^i - \tfrac{q_*^i}{\eta_t}\right\}, \tag{2}$$

where q_* is a *single fixed* vector with independently exponentially distributed components. Since we have to be careful to take expectations w.r.t. the appropriate randomness, we explicitely refer to the randomness in the notation by writing e.g. $\mathbf{E}c_t^{\mathrm{FPL}} = \mathbf{E}_{q_t} c_t^{\mathrm{FPL}}$. Then the following statement trivially holds, as q_t and q_* have the same distribution.

Proposition 1. *At each time $t \leq T$, we have $\mathbf{E}_{q_t} c_t^{\mathrm{FPL}} = \mathbf{E}_{q_*} c_t^{\widetilde{\mathrm{FPL}}}$.*

This means that in order to analyze FPL, we may now proceed by considering the expected costs of $\widetilde{\mathrm{FPL}}$ instead. We can use the standard analysis based on the tools by Kalai and Vempala [5], which requires that $\widetilde{\mathrm{FPL}}$ is executed on a sequence of cost vectors that is fixed and not known in advance. Actually, in contrast to the full observation game analysis, the bandit analysis will *never* require the true cost vectors to be revealed, but rather the estimated cost vectors. For the cost vectors generated by A in response to FPL, the prerequisite for $\widetilde{\mathrm{FPL}}$ is satisfied – just consider $\widetilde{\mathrm{FPL}}$ as a *virtual* or *hypothetic* algorithm which is not actually executed. Therefore it does not make any decisions or cause any response from the adversary. Just for the sake of analysis we *pretend* that it runs and evaluate the expected cost it incurs, which is the same as FPL.

Since our key argument and the way it is used in the analysis appears quite subtle at the first glance, we encourage the reader to thoroughly verify each of the subsequent formal steps.

3 The Standard Strategy Against Adversarial Bandits

The first algorithm we consider, *bandit-FPL* (bFPL), is specified in Figure 1 and proceeds as follows. At time t, it decides if to perform an exploration or an exploitation step according to some *exploration probability* $\gamma_t \in (0,1)$. This is realized by sampling $r_t \in \{0,1\}$ independently from all other randomness with $P[r_t = 1] = \gamma_t$. In case of exploration ($r_t = 1$), the decision I_t^b is uniformly sampled from $\{1 \dots n\}$, independently from all other randomness. We denote this choice by u_t. (For notational convenience, we will also refer to the irrelevant u_t's in the exploitations steps later.) In case of exploitation ($r_t = 0$), bFPL obtains its decision I_t^b by invoking FPL according to (1). After bFPL has played its decision, it observes its own costs $c_t^{I_t^b}$. Finally, only in case of exploration ($r_t = 1$), the estimated cost vector is set to something different from 0. This is the *standard way* of constructing an FPL variant against an adversarial bandit

```
For $t = 1, 2, 3, \dots$
    set $\hat{c}_t^i = 0$ for all $i$
    sample $r_t \in \{0,1\}$ independently s.t. $P[r_t = 1] = \gamma_t$
    If $r_t = 0$ Then set $I_t^b = \mathrm{FPL}(t, \hat{c}_{<t})$ according to (1)
    If $r_t = 1$ Then sample $I_t^b$ from $\{1 \dots n\}$ uniformly ($I_t^b = u_t$)
    play decision $I_t^b$ and observe cost $c_t^{I_t^b}$
    If $r_t = 1$ Then set $\hat{c}_t^{I_t^b} = n \cdot c_t^{I_t^b} / \gamma_t$
```

Fig. 1. The algorithm bFPL. The exploration rate γ_t and the learning rate η_t (used by subroutine FPL) will be specified in Theorem 1.

[7,8]. We will discuss how to make use of *all* observations in the next section. Here is the formal specification of the algorithm again.

$$I_t^b = \mathrm{bFPL}(t, \hat{c}_{<t}) = \begin{cases} u_t & \text{if } r_t = 1 \\ \mathrm{FPL}(t, \hat{c}_{<t}) & \text{otherwise,} \end{cases} \qquad \hat{c}_t^i = \begin{cases} \frac{nc_t^i}{\gamma_t} & \text{if } r_t = 1 \wedge i = I_t^b \\ 0 & \text{otherwise.} \end{cases}$$

Consequently, the estimated cost vector is chosen *unbiasedly*, i.e. $\mathbf{E}_{r_t,u_t}\hat{c}_t^i = c_t^i$. This technique was introduced in [2].

Theorem 1. *Let* $\gamma_t = \min\{1, t^{-\frac{1}{3}}(n\sqrt{\log n})^{\frac{2}{3}}\}$ *and* $\eta_t = \frac{\gamma_t}{n^2} t^{-\frac{1}{3}}(n\sqrt{\log n})^{\frac{2}{3}}$. *Then, for any* $T \geq (n \log n)^2$, *each expert* $i \in \{1 \ldots n\}$, *and* any *adaptive assignment of the costs* $c_1, c_2, \ldots$, *bFPL satisfies the regret bound*

$$\mathbf{E}c_{1:T}^{\mathrm{bFPL}} - c_{1:T}^i \leq 4\Big(Tn\sqrt{\log n}\Big)^{\frac{2}{3}}. \tag{3}$$

(For $T < (n \log n)^2$, *the regret is clearly at most* $(n \log n)^2$.*)*

Proof. All computations we use in the subsequent proof have been taken or adapted from other work. Our point is to bring them into the right order and to carefully check that in this context, against an adaptive adversary, all operations are legitimate. In particular we have to take care that all expectations are w.r.t. the appropriate randomness. Again, we make this explicit in the notation and write e.g. $\mathbf{E}c_t^{\mathrm{bFPL}} = \mathbf{E}_{q_t,r_{1:t},u_{1:t}}c_t^{\mathrm{bFPL}}$. Note that according to the definition of bFPL, $\mathbf{E}c_t^{\mathrm{bFPL}}$ in fact does *not* depend on $q_{<t}$. During the proof, we will avoid the use of unspecified expectation (without subscripts). Let's introduce abbreviation $h_{<t} = (r_{<t}, u_{<t}, q_{<t})$ for the randomization history, i.e. the tuple containing all past random variables.

Moreover, we will use *conditional expectation.* For instance, $\mathbf{E}_{q_t}[c_t^{\mathrm{FPL}}|h_{<t}]$ denotes a random variable depending on the randomization history $h_{<t}$, where for each possible history the expectation is taken w.r.t. q_t. Since we admit adaptive assignments, we must be aware that they may depend on bFPL's past randomness. To make this explicit, we use the notation $\mathbf{E}[c_t^i|h_{<t}]$ for the adversary's decisions and rewrite our bound to show (3) as

$$\sum_{t=1}^{T} \mathbf{E}_{q_t,r_t,u_t}[c_t^{\mathrm{bFPL}}|h_{<t}] - \sum_{t=1}^{T} \mathbf{E}[c_t^i|h_{<t}] \leq 4\Big(Tn\sqrt{\log n}\Big)^{\frac{2}{3}}. \tag{4}$$

In order to keep the presentation simple, we assume the adversary to be deterministic. Then for given randomization history, c_t^i is constant. The same proof (and hence the theorem) remains valid if we admit randomized adversaries.

First note that $\mathbf{E}_{q_t,r_t,u_t}[c_t^{\mathrm{bFPL}}|h_{<t}] \leq \mathbf{E}_{q_t}[c_t^{\mathrm{FPL}}|h_{<t}] + \gamma_t$ holds in each time step t by definition of bFPL and $c_t^{I_t^b} \leq 1$. Since $\gamma_t \leq t^{-\frac{1}{3}}(n\sqrt{\log n})^{\frac{2}{3}}$, we have

$$\sum_{t=1}^{T} \gamma_t \leq \sum_{t=1}^{T} t^{-\frac{1}{3}}(n\sqrt{\log n})^{\frac{2}{3}} \leq \tfrac{3}{2}\Big(Tn\sqrt{\log n}\Big)^{\frac{2}{3}}. \tag{5}$$

Therefore, (4) follows from

$$\sum_{t=1}^{T} \mathbf{E}_{q_t}[c_t^{\mathrm{FPL}}|h_{<t}] - \sum_{t=1}^{T} \mathbf{E}[c_t^i|h_{<t}] \leq \tfrac{5}{2}\Big(Tn\sqrt{\log n}\Big)^{\frac{2}{3}}. \tag{6}$$

Consider this form of FPL (i.e. FPL executed in each time step) as a *virtual* algorithm: It does not run in that way on the inputs. Rather, for the sake of analysis, we pretend that it runs with the $\hat{c}_t$ obtained from bFPL and try to evaluate its (virtual) performance.

We then use Proposition 1 to bring into the play another virtual algorithm, namely $\widetilde{\mathrm{FPL}}$. Since for given randomization history, the expected performance of FPL and $\widetilde{\mathrm{FPL}}$ coincide, (6) is proven if we can show

$$\sum_{t=1}^{T} \mathbf{E}_{q_*}[c_t^{\widetilde{\mathrm{FPL}}}|h_{<t}] - \sum_{t=1}^{T} \mathbf{E}[c_t^i|h_{<t}] \leq \tfrac{5}{2}\Big(Tn\sqrt{\log n}\Big)^{\frac{2}{3}}. \tag{7}$$

Next, we perform the transition from real to estimated costs. Since the estimate $\hat{c}$ was defined to be unbiased, we have $\mathbf{E}[c_t^i|h_{<t}] = \mathbf{E}_{r_t,u_t}[\hat{c}_t^i|h_{<t}]$. By the same argument, since the choice of $\widetilde{\mathrm{FPL}}$ actually does not depend on r_t and u_t, $\mathbf{E}_{q_*}[c_t^{\widetilde{\mathrm{FPL}}}|h_{<t}] = \mathbf{E}_{q_*,r_t,u_t}[\hat{c}_t^{\widetilde{\mathrm{FPL}}}|h_{<t}]$ holds. Hence, (7) follows from

$$\sum_{t=1}^{T} \mathbf{E}_{q_*,r_t,u_t}[\hat{c}_t^{\widetilde{\mathrm{IFPL}}}|h_{<t}] - \sum_{t=1}^{T} \mathbf{E}_{r_t,u_t}[\hat{c}_t^i|h_{<t}] \leq \tfrac{5}{2}\Big(Tn\sqrt{\log n}\Big)^{\frac{2}{3}}. \tag{8}$$

Note that, somewhat curiously, $\widetilde{\mathrm{FPL}}$ (like FPL) only incurs estimated costs in case of exploration, i.e. where it actually did not decide the action. We need yet another virtual algorithm, *infeasible* $\widetilde{\mathrm{FPL}}$ or $\widetilde{\mathrm{IFPL}}$, defined as

$$\widetilde{\mathrm{IFPL}}(t, \hat{c}_{1:t}) = \arg\min_{1\leq i\leq n}\Big\{\hat{c}_{1:t}^i - \tfrac{q_*^i}{\eta_t}\Big\}, \tag{9}$$

which uses the same perturbation q_* as $\widetilde{\mathrm{FPL}}$. It is not feasible because at time t it makes use of the information $\hat{c}_t$, which is only available afterwards. As it is a virtual algorithm, this does not cause any problems. By [6, Theorem 4], which is proven by an argument very similar to (13) below, in case of exploration (i.e. $r_t = 1$) it holds that $\mathbf{E}_{q_*}[\hat{c}_t^{\widetilde{\mathrm{FPL}}}|h_{<t}, r_t = 1] \leq \mathbf{E}_{q_*}[\hat{c}_t^{\widetilde{\mathrm{IFPL}}}|h_{<t}, r_t = 1] + \eta_t\big(\frac{n}{\gamma_t}\big)^2$. We remark that this step is valid also for independently sampled perturbations q_t. Clearly, $\mathbf{E}_{q_*}[\hat{c}_t^{\widetilde{\mathrm{FPL}}}|h_{<t}, r_t = 0] = \mathbf{E}_{q_*}[\hat{c}_t^{\widetilde{\mathrm{IFPL}}}|h_{<t}, r_t = 0]$ in case of exploitation ($r_t = 0$). Thus in expectation w.r.t. q_* and r_t, and for any u_t,

$$\mathbf{E}_{q_*}[\hat{c}_t^{\widetilde{\mathrm{FPL}}}|h_{1:T}] = \mathbf{E}_{q_*,r_t}[\hat{c}_t^{\widetilde{\mathrm{FPL}}}|h_{<t}] \leq \mathbf{E}_{q_*,r_t}[\hat{c}_t^{\widetilde{\mathrm{IFPL}}}|h_{<t}] + \tfrac{\eta_t n^2}{\gamma_t}.$$

The sum over $\frac{\eta_t n^2}{\gamma_t} \leq t^{-\frac{1}{3}}\big(n\sqrt{\log n}\big)^{\frac{2}{3}}$ is bounded as in (5), and we see that (8) holds if we can show

$$\sum_{t=1}^{T} \mathbf{E}_{q_*,r_t,u_t}[\hat{c}_t^{\widetilde{\mathrm{IFPL}}}|h_{<t}] - \sum_{t=1}^{T} \mathbf{E}_{r_t,u_t}[\hat{c}_t^i|h_{<t}] \leq \Big(Tn\sqrt{\log n}\Big)^{\frac{2}{3}}. \tag{10}$$

The rest of the proof now follows as in [5] or [6]. In order to maintain self-containedness, we give it here. Actually we verify (10) for *any* choice of $r_{1:T}, u_{1:T}$, then it also holds in expectation.

In the following, we suppress the dependency on $r_{1:T}, u_{1:T}$ in the notation. Then all expectations are w.r.t. q_*. We use the following convenient notation from [5]: For a vector $x \in \mathbb{R}^n$, let $M(x)$ be the unit vector which has a 1 at the index $\arg\min_i\{x^i\}$ and 0's at all other places. Then the process of selecting a minimum can be written as scalar product: $\min_i\{x^i\} = M(x)\circ x$. For convenience, let $\eta_0 = \infty$ and $\tilde{c}_{1:t} = \hat{c}_{1:t} - \frac{q_*}{\eta_t}$. Then it is easy to prove by induction [5,6] that

$$\hat{c}_{1:t}^{\widetilde{\text{IFPL}}} - \sum_{t=1}^{T} M(\tilde{c}_{1:t})\circ q_*\Big(\tfrac{1}{\eta_t} - \tfrac{1}{\eta_{t-1}}\Big) \sum_{t=1}^{T} M(\tilde{c}_{1:t})\circ\tilde{c}_t \le M(\tilde{c}_{1:T})\circ\tilde{c}_{1:T}. \tag{11}$$

In order to estimate $\mathbf{E}\hat{c}_{1:t}^{\widetilde{\text{IFPL}}}$, we take expectations on both sides. Then observe $\mathbf{E}M(\tilde{c}_{1:T})\circ\tilde{c}_{1:T} \le \mathbf{E}M(\hat{c}_{1:T})\circ\tilde{c}_{1:T} = \min_j\{\hat{c}_{1:T}^j\} - \frac{\mathbf{E}M(\hat{c}_{1:T})\circ q_*}{\eta_T} \le \hat{c}_{1:T}^i - \frac{1}{\eta_T}$ by definition of M. The negative term on the l.h.s. of (11) may be bounded by $\sum_{t=1}^T M(\tilde{c}_{1:t})\circ q_*\Big(\frac{1}{\eta_t} - \frac{1}{\eta_{t-1}}\Big) \le \sum_{t=1}^T M(-q_*)\circ q_*\Big(\frac{1}{\eta_t} - \frac{1}{\eta_{t-1}}\Big) = \frac{\max_i\{q_*^i\}}{\eta_T} \le \frac{1+\log n}{\eta_T}$ (see [5] or [6] for the last estimate). Plugging these estimates back into (11) while observing $\frac{1}{\eta_t} - \frac{1}{\eta_{t-1}} \ge 0$ and $\eta_T = T^{-\frac{2}{3}}\big(\frac{\log n}{n}\big)^{\frac{2}{3}}$ (which holds because of $T \ge (n\log n)^2$), finally shows (10) and concludes the proof of the theorem. □

Proposition 2. *(High probability bound) For each $T \ge 1$ and $0 \le \delta \le 1$, the actual costs of bFPL are bounded with probability at least $1-\delta$ by*

$$c_{1:T}^{\text{bFPL}} \le \mathbf{E}c_{1:T}^{\text{bFPL}} + \sqrt{2T\log\tfrac{2}{\delta}}.$$

Proof. Again we use the explicit notation from the proof of the previous theorem. It is easy to see that the sequence of random variables $X_T = \sum_{t=1}^T \big(c_t^{\text{bFPL}} - \mathbf{E}_{r_t,u_t,q_t}[c_t^{\text{bFPL}}|h_{<t}]\big)$ is a martingale w.r.t. the filter of sigma-algebras generated by the randomization history $h_{1:t}$. Moreover, its differences are bounded by $|X_t - X_{t-1}| \le 1$. Consequently, by Azuma's inequality, the probability that X_t exceeds some $\lambda > 0$ is bounded by $\delta = 2\exp\big(-\frac{\lambda^2}{2T}\big)$. Solve this for λ to obtain the assertion. □

4 Using All Observations

The algorithm bFPL considered so far does only uses a γ-fraction of all the input. It is thus a *label efficient* decision maker [9,10]. One possible way to specify a label efficient problem setup is to require that the master usually does not observe anything, and it incurs maximal cost if it decides to observe something [10]. Since just before (5), we upper bounded the costs in case of exploration by 1, it is immediate that the same analysis and hence also Theorem 1 transfer to the label efficient case. [10, Sec. 5] prove that there is a label efficient prediction problem such that *any* forecaster incurs a regret proportional to $t^{\frac{2}{3}}$. Hence the bound in Theorem 1 is essentially sharp for bFPL.

Of course, the usual bandit setup does not require the master to make use of only a tiny fraction of all information available. For weighted forecasters, it is very easy to produce an unbiased cost estimate if each round's inputs are used. It turns out that then regret bound proportional to $\sqrt{t}$ can be obtained [3]. Unfortunately this is different for FPL, as here the sampling probabilities are not explicitely available. In the following, we will first discuss the computationally infeasible case assuming that we know the sampling probabilities. After that, we show how to approximate them by a Monte Carlo simulation to sufficient accuracy.

Surprisingly, it is possible to work with the plain FPL algorithm from (1), without exploration. We just have to use the correct estimated cost vectors,

$$\hat{c}_t^i = \begin{cases} c_t^i / \mathbf{P}(I_t^{\mathrm{FPL}} = i) & \text{if } i = I_t^{\mathrm{FPL}} \\ 0 & \text{otherwise,} \end{cases} \tag{12}$$

where I_t^{FPL} was FPL's choice at time t. We assume that the values $\mathbf{P}(I_t^{\mathrm{FPL}} = i)$ are provided by some oracle.

It is not hard to adapt the proof of Theorem 1 to analyze FPL under these conditions. As in the steps up to (8),

$$\mathbf{E}_{q_t}[c_t^{\mathrm{FPL}} | h_{<t}] = \mathbf{E}_{q_*,q_t,r_t,u_t}[\hat{c}_t^{\widetilde{\mathrm{FPL}}} | h_{<t}] \mathbf{E}_{q_*,q_t,r_t,u_t}[\hat{c}_t^{\widetilde{\mathrm{FPL}}(q_*)}(q_t) | h_{<t}].$$

The overly explicit notation $\hat{c}_t^{\widetilde{\mathrm{FPL}}(q_*)}(q_t)$ serves to remind that the cost vector estimated is obtained using q_t, while $\widetilde{\mathrm{FPL}}$'s choice incurring cost stems from q_*. It is essential that q_t and q_* are independent. Observe that in general, $\mathbf{E}_{q_*,q_t,r_t,u_t}[\hat{c}_t^{\widetilde{\mathrm{FPL}}(q_*)}(q_t) | h_{<t}] \lneq \mathbf{E}_{q_t,r_t,u_t}[\hat{c}_t^{\widetilde{\mathrm{FPL}}(q_t)}(q_t) | h_{<t}]$: the latter quantity, which is the actual estimated cost of $\widetilde{\mathrm{FPL}}$'s choice, is biased and too large.

Abbreviate $p^i = \mathbf{P}(I_t^{\widetilde{\mathrm{FPL}}} = i)$ and $\pi^i = \mathbf{P}(I_t^{\widetilde{\mathrm{IFPL}}} = i)$. Denote the exponential distribution by μ and integration with respect to $q^1 \dots q^n$ without the ith coordinate by $\int \dots d\mu(q^{\neq i})$. Moreover, for $x \in \mathbb{R}$, let $x^+ = \max\{x, 0\}$. Then, similarly to the proof of [6, Theorem 4],

$$\begin{aligned} p_i &= \int \int_{\max\limits_{j \neq i}\{\eta_t(\hat{c}_{<t}^i - \hat{c}_{<t}^j) + q^j\}}^{\infty} d\mu(q^i) d\mu(q^{\neq i}) \int e^{-(\max_{j \neq i}\{\eta_t(\hat{c}_{<t}^i - \hat{c}_{<t}^j) + q^j\})^+} d\mu(q^{\neq i}) \qquad (13) \\ &\leq \int e^{\frac{\eta_t}{p^i}} e^{-(\max_{j \neq i}\{\eta_t(\hat{c}_{<t}^i - \hat{c}_{<t}^j) + q^j\} + \frac{\eta_t}{p^i})^+} d\mu(q^{\neq i}) \\ &\leq e^{\frac{\eta_t}{p^i}} \int e^{-(\max_{j \neq i}\{\eta_t(\hat{c}_{1:t}^i - \hat{c}_{1:t}^j) + q^j\})^+} d\mu(q^{\neq i}) = e^{\frac{\eta_t}{p^i}} \pi^i. \end{aligned}$$

Hence, $\pi^i \geq p^i e^{-\frac{\eta_t}{p^i}} \geq p^i \left(1 - \frac{\eta_t}{p^i}\right) = p^i - \eta_t$, which implies

$$\begin{aligned}\mathbf{E}_{q_*,q_t,r_t,u_t}[\widetilde{\hat{c}_t^{\mathrm{FPL}}}|h_{<t}] &= \sum_{i=1}^n p^i \sum_{j=1}^n p^j \mathbb{1}_{i=j} \frac{c_t^i}{p^i} = \sum_{i=1}^n p^i c_t^i \\ &\leq \sum_{i=1}^n \pi^i c_t^i + n\eta_t = \mathbf{E}_{q_*,q_t,r_t,u_t}[\widetilde{\hat{c}_t^{\mathrm{IFPL}}}|h_{<t}] + n\eta_t.\end{aligned}$$

This shows the step from feasible to infeasible FPL. The last step from infeasible FPL to the best decision in hindsight proceeds as shown already above and in [5,6]. Like before, it causes the upper bound of the cumulative regret to increase by $\frac{\log n}{\eta_T}$. This is true for any $(q_{1:T}, r_{1:T}, u_{1:T})$, hence also in expectation. The total regret is thus upper bounded by $\frac{\log n}{\eta_T} + n\sum_{t=1}^T \eta_t$, and we have just proved:

Theorem 2. *The algorithm FPL (1), obtaining cost estimates according to (12) and with learning rate* $\eta_t = \sqrt{\frac{\log n}{2nt}}$ *achieves a regret of at most*

$$\mathbf{E}c_{1:T}^{\mathrm{FPL}} - c_{1:T}^i \leq 2\sqrt{2Tn\log n} \quad \textit{for any } i \in \{1 \dots n\}. \tag{14}$$

We would like to point to a remarkable symmetry break here. It is straightforward to formulate FPL and the analysis from Section 3 for *reward maximization* instead of cost minimization. Then the (perturbed) leader is the expert with the highest (perturbed) reward, and perturbations are added to the scores. In the full information game, this reward maximization is perfectly symmetric to cost minimization by just setting $\textit{reward}_t^i = 1 - c_t^i$: all probabilities, distributions, and outcomes will be exactly the same. This is different in the partial observation case: There, in case of reward, the expert by FPL is the only one which can gain score. This is an advantage, in contrast to the disadvantage in case of loss minimization: Here, the selected expert is the only one to worsen its score. Put it differently, there is an automatic exploration or self-stabilization in the cost minimization case. With this intuition, it is less surprising that we did not need explicit exploration in Theorem 2. The corresponding result for reward maximization would not hold, as simple counterexamples show. Formally, it is the step from FPL to infeasible FPL which fails: A computation similar to (13) only shows $\pi^i \leq p^i e^{\frac{\eta_t}{p^i}}$, which does not imply a sufficiently strong assertion in general. However, reintroducing the exploration rate γ_t, we may set $\eta_t = \frac{\gamma_t}{n}$. This implies $\frac{\eta_t}{p^i} \leq 1$ for all i, hence $e^{\frac{\eta_t}{p^i}} \leq 1 + 2\frac{\eta_t}{p^i}$. Letting $\gamma_t = \sqrt{\frac{n\log n}{t}}$, we can conclude a bound like (14).

4.1 A Computationally Feasible Algorithm

We conclude this section by discussing a computationally feasible variant of FPL using all observations. This algorithm is constructed in a straightforward way: Select the current action $i = I_t^{\mathrm{FPL}}$ according to FPL and substitute the estimate $\hat{c}_t^i$ from (12) by $\hat{c}_t^i = \frac{c_t^i}{\hat{p}_t^i}$. It remains to estimate $\hat{p}_t^i$ by a Monte Carlo simulation.

There are two possibilities of error: either $\hat{p}_t^i$ overestimates p_t^i, or it underestimates p_t^i. The respective consequences are different: If $\hat{p}_t^i > p_t^i$, then the instantaneous cost of the selected expert is just underestimated. We can account for this by adding a small correction to the instantaneous regret. At the end of the game, we perform well with respect to the underestimated costs, which are upper bounded by the true costs. This does not cause any further problems. The case $\hat{p}_t^i < p_t^i$ is more critical, since then at the end of the game we perform well only w.r.t. overestimated costs. We therefore have to treat this case more carefully.

Problems arise if the true probability p_t^i is very close to 0, as then the Monte Carlo sample might contain very few or no hits and the variance of the estimated cost is high. Since FPL does not prevent this case, we reintroduce γ_t as an "exploration threshold". Let $\gamma_t = \frac{1}{2\sqrt{t}} \leq \frac{1}{2}$. We first assume that $p_t^i \geq \gamma_t$. If this assumption is false but we use $\hat{p}_t^i \geq \gamma_t$, then $\hat{p}_t^i$ is an overestimate and we have to consider an additional instantaneous regret. This case has probability at most γ_t. Consequently, as (true) instantaneous costs are always bounded by 1, the additional instantaneous regret is at most γ_t.

We sample the perturbed leader $k \in \mathbb{N}$ times and denote by $a^i(k)$ the number of times the leader happens to be expert i. Recall that expert i is the one already selected by FPL. By Hoeffding's inequality, the distribution of $\frac{a^i(k)}{k}$ is sharply peaked around its mean p^i:

$$\mathbf{P}\Big[\tfrac{a^i(k)}{k} - p^i \geq \tfrac{\gamma_t^2}{\sqrt{2}}\Big] \leq e^{-\gamma_t^4 k} \text{ and } \mathbf{P}\Big[\tfrac{a^i(k)}{k} - p^i \leq -\tfrac{\gamma_t^2}{\sqrt{2}}\Big] \leq e^{-\gamma_t^4 k}.$$

We choose k such that the probability bounds on the r.h.s. are at most γ_t, i.e. $e^{-\gamma_t^4 k} \leq \gamma_t$. Consequently we should sample $k = \lceil \gamma_t^{-4} \log(\gamma_t^{-1}) \rceil = \lceil 2t^2 \log(2\sqrt{t}) \rceil$ times. Hence the sampling complexity of the algorithm is $O(t^2 \log t)$. Let

$$\hat{p}_t^i := \max\big\{\gamma_t, \tfrac{a^i(k)}{k} - \tfrac{\gamma_t^2}{\sqrt{2}}\big\},$$

then $\hat{p}_t^i \leq p_t^i$ with probability at least $1 - \gamma_t$ (recall the assumption $p_t^i \geq \gamma_t$). Hence the possibility of overestimate $\hat{p}_t^i > p_t^i$ causes an additional regret of γ_t.

Finally we need to deal with possible underestimates. For some integer $m \geq 1$, the probability that $\hat{p}_t^i$ falls below $p_t^i - \frac{(\sqrt{m}+1)\gamma_t^2}{\sqrt{2}}$ is at most

$$\mathbf{P}\Big[\tfrac{a^i(k)}{k} - p^i \leq -\tfrac{\sqrt{m}\gamma_t^2}{\sqrt{2}}\Big] \leq e^{-m\gamma_t^4 k} \leq \gamma_t^m \tag{15}$$

by Hoeffding's inequality. We partition the interval $[\gamma_t, p_i^t)$ of all possible underestimates into subintervals $A_1 = \big[p_t^i - \frac{2\gamma_t^2}{\sqrt{2}}, p_t^i\big)$ and

$$A_m = \Big[p_t^i - \tfrac{(\sqrt{m}+1)\gamma_t^2}{\sqrt{2}}, p_t^i - \tfrac{(\sqrt{m-1}+1)\gamma_t^2}{\sqrt{2}}\Big), \quad m \geq 2.$$

We do not need to consider m with the property $A_m \cap [\gamma_t, p_i^t) = \emptyset$. That is, we can restrict to m small enough that $p_i^t - \sqrt{\frac{1}{2}}(\sqrt{m}+1)\gamma_t^2 \geq \gamma_t - \sqrt{\frac{1}{2}}\gamma_t^2$. Let M

be the largest m for which this condition is satisfied, then one can easily see $\sqrt{m}+1 \le \sqrt{M}+1 \le \sqrt{2}(p-\gamma_t+\sqrt{\frac{1}{2}\gamma_t^2})/\gamma_t^2$.

Claim. If $m \le M$, then $\frac{c_t^i}{p_t^i-(\sqrt{m}+1)\gamma_t^2/\sqrt{2}} \le \frac{c_t^i}{p_t^i}+\gamma_t(\sqrt{m}+1)$.

This follows by a simple algebraic manipulation. Consequently, for $\hat{p}_i^t \in A_m$, we have $\mathbf{E}\hat{c}_t^i \le c_t^i + (\sqrt{m}+1)\gamma_t$. Moreover, $\hat{p}_t^i \in A_m$ occurs with probability at most γ_t^{m-1} according to (15). By bounding the expectation over all A_m, we thus obtain an additional regret of at most

$$\sum_{m=1}^{M}(\sqrt{m}+1)\gamma_t^m \le \gamma_t \sum_{m=0}^{\infty}(m+2)\gamma_t^m \le \frac{2\gamma_t}{1-\gamma_t}+\frac{\gamma_t^2}{(1-\gamma_t)^2} \le 5\gamma_t,$$

since $\gamma_t \le \frac{1}{2}$. Altogether, this proves the following theorem.

Theorem 3. *Let $\gamma_t = \frac{1}{2\sqrt{t}}$ be the exploration threshold. In each time step, after selecting one expert i, let FPL obtain an estimate $\hat{p}_t^i = \max\{\gamma_t, \frac{a^i(k)}{k} - \frac{\gamma_t^2}{\sqrt{2}}\}$ for $\mathbf{P}(I_t^{\text{FPL}}=i)$, by sampling the perturbed leader $k = \lceil 2t^2 \log(2\sqrt{t})\rceil$ times and counting the number of hits $a^i(k)$. Let the estimated cost of the selected expert be $\hat{c}_t^i = c_t^i/\hat{p}_t^i$, and the estimated cost of all other experts be zero. Then the algorithm FPL (1) with learning rate $\eta_t = \sqrt{\frac{\log n}{2nt}}$ achieves a regret of at most*

$$\mathbf{E}c_{1:T}^{\text{FPL}} - c_{1:T}^i \le 2\sqrt{2Tn\log n} + 7\sqrt{T} \quad \textit{for any } i \in \{1 \ldots n\}. \tag{16}$$

5 Discussion

The main statement of this paper is the following:

> *If we have a regret minimization algorithm with a bound guaranteed against an oblivious adversary, and if the algorithm chooses the current action/expert by some independent random sampling based on past cumulative scores (e.g. FPL or weighted majority), then the same bound also holds against an adaptive adversary. This is true both for full and partial observations.*

We have used this argument for showing bounds for FPL in the adversarial bandit problem. The strategy to use only feedback from exploration rounds which is common for FPL achieves a regret bound of $O(t^{\frac{2}{3}})$. As the algorithm is label efficient, this bound is sharp. Using all observations allows to push the regret down to $O(\sqrt{t})$. Then however the sampling probabilities have to be approximated.

In the same way, it is possible to use our argument for the general geometric online optimization problem [7,8], also resulting in a $O(t^{\frac{2}{3}})$ regret bound against adaptive adversary. An interesting open problem is the following: Under which conditions and how is it possible to use all observations in the geometric online optimization problem, hopefully arriving at a $O(\sqrt{t})$ bound?

We conclude with a note on *regret against an adaptive adversary.* We considered the *external* regret w.r.t. the best action/strategy/expert from a pool. There are two directions from here. One is to go to different regret definitions, such as internal regret. The other one is to change the reference and compare to the *hypothetical* performance of the best strategy, in this way accepting a stronger type of dependency of the future costs from the currently selected action (see e.g. [11] and the references therein). It is one of the major open problems to propose refined algorithms and prove better bounds in this model.

References

1. Freund, Y., Schapire, R.E.: A decision-theoretic generalization of on-line learning and an application to boosting. Journal of Computer and System Sciences **55** (1997) 119–139
2. Auer, P., Cesa-Bianchi, N., Freund, Y., Schapire, R.E.: Gambling in a rigged casino: The adversarial multi-armed bandit problem. In: Proc. 36th Annual Symposium on Foundations of Computer Science (FOCS), IEEE (1995) 322–331
3. Auer, P., Cesa-Bianchi, N., Freund, Y., Schapire, R.E.: The nonstochastic multi-armed bandit problem. SIAM Journal on Computing **32** (2003) 48–77
4. Hannan, J.: Approximation to Bayes risk in repeated plays. In Dresher, M., Tucker, A.W., Wolfe, P., eds.: Contributions to the Theory of Games 3. Princeton University Press (1957) 97–139
5. Kalai, A., Vempala, S.: Efficient algorithms for online decision. In: Proc. 16th Annual Conference on Learning Theory (COLT). Springer (2003) 506–521
6. Hutter, M., Poland, J.: Adaptive online prediction by following the perturbed leader. Journal of Machine Learning Research **6** (2005) 639–660
7. McMahan, H.B., Blum, A.: Online geometric optimization in the bandit setting against an adaptive adversary. In: 17th Annual Conference on Learning Theory (COLT), Springer (2004) 109–123
8. Awerbuch, B., Kleinberg, R.D.: Adaptive routing with end-to-end feedback: distributed learning and geometric approaches. In: STOC '04: Proceedings of the thirty-sixth annual ACM symposium on Theory of computing. (2004) 45–53
9. Cesa-Bianchi, N., Lugosi, G., Stoltz, G.: Minimizing regret with label efficient prediction. In: 17th Annual Conference on Learning Theory (COLT). Springer (2004) 77–92
10. Cesa-Bianchi, N., Lugosi, G., Stoltz, G.: Regret minimization under partial monitoring. Technical report (2004)
11. Poland, J., Hutter, M.: Defensive universal learning with experts. (2005) International Conference on Algorithmic Learning Theory (ALT), to appear.

On Improved Least Flexibility First Heuristics Superior for Packing and Stock Cutting Problems

Yu-Liang Wu and Chi-Kong Chan

The Chinese University of Hong Kong, Shatin, Hong Kong
{ylw, chanck}@cse.cuhk.edu.hk

Abstract. Two dimensional cutting and packing problems have applications in many manufacturing and job allocation problems. In particular, in VLSI floor planning problems and stock cutting problems, many simulated annealing and genetic algorithms based methods have been proposed in the last ten years. These researches have mainly been focused on finding efficient data structures for representing packing results so the search space and processing time of the underlying search engine can be minimized. In this paper, we tackle the problem from a different approach. Instead of using stochastic searches, we introduce an effective deterministic optimization algorithm for packing and cutting. By combining an improved Least Flexibility First principle and a greedy search based evaluation routine, we can obtain very encouraging results: In stock cutting problems, our algorithm achieved over 99% average packing density for a series of public rectangle packing data sets, which is significantly better than the 96% packing density obtained by meta-heuristics (simulated annealing) based results while using much less CPU time; whereas in rectangle packing applying the well-known MCNC and GSRC benchmarks, we achieved the best (over 96%) packing density among all known published results packed by other methods. Our encouraging results seem to suggesting a new experimental direction in designing efficient deterministic heuristics for some kind of hard combinatorial problems.

1 Introduction

The two dimensional packing problem is defined as follow: given a set of *n* rectangular items of size (w_1,h_1), (w_2,h_2), … , (w_n,h_n), and a larger container box, we want to find the minimum box size that would allow all the rectangles to be fit inside the box without overlapping. Packing problem [17-26] has applications in many areas ranging from newspaper layout editing to VLSI floor planning problems. Another problem closely related to packing is the stock cutting problem[1][6][7][10][11][13], which requires the rectangles to be packed in a long strip of fixed width W, and of undefined height. The objective is to minimize the height of the strip used for the packing. This problem occurs in many practical applications ranging from steel sheet pattern cutting to resource allocation and job scheduling problems.

Both packing and stock cutting are the 2-dimensional extension of the 1-D bin packing problem, which has been shown to be NP-complete. As a result, most researches in this subject are focusing on approximate solutions, commonly employing heuristic and meta-heuristics algorithms such as simulated annealing (SA) or genetic algorithms (GA).

In stock cutting, one of the earliest approach is the *Bottom-Left* (BL) heuristic. In BL, the rectangles to be packed are, starting from the top-right corner of the container box, first slide vertically downward as far as possible, followed by sliding horizontally as far as possible to the left. Several enhanced version of BL exists. One variant is the *Bottom-Left-Fill* (BLF) heuristic. In BLF, instead of sliding downward and leftward, the rectangles are placed directly into the lowest positions available, left justified. BLF can fill "holes" that are not reachable using BL, but has a higher complexity ($O(n^3)$ Vs $O(n \log n)$) [12].

Both BL and BLF are later used in meta-heuristics based approaches. Jacobs [6] proposed an approach that used GA to determine the packing order of the rectangles which are to be packed using BL, Dagli and Poshyanonder [11] proposed a hybrid GA and neural network approach based on slicing method. Burke and Kendall [10] used SA to solve a related but more general problem of convex polygon packing, and applied it in stock cutting in their experiments. Finally, Hopper and Turon [12] studied the result of different combination of SA and GA, with various placement heuristics such as BL or BLF, and concluded that the SA + BLF combination is the best in term of solution quality for stock cutting problems.

Researches in packing problems have been even more active than in stock cutting, particularly in the VLSI floor-planning communities where many SA based approaches have been proposed. These researches have mainly been focusing on finding an efficient data structure for representing packing results such that the search space and processing time of the underlying SA search engine can be minimized. Typically, a packing solution is encoded as some type of sequences, graphs or trees based representation. Some corresponding moves are then defined for transforming one packing solution to another and an SA engine is then responsibility for finding a good packing result. One such early work is the Normalized Polish Expression [21] for handling a special type of packing problems known as *slicing floorplans packing*.

Another influential SA-based packing approach is the sequence pair (SP) [9]. In [9], a floorplan is represented as two sequences, where each sequence contains a permutation of the rectangles to be packed. The relative locations or any two rectangles in the two sequences define the geometric relations between the two rectangles in the packing. The original SP algorithm required $O(n^2)$ time for decoding and evaluating any pair of sequences. Later, a faster version [19] reduced it to $O(n \log n)$ time.

Soon after the introduction of SP, many other competing representations began to appear. In the O-tree approach [24], each rectangle is represented by a node in a tree, where the x axis coordinate of any rectangles is determined by the location and width of its parent node rectangle only, and the y-axis coordinate is to be decided by the insertion orders of the nodes in the tree. The B*-tree [25] approach proposed to use a binary tree representation, with left child of a node representing an adjacent rectangle located on the top, and the right child represents the lowest unvisited adjacent rectangle located on the right. Moves are defined for rotating rectangles, moving nodes and swapping nodes within the tree for the benefit of the SA engine.

Another graph based method is TCG [20] which was later enhanced to a faster version called TCG-S [21]. In TCG (and TCG-S) a packing is represented as a pair of transitive closure graphs. Here, each graph directly encodes the relative positions of any two rectangles. TCG and TCG-S have been shown to equivalent to SP (each TCG pair of graphs corresponds to one sequence pair and vice versa). Finally, an approach called ACG [23] was published recently. Instead of encoding the relative location of each pair of rectangles as done in SP and TCG, ACG uses a constraint graph to represent only rectangles that are adjacent to each other. Their results showed that ACG is superior in term of packing area while using similar annealing schedule.

Apart from general non-slicing floorplan packing, there are also several works dedicated to a special type of floorplans packing problem known as *Mosaic* floorplans. Basically, a Mosaic floorplan is one that contains no empty rooms (*i.e.* no unoccupied empty space between two rectangles). Examples of these works include the corner list (ECBL) [17], the Q-sequence [18] and the twin binary sequence (TBS)[26]. These algorithms can be extended for handling general packing problems by inserting dummy rooms, but at various costs.

So far, these SA and GA based approaches are producing close to optimal results in a series of small to medium size benchmark problems. For larger problems with 100 rectangles or more, however, the results are not as convincing. Yet, experience tells us that for these larger problems it should actually be *easier* to produce good results in term of packing density. The reason is that there are more rectangles of various sizes available to fill up any packing gaps. (Small problems, on the other hands, are already solved by exhaustive searches, meaning that approximate solutions are not required, even if the SA results are good.[27]

Instead of using stochastic algorithms, our research took another direction. In our previous work we proposed a “Least Flexibility First” (LFF) packing principle [8] in which the least flexible rectangles, which are the longest ones, are placed into the least flexible locations of the box, the corners, using a search that involves evaluation by greedy algorithm. Our current work is developed on the foundation of the LFF principles. In this paper, we take the LFF principle one step further by considering the both the flexibility of candidate placement locations and that of the rectangles. By introducing a tightness measure for representing the degree of fitting between rectangles and placement locations, we can propose an enhanced version of LFF, called LFFT, which will be discussed in the following sections. We will see that the result of applying LFFT in both packing and cutting is very encouraging, in terms of both packing density and running time, compared with the above mentioned algorithms. We also propose a fast version of the algorithm known as LFFT-fast for handling large problem sets efficiently.

The organization of the remaining parts of this paper is as follow. Section 2 describes the Least-Flexibility-First principle and the LFFT algorithm. Section 3 and section 4 handles the stock cutting problems and the packing problems respectively. Section 5 concludes our work.

2 The Least Flexibility First Principle

A 2-d packing problem can be stated as follow: Given an initially empty rectangular container box, with coordinate and a set of smaller rectangles (w_1,h_1), $(w_2,h_2)\ldots(w_n,h_n)$, where each w and h represent the width and height of a rectangle, we need to find a packing solution such that each rectangle can be placed inside the box without overlapping. If no such solutions are found, depending on the application requirement, we can either obtain a partial solution in which the amount of unused space is minimized or we can repeat with larger container boxes in order to find the smallest box size that will fit (as in stock cutting problems and packing problems).

LFF is based on the idea of flexibility. Two types of flexibilities are considered: (1) *Flexibility of rectangle*: The flexibility of a rectangle can be defined as $Flex_r = 1 / (\max (W_h, H_h))$ where W_h and H_h are width and height of the rectangle respectively. In LFF, the rectangles to be packed are pre-sorted such that the longer rectangles (i.e. the less flexible ones) will have higher packing priority. (2) *Flexibility of location*: the ancient Chinese farmers and masons had an approximating heuristic to the packing problems that can be summarized by the following rule-of thumb: "*Golden are the corners; silvery are the sides; and strawy are the voids*" The saying assign higher priorities to the rectangle placement locations that situate at corners of the box or corners formed by other rectangles, followed by locations along the sides, as the corners locations are considered to have less flexibilities than the side locations and void spaces. The idea is illustrated by Figure 1. The corner placement locations (C and D), are less flexible than the side location (B) and the void space (A). In this case, location (D) is the preferred placement location of the rectangle.

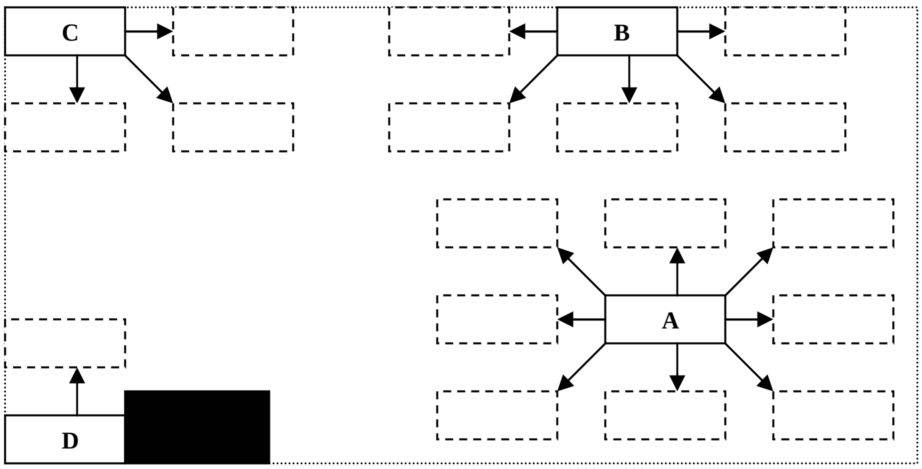


Fig. 1. The least flexibility first principle

The basic principle of LFF is thus to pack the least flexible remaining rectangles, which is the longest one, in one the least flexible packing locations, the corners. However, as illustrated in Figure 1, for a given rectangle, some corner locations (D) are more desirable than others (C). To capture this idea, we can define, given and a list of already packed rectangles *P*, the *degree of fitting* between a rectangle *R* and a candidate location *L* as *Fit(R,L,P) = SharedPerim(R,L,P) / Perim (R)* where *SharedPerim(R,L,P)* is the length of the perimeter of *R* that is shared by the edge of the box, or by other rectangles in *L*, and Perim(*R*) is the perimeter of *R*.

The idea of rectangle flexibility, location flexibility and the degree of fitting is utilized by the LFFT algorithm as given in Figure 3. The central part of the LFFT algorithm is a series of independent greedy search that evaluate the *fitness function value* (FFV) of a list of (*rect, corner*) pairs, where *rect* is a rectangle to be packed, *corner* is a corner location that the rectangle can be placed without overlapping. Each pair forms a *corner-occupying packing move* (COPM). In each iteration, each COPM is *pseudo-packed* in turn and is evaluated using a greedy search such that, in each step of the greedy-search, the least flexible remaining rectangle (the longest one), is pseudo-packed in the corner location with the highest *degree of fitting*. (The pseudo-packed rectangles are removed after each greedy search), until no more rectangles can be packed. The total packed area at the end of the greedy search is then used as the fitness function value (FFV) for that COPM. The COPM with the highest FFV is chosen as the next packing move.

In our implementation, the degree of fitting for each COPM is approximated using an eight point *tightness* heuristic which is defined as follow. For each candidate rectangle placement location, we look at the points that are immediately adjacent to each corner of the candidate rectangle. That is, suppose a COPM is bounded by the points $\{(x_0,y_0), (x_0,y_1), (x_1,y_1), (x_1,y_0)\}$, we check whether the 8 *corner adjacent points* $\{(x_0 , y_0 - \delta), (x_0 - \delta , y_0), (x_0 - \delta ,y_1), (x_0, y_1 + \delta) (x_1, y_1 + \delta), (x_1 + \delta ,y_1), (x_1 + \delta ,y_0), (x_1, y_0 - \delta)\}$ are occupied. In Figure 2, the locations of the eight points are marked by circle for both placement location *A* and *B*.

Our tightness value heuristic is then simply defined as the number of corner adjacent points that is not located in open space (that is, either in packed area or outside the box boundary). In the Figure 2 example, the tightness value of rectangle *A* and *B* are 3 and 4 respectively.

LFFT differs from the previous version (LFF) [9] in two aspects. The first is a way to choose the next rectangle to be greedily packed. In the original approach, we only need to find the longest rectangle that has a valid move. As we don't care where it is placed we can simply pick the first valid corner in the list. In LFFT, however, we need also to examine the tightness value of each valid corner moves for that chosen rectangle. That is, for each step in the greedy search, we find the longest rectangle that has a valid move, and pack it in the corner location that has the highest tightness value. The second difference is the introduction of a parameter *q*. In the original LFF algorithm, at the beginning and after any rectangle is packed, an updated list of COPMs is generated for each of the remaining unpacked rectangle. Each of the COPMs is then evaluated by a greedy search to determine the fitness function value (FFV) and the COPM with the highest FFV is picked as the next move. However, observations indicated that the COPM that has the highest FFV is much more likely to belong to one of the longest few rectangles. This means that most of the greedy searches performed for the COPMs of the shorter rectangles have little effects on the outcome. Because of this, a *top-level branch factor* parameter *q* is introduced in LFFT. In each iteration, we perform greedy searches for the COPMs of the *q* longest rectangles only, where $1 \leq q \leq n$, and *n* is the number of rectangles in the problem set.

As in the original LFF algorithm, the placement of a rectangle in each iteration of the LFFT algorithm will occupy one or more corners, but, in the same time, also generates some new corners. The upper bound of the number of corners should be proportional to *n*, where *n* is the total number of rectangles to be packed. Therefore the length of the COPMs list in each iteration will be bounded by $O(q * n)$. For each entry in the COPM list, a greedy search is performed so that each remaining rectangle will be pseudo-packed. In our implementation, both the packed and pseudo-packed rectangles are stored using a k-d tree data structure [2], which is basically a multi-dimensional binary tree that can be used for supporting area operations on 2-space. The k-d tree helps to reduce the complexity by providing a fast O(log n) region search operations. As a result, the complexity of one iteration of LFFT will be $O(q * n * n^2 * \log n)$. As the process is repeated once for each rectangle packed (i.e. O(n)), the worst case time complexity of LFFT is thus $O(n^4 \log n)$ if a constant value q is used, or $O(n^5 \log n)$ if $q = n$.

The value to be chosen for q would depend on the problem set size, the desired result quality and the speed of the machine available. Our experiments are run on a 1.8GHz Pentium 4 PC with 256MB memory with all programs coded in C. Under this setting, experience suggested that we can use q=n for very small problems with less than 50 rectangles; use q=5 or q=10 for medium sized problems with 150 rectangles or less, and use q=1 for larger problems with 150 rectangles or more.

Apart from the LFFT algorithm presented above, we also implemented a fast version of the algorithm, labeled LFFT-fast. Unlike the full version of LFFT, LFFT-fast do not perform multiple greedy searches for each COPM. Instead, in each iteration, the least flexible remaining rectangle (the longest one) is packed directly at the location with the highest tightness value. The algorithm is shown in Figure 4.

Experiment results for both LFFT and LFFT-fast are presented in the following sections.

3 Stock Cutting Problems Experiments

In this section we apply LFFT to the stock cutting problem, which is described as follows: Given a set of m rectangles $\{r_1, r_2, r_3 \ldots r_m\}$, where each r_i has a width w_i and height h_i, and a container box of width W and undefined height, we want to pack the rectangles in into the bottom of the box such that the height of the used section of the box is minimized. The rectangles can be rotated by 90 degrees if necessary.

We apply our algorithm to stock cutting by a simple approach that we repeatedly apply LFFT using a small top level branch factor q, and systematically increase the height of the container box by 0.1% in subsequence calls until a successful complete packing is achieved. Experiment results show that LFFT achieves a high packing density of over 97% for most of the larger (size > 50) benchmark datasets we tested, so the number of iterations is manageable. The value to be chosen for the top-level branch factor parameter (q) would depend on the data set size, the desired result quality and the speed of the machine available as mentioned above. Experiment results are given in the following sub-sections where all running time are rounded to the nearest seconds. Running times of less than 1 second are displayed as ‘1’. The running time of LFFT includes the running time of the final successful run where all rectangles are packed and the running times of all previous unsuccessful attempts. The packing densities, which are defined as the total packed area divided by the container box area, are displayed in percent.

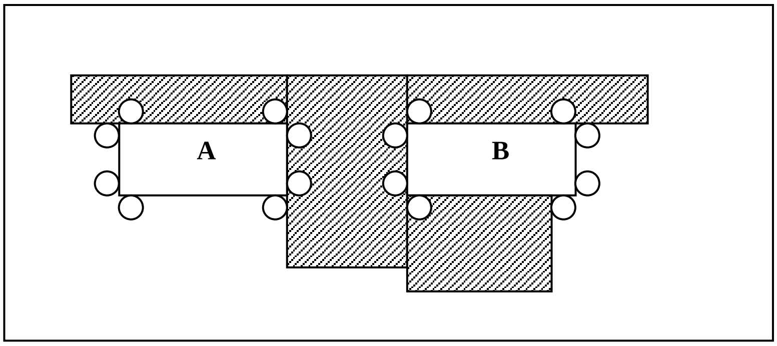


Fig. 2. Tightness measure

```
1 Choose the longest $q$ rectangles from the list of rectangles that are not yet packed
  1.1 Generate a list of (rect, corner) pairs, where rect is in the list of the $q$ longest unpacked
      rectangles, corner is a corner location that the rectangle can be placed without overlap-
      ping. Each pair forms a corner-occupying packing move (COPM)
  1.2 For each COPM
    1.2.1 Pseudo pack this COPM . (i.e., temporarily pack this COPM for evaluation)
    1.2.2 Sort all the remaining unpacked rectangles by their longest side and pseudo pack all
          these remaining rectangles greedily : -
      1.2.2.1 While there are rectangles remaining and there are spaces in the box,
        1.2.2.1.1 If there is a location that the next rectangle in the sorted list can fit, then,
                    - For each legal corner moves of that rectangle, calculate the tightness values.
                    - Choose the corner move with the highest tightness values and pseudo pack this
                      rectangle there.
                    - Update the corner lists and move to the next rectangle in the list
                  Else
                    - Skip this rectangle
      1.2.2.2 . At the end of the greedy search, return the total packed area as the fitness function
                value (FFV) of this COPM.
1.  1.3 Choose COPM with the highest FFV as the next move, update the list of packed rectangles
        and corners, and go to step 1 to repeat the process if there are rectangles still not packed
.
  Note: All pseudo packed rectangles are removed before the next COPM is considered.
```

Fig. 3. LFFT algorithm

3.1 Hopper and Turton Stock Cutting Data Sets

The first set of tests is done using the Hopper and Turton data sets defined in [13]. There are 21 data sets with size ranging from 16 to 196 rectangles grouped in 7

```
1 Choose the longest rectangle from the list of rectangles that are not yet packed
    1.1 Generate a list of (rect, corner) pairs, where rect is the longest unpacked rectangles,
        corner is a corner location that the rectangle can be placed without overlapping. Each pair
        forms a corner-occupying packing move (COPM)
    1.2  If the list of COPM is not empty
        1.2.1  For each COPM
            1.2.1.1 Calculate the tightness value
        1.2.2  Choose the COPM with the highest tightness value as the next move, update the lis
  packed rectangles and corners, and go to step 1 to repeat the process if there are rectangles
still not packed.
    1.3  Else
        1.3.1  Report failure.
```

Fig. 4. LFFT-fast algorithm

different sized categories, with the optimal packing density being 100% for all problems. The result for the data sets using LFFT with q=5 is shown in Figure 5. The corresponding BLF, BLF + GA and BLF + SA results as quoted from [13] are also listed for comparison. It should be noted that their tests are run on a machine where the speed is only about 10% of ours. But as shown below, our results are still good in terms of running time even after we take the machine speed into account. In fact, LFFT achieved the optimal solution of 100% packing density in 12 cases out of the total 21 cases, over 99% packing density in 18 cases (The results would be even better if a higher value of *q* is used, but the running time is also proportional to *q*). The running time for the largest 197 rectangles problem set (Category 7 problem 3) is 2234 seconds for our PC machine, which is acceptable.

	Size	*Width*	*BLF*		*BLF + GA*		*BLF + SA*		*LFFT (q=5)*	
			Density	*Time (s)*	*Density*	*Time (s)*	*Density*	*Time (s)*	*Density*	*Time (s)*
Cat 1	16/17	20	89%	1	96%	60	96%	42	100%	1
Cat 2	25	40	84%	1	93%	120	94%	144	100%	1
Cat 3	28/29	60	88%	1	95%	180	95%	240	99%	2
Cat 4	49	60	95%	1	97%	780	97%	1980	98%	15
Cat 5	73	60	95%	1	96%	2180	97%	6900	99%	31
Cat 6	97	80	95%	1	96%	5160	97%	22920	99%	92
Cat 7	196/197	160	95%	1	95%	46620	96%	258600	99%	2150
Average			*92%*		*95%*		*96%*		*99%*	

Fig. 5. Stocking cutting results for Hopper Dataset

3.2 Other Stock Cutting Data Sets from the Literature

Next, we tested our algorithm using problem sets from other literatures. Unfortunately the running time is not published in manyof the cases so most comparisons are on packing density only. Because of the small data set size we used LFFT with *q*=*n* for all cases. As shown in Figure 6, our results are also good comparing with other known approaches. For the Kendall dataset, our LFFT result of height 148 is not too far from the optimal result of 140, and is better than the SA [10] result of 158. For the Jacobs data sets J1 and J2, our result of height 16 is better than the mean height of 17 obtained using GA [6][16]. For the Dagli data sets D1 and D2, our packing densities are 97.8% and 97.6% respectively, which achieved more than 5% improvements over the GA result of 92%[14]. Hopper and Turton reported results in the range of 92% to 98% using BLF + SA [13] for these two problems, so in comparison our results are good. For the Dagli dataset D3, our result of 98.6% is better than the GA best-case result of 94% reported in [15]. For the Dagli dataset D4 our result is better than both the reported result of a hybrid GA + neural network approach [11] and the result using BLF + SA[13].

Data set	Size	Box Width	LFFT (q=n)			Other Algorithms	
			Height	Packing Density	Time (s)	Algorithm	Packing Density
Kendall [10]	12	80	148	94.6%	3	SA [10]	88.6%
Jacobs J1 [6]	25	40	16	93.7%	13	GA [6][16] BL [6]	88.2% 71.4%
Jacobs J2 [6]	50	40	16	93.7%	88	GA [6][16] BL [6]	88.2% 71.4%
Dagli D1 [14]	31	31	46	97.8%	17	GA [14] BLF + SA [13]	92% 92 % to 98%
Dagli D2 [14]	21	21	41	97.6%	4	GA[14] BLF + SA [13]	92% 92 % to 98%
Dagli D3 [15]	37	37	113	98.6%	32	GA[15] #	94%
Dagli D4 [11]	37	37	164	97.6%	40	Neural GA [11] BLF + SA[13]	95% to 97% 95% to 96%
# Best case result							

Fig. 6. Stocking cutting results (other problem sets)

4 Packing Problems Experiments Using LFFT

In this section we discuss the result of applying LFFT in packing problems, which can be described as follow: given a set of *m* rectangles $\{r_1, r_2, r_3,\ldots r_m\}$, where each rectangle r_i has a width w_i and height h_i and a container box of to-be-determined size, we want to pack the rectangles in into the box such that the area of the container box is minimized. The rectangles can be rotated by 90 degrees if necessary. As with stock cutting problems, we handle this by repeatedly applying LFFT using a small top-level branch factor (*q*), and systematically increase the width height of the container box in subsequence calls until a successful complete packing is achieved. As before, the choice of *q* depends on the problem size and the desired quality of the packing result.

Results of applying LFFT on MCNC and GSRC benchmark problems are shown in Figure 7. We tested the two medium size MCNC problems *ami33* and *ami49* using LFFT with *q*=1, whereas the larger GSRC data sets *n100*, *n200* and *n300* are handled using LFFT-fast. We did not perform test on the smaller MCNC benchmarks *apte, xerox* and *hp* as these problems have already been solved by exhaustive search, making approximate algorithms meaningless [27]. Results quoted from previous SA based approaches are also listed for comparison, with the exception of the TBS results [26] where we, for the sake of fair comparison, choose to repeat their results with a longer running time than they previously allowed[1]. LFFT achieved packing density of 96.8% for *ami33*, 97.8% for *ami49*, 99.8% for *n100*, 95.9% for *n200*, 96.5% for *n300*, and, as seen in figure 7, LFFT is better than the SA based approaches in term of smaller used

	ami33		ami49		n100		n200		n300	
	Area	Time	Area	Time	Area	Time	Area	Time	Area	Time
TCG[2] [20]	1.2	91	37.04	590	197800	2487				
TCG-S[2] [22]	1.24	23	38.47	63	192000	458	197000	4233	307000	163000
ACG[2] [23]	1.2	2	36.92	7	187500	28	187100	129	293700	322
ENPA[3] [24]	1.242	119	37.73	406	-		-		-	
ECBL[4] [17]	1.192	73	36.70	117	-		-		-	
TBS[2] [26]	1.186	50	37.01	985	-		-		-	
B* tree[4] [25]	1.27	3417	36.8	4752	-		-		-	
Fast-SP[4][19]	1.185	28	36.82	48	191200	128	197500	350	312300	653
Q-Seq[4] [18]	1.194	40	36.75	57	-		-		-	
LFFT	1.177	10	36.24	77	179776	1	183184	6	283024	17

Fig. 7. Packing results for MCNC and GSRC problem sets

area, while uses less computation time. For instance, LFFT-fast result for the largest problem (the 300 rectangles *n300*) is a big improvement over the currently best published result to our knowledge using ACG (packing density of 96.5% Vs 93.0%) while using much less CPU time (17 sec Vs 322 sec).

Additionally, we also tested the *playout* data set and achieved an encouraging packing density of 98.9% in 148 seconds using LFFT with q=10.

5 Conclusion

Packing and stock cutting problems are active research areas where many stochastic or non-deterministic algorithms (like SA and GA) approaches were proposed in the past. While being able to obtain good results for many small to medium size benchmark problems, the CPU overhead has imposed a severe obstacle for their applications upon larger problem instances (for example, larger problems with 100 or more rectangles). In this paper, we handle the packing and cutting problem by extending our greedy-search based Least-Flexibility-First packing algorithm using an eight-

[1] We would like to thank E.F.Y.Young, author of [26], for kindly providing us their source-code for this purpose.

[2] Best result from 5 runs.

[3] Best result from 100 runs.

[4] It is not clear that whether they are publishing the best result or the average results from several runs.

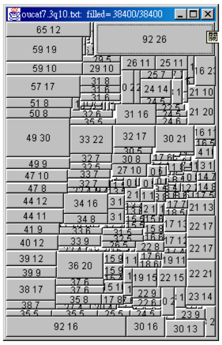
points tightness measure. Our result is very encouraging. The improvement in packing density allowed us to apply the enhanced algorithm for both stock-cutting and packing problems by repeatedly applying the algorithm using a small top-level branch factor. This approach achieved a high average packing density of over 99% in one set of stock cutting benchmark data sets with the optimal solution found in over half of the cases, and we also obtained close to optimal results in other datasets from the literature. The result of applying the algorithm to packing problems is also excellent since our results is better than all other known results produced by SA based algorithms while we used much less CPU time.

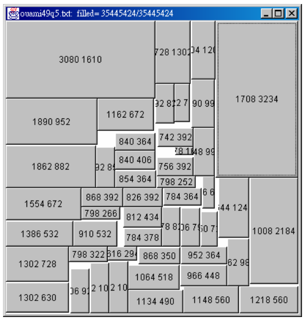
References

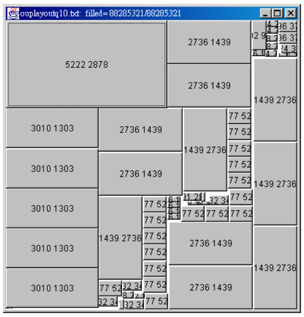
1. B.S. Baker, E.G. Coffman, Jr. and R.L. Rivest, Orthogonal Packing in two dimensions. SIAM Journal on Computing 9 (1980), pg 846-855
2. J.L. Bentley, Multidimensional binary search trees used for associative searching, Communication of ACM 18 (9) (1975) pg 507-517
3. C. Kenyon and E. Remila, Approximate strip packing, Proc. 37th IEEE Symposium on Foundations of Computer Science (1996) pg 31-36
4. Kenyon and Remila, A near optimal solution to a two-dimensional cutting stock problem, Mathematics of Operations Research Vol 25, Issue 4 (2000)
5. K. Dowsland, Some experiments with simulated annealing techniques for packing problems. European Journal of Operational Research 68 (1993), pg 389-399
6. S. Jacobs, On Genetic algorithms for the packing of polygons. European Journal of Operational Research 88 (1996), pg 165-181
7. T.W. Leung, C.K. Chan, M.D. Troutt, Mixed simulated annealing-Genetic algorithm Application of a mixed simulated annealing-genetic algorithm heuristic for the two-dimensional orthogonal packing problem, European Journal of Operational Research 145 (2003), pg 530-542
8. Y.L Wu, W.Q. Huang, S.C. Lau, C.K. Wong and G.H. Young, An effective quasi-human heuristic for solving the rectangle packing problem, European Journal of Operational Research 141 (2002), pg 341-358
9. H. Murata, K. Fujiyoshi, M. Kaneko, VLSI/PCB placement with obstacles based on sequence pair, IEEE Transactions on Computer-aided Design of Integrated Circuits and Systems 17 (1) (1998) 60-67
10. Burke E. and Kendall G., Applying Simulated Annealing and the No Fit Polygon to the Nesting Problem. Proceedings of the World Manufacturing Congress, Durham, UK (1999), pg 27-30
11. Dagli, CH, Poshyanonder, new approaches to nesting rectangular patterns, New Ap proaches to Nesting Rectangular Patterns, Journal of Intelligent Manufacturing, Vol 8, 3 (1997), pp 177-190.
12. Chazzelle B., The Bottom-Left Bin Packing Heuristic: An efficient Implementation. IEEE Transactions on Computers c32/8. (1983) pg 697-707
13. Hopper, E. and Turton, B., C.,H , An empirical investigation of meta-heuristic and heuristic algorithms for a 2D packing problem", European Journal of Operational Research 128/1 (2000), pg 34-57
14. Ratanapan K. and Dagli C. H., An object-based evolutionary algorithm for solving irregular nesting problems, Proceedings for Artificial Neural Networks in Engineering Conference, vol. 7, ASME Press, New York (1997), pp. 383-388.

15. Ratanapan K. and Dagli CH, An object-based evolutionary algorithm: the nesting solution, IEEE (Eds.) Proceedings of the International Conference on Evolutionary Computation 1998, ICEC '98, IEEE, Piscataway, NJ, USA, pp. 581-586
16. Liu, D., H. Teng. An Improved BL-algorithm for Genetic Algorithm of the Orthogonal. Packing of Rectangles. European Journal of Operational Research. 112 (1999) pg 412-420.
17. S. Zhou, S.Dong, X.Hong, Y.Cai, CK Cheng, and J. Gu, ECBL, An Extended Corner BlockList with solution Space including Optimum Placement, Proceeding of International Symposium on Physical Design, (2001) pp. 156-161
18. C Zhuang, K. Saknushi, L.Jin and Y.Kajitani, An Enhanced Q-Sequence Augmented with Empty-Room-Insertion and Parenthesis Trees, Proceedings of Design, Automation and Test in Europe (2002), pp 61-68
19. Xiaoping Tang and D.F. Wong, FAST-SP: A Fast Algorithm for Block Placement based on Sequence Pair, Proceeding of IEEE Asia South Pacific Design Automation Conference(2001), pp 521-526
20. J-M Lin and Y-W Chang, TCG: A Transitive Closure Graph-Based Representation for Non-Slicing Floorplans. Proceedings of the 38th ACM/IEEE Design Automation Conference (2001), pp 764-769
21. D.F.Wong and C.L.Liu, Anew algorithm for floorplanning design, DAC 1986, pp 101-107
22. J-M Lin and Y-W Chang, TCG-S: orthogonal coupling of P*-admissable representation for general floorplans, DAC 20002 pp 764-769
23. H.Zhou and J.Wang, ACG- adjacent constraint graph for general floorplans, ICCD 2004 pp 572-575
24. P.Wang, C.K.Cheng and T.Yoshimura, An enhanced perturbing algorithm for floorplan design using the O-tree representation, ISPD 2000, pp 168-173
25. Y.C.Chang and Y.W.Chang, E.M.Wu and S.W.Wu, B*-trees: a new representation for non-slicing floorplans, DAC 2000, pp 458-463
26. E.F.Y.Young , C.C.N.Chu and Z.C.Sgen, Twin binary sequences : a non-redundant , representation for general non-slciing floorplan, IEEE Transaction on CAD 22(4), pp457-469, 2003
27. H.H.Chan and I.L.Markov, Practical slicing and non-slicing block-packing without simulated annealing, IEEE Great Lake Symp. On VLSI 2004, pp 282-287

Appendix: Sample Results







Hopper and Turton category 7 problem number 3 (left), ami49 (middle) and playout (right)

Evolutionary Testing Techniques

Joachim Wegener

DaimlerChrysler AG, Research and Technology,
Alt-Moabit 96 a, D-10559 Berlin, Germany
Joachim.Wegener@daimlerchrysler.com

Abstract. The development and testing of software-based systems is an essential activity for the automotive industry. 50-70 software-based systems with different complexities and developed by various suppliers are installed in today's premium vehicles, communicating with each other via different bus systems. The integration and testing of systems of this complexity is a very challenging task. The aim of testing is to detect faults in the systems under test and to convey confidence in the correct functioning of the systems if no faults are found during comprehensive testing. Faults not found in the different testing phases could have significant consequences that range from customer dissatisfaction to damage of physical property or, in safety relevant areas, even to the endangering of human lives. Therefore, the thorough testing of developed systems is essential. Evolutionary Testing tries to improve the effectiveness and efficiency of the testing process by transforming testing objectives into search problems, and applying evolutionary computation in order to solve them.

1 Introduction

In today's premium vehicles 50-70 embedded systems with different micro-controllers and different operating systems (OSEK, QNX…) communicate via several bus systems (CAN-B, CAN-C, MOST, Flexray…) exchanging thousands of signals and messages. In order to realize complex functionalities, several embedded systems have to interact closely, e.g. an ESP system stabilizing a car in critical driving situations has to control at least the brakes, gear, and engine. The software-based systems integrated in a vehicle are developed and produced by a large number of different suppliers. A single software-based system can contain up to 4,000,000 lines of software code. The size of the code in the entire vehicle can easily reach more than 10,000,000 lines of software code resulting in up to 500 MB of on-board software. Unforeseen feature interaction can occur due to the interplay of electronics and software technology within a network made up of systems of such complexity.

The testing of systems of this degree of complexity is a very challenging task which is considerably more complex than the testing of conventional software systems. This is due to the technical features of automotive systems, e.g. the computational accuracy of the target system, memory space requirements which have to be guaranteed during program execution, or the synchronization of parallel processes running on different embedded systems. It is also due to special requirements made on these kinds of systems, e.g. the fulfillment of functional as well as non-functional requirements such as real-time and safety requirements. Therefore, testing of software-based systems takes up to 50% of the overall development effort and budget for automotive embedded systems.

The most significant weakness of testing is that the postulated functioning of the tested system can, in principle, only be verified for those input situations which were selected as test data. According to Dijkstra [1], testing can only show the existence but not the non-existence of errors. Proof of correctness can only be produced by a complete test, i.e. a test with all possible input values, input value sequences, and input value combinations under all practically possible constraints. In practice, complete testing is usually impossible because of the vast amount of possible input situations. Therefore, testing is only a sampling method. Consequently, one of the major challenges associated with testing is that of finding test cases that are effective at finding faults without requiring an excessive number of tests to be carried out. If test cases relevant to the practical deployment of the systems are omitted, the probability of detecting faults declines – faults may occur, leading to customer dissatisfaction or to damage to physical property.

Since manual test case design is a time-consuming, tedious, difficult, and error-prone activity, DaimlerChrysler, Research and Technology develops testing methods to increase the effectiveness and efficiency of the test and thus to reduce the overall development costs for software-based systems. An extensive automation of testing can be achieved by transforming testing objectives into search problems which are then solved using evolutionary computation. The application of evolutionary computation to test automation is called Evolutionary Testing [2]. This paper surveys some of the work undertaken by DaimlerChrysler on the application of Evolutionary Testing to solve different testing problems occurring during the development of automotive systems.

The second chapter introduces the basic principles of applying evolutionary algorithms to testing. The third chapter discusses the use of Evolutionary Testing for functional testing. The fourth chapter describes its application for the automation of structural testing, and the fifth chapter presents results for real-time testing. Finally, chapter 6 concludes the paper with a summary and an outlook to future work.

2 Evolutionary Testing

Due to the non-linearity of software (if-statements, loops, etc.), the conversion of testing objectives into optimization tasks usually results in complex, discontinuous, and non-linear search spaces. Neighbourhood search methods such as hill climbing are not suitable in such cases. Therefore, meta-heuristic search methods, such as evolutionary algorithms, are employed because their robustness and suitability for the solution of different test tasks has already been proven in previous work, e.g. [3], [4], and [5].

The suitability of evolutionary algorithms for testing is based on their ability to produce effective solutions for complex and poorly understood search spaces with many dimensions. The dimensions of the search spaces are directly related to the number of input dimensions of the system under test. The execution of different program paths and the nested structures in software systems lead to multi-modal search spaces when testing. Apart from many local optima, these are also marked by jumps and levels of identical objective function values. Input parameter dependencies within the system under test may result in definition gaps. A noisy objective function may also be caused by internal system states. In this case, identical input values may result in different objective function values.

During optimization, evolutionary algorithms identify the building blocks of an ideal solution, and store those blocks in the individuals within the population. Building blocks are the partial solutions (genetic modules) of which good solutions are composed. The combination of individuals and different building blocks results in more qualified individuals as optimization continues. As well as being particularly well-suited to the treatment of complex search spaces, evolutionary algorithms also represent a very robust optimization procedure.

In order to automate tests using evolutionary algorithms, the test aim must be transformed into an optimization task. A numerical representation of the test aim is necessary, from which a suitable objective function for the evaluation of the generated test scenarios can be derived. Depending on which test aim is pursued, different objective functions emerge for the evaluation of the test scenarios. If an appropriate objective function can be defined for the test aim, and evolutionary computation is applied as the search technique, then the Evolutionary Test proceeds as follows.

The initial set of test scenarios is generated, usually at random. In principle, if test scenarios have been obtained by a previous test, they can also be used as an initial population. The Evolutionary Test could thus benefit from the tester's knowledge of the system under test.

Each individual within the population represents a test scenario with which the system under test is executed. For each test scenario, execution is monitored and the objective value is calculated for the corresponding individual.

Next, individuals with high objective values are selected with a higher probability than those with a lower value and are subjected to combination and mutation processes to generate new offspring individuals. It is important to ensure that the test scenarios generated are valid with respect to the input specification of the system under test. The test scenarios resulting from the offspring individuals are again evaluated by executing the system under test and monitoring test execution.

Finally, a new population of individuals is formed by merging offspring and parent individuals according to the survival procedures laid down. It is decided which parent individuals are replaced by offspring individuals.

From here on, the process repeats itself, starting with selection until the test objective is fulfilled or another given stopping condition is reached (compare Fig. 1).

In general, there are few prerequisites for the application of the Evolutionary Test. An interface specification of the system under test is required to guarantee the generation of valid test scenarios. For structural testing, the source code of the test object is required. The most important prerequisite is a numerical presentation of the testing objective, from which a suitable objective function for the evaluation of generated test scenarios can be derived. Different fitness functions emerge for the evaluation of test scenarios depending on which test aim is pursued (compare section 3 - 5). Finally, it must be possible to execute the system under test with the generated test scenarios. However, this is a general testing requirement that always has to be fulfilled when testing.

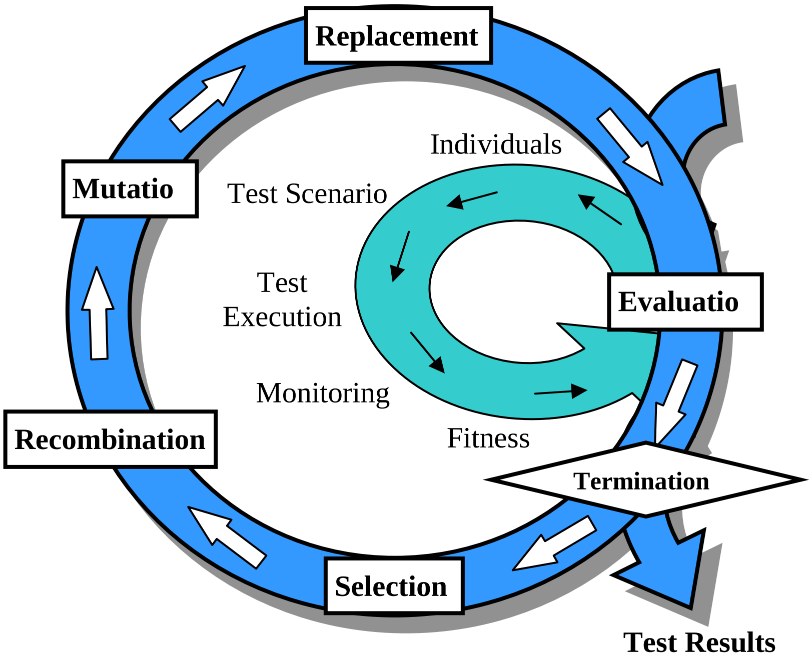


Fig. 1. Evolutionary Testing

3 Evolutionary Functional Testing

As a leading car manufacturer, DaimlerChrysler is constantly developing innovations to improve vehicle safety, quality, and comfort. Within this context, automotive systems of increasing complexity are developed and have to be tested. Examples of such applications are: Distronic (an adaptive cruise control system) with which the vehicle maintains a constant distance from a vehicle ahead, automatic vehicle parking, or emergency braking systems. In fact, all of the examples are control systems which introduce novel comfort and safety functions on the basis of measuring the distance of the vehicle to other objects – an application field for automotive systems which is becoming increasingly important. Furthermore, all the examples represent very complex systems with a multitude of components, whose function is, however, narrowly focused. For such systems, DaimlerChrysler is evaluating the possibility of using evolutionary testing to improve testing quality and to increase testing efficiency. Two applications will be described in the following: the Evolutionary Testing of an automatic parking system, and the Evolutionary Testing of a brake assistant system. The aim of functional testing is to uncover errors in the functional behaviour of the system to be tested. Whilst the objective function can be generalised for structural and real-time testing, the objective functions for functional testing are specifically tailored to the function of the respective system under test.

3.1 Evolutionary Testing of an Automatic Parking System

The automatic parking system [6] is intended to automate parking lengthways into a parking space. To this end, the vehicle is equipped with short range environmental sensors, which register objects surrounding the vehicle. On passing, the system can recognize sufficiently large parking spaces and signals to the driver that a parking space has been found. If the driver decides to park in the detected parking space the vehicle does this automatically.

Input to the automatic parking system is provided by sensor signals, which receive information on the state of the vehicle, e.g. vehicle speed or steering wheel position, as well as information from the environmental sensors, which register objects in the environment of the vehicle. The parking space detection of the automatic parking system processes the data from the environmental sensors and delivers the geometry of parking spaces detected as being sufficiently large. The automatic parking system uses the geometry data on the parking space together with the data from the vehicle sensors to steer the vehicle through the parking procedure. For this purpose, velocity and steering angle are preset for the vehicle actors. A fundamental requirement for the automatic parking system is that the vehicle controlled by the system must not damage or even touch other objects in the surrounding of the vehicle. Naturally, the fulfilment of this requirement has to be tested thoroughly before such a system may be released.

In order to test the functional behaviour of the automatic parking system an objective function has to be defined that calculates a numerical value for the parking manoeuvre driven by the system for a generated parking scenario. The objective function represents a quality figure for the driven parking manoeuvres and is intended to guide the evolutionary search towards finding faulty system behaviour. Therefore, good objective function values are assigned to parking scenarios which lead the system into a collision or end up in an inadequate parking situation, bad objective function values arise for scenarios which attain a good parking position with enough clearance from the collision area. Different objective functions could be defined to search for parking scenarios leading to faulty system behaviour, e.g. measuring the minimum distance occurring between the vehicle’s surface and the borders of the parking space during the parking manoeuvre (Fig. 2) [6]. Alternatively, the time to contact can be calculated – taking into account not only the distance to a collision but also the speed of the vehicle. The shortest time to contact measured for the parking manoeuvre could form the objective function value for the parking manoeuvre calculated by the automatic parking system. Another option is to use the area included between the driving path of the vehicle, and the geometry of the parking space [7]. In our experiments all three fitness functions led to good results.

For the testing of the automatic parking system the tests have to be performed in a simulation environment. The simulation environment simulates the properties of the vehicle and the surrounding environment. It runs with the control unit "in-the-loop" meaning that the simulation environment calculates the sensor data of the vehicle and presents it to the automatic parking system. The automatic parking system processes this sensor data and reacts to it with control data for the actors captured and evaluated in the simulation environment. In this way the complete parking manoeuvre is simulated. The parameters necessary for the simulation of a parking scenario are represented by the individuals of the evolutionary search. Parameters are, for example, positions of cars which are close by and form the parking space, the position of the controlled vehicle itself with respect to the parking space, or the driving manoeuvre performed when passing the parking space initially. After the simulation of a parking manoeuvre the objective function value is calculated for the parking manoeuvre, and assigned to the corresponding individual.

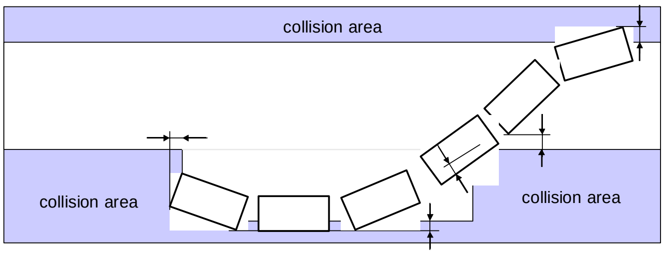


Fig. 2. Objective Function Value Calculation for a Parking Scenario with the Minimum Distance Criterion

In this way, a simulation of the corresponding parking manoeuvre is performed for each individual of a generation, determining the objective function value of the individual. When all the individuals in a generation have a fitness value, the evolutionary algorithm continues with the production of the next generation of individuals. The test ends when a parking scenario is found leading either to a collision of the vehicle with surrounding objects, to an inadequate parking position or once a predefined number of parking manoeuvres has been performed successfully without any collision.

During tests on a research prototype of the automatic parking system, different errors were found fully automatically. Examples are shown in Fig. 3. It was possible to significantly improve the quality of the prototype by eliminating the errors found.

3.2 Evolutionary Testing of a Brake Assistant System

Whilst the automatic parking system is still not yet ready for series production in customer vehicles, series production of the brake assistant system [8] described in the following, and its delivery to end customers will already begin this year.

A brake assistant is a function which helps drivers to slow down and stop their vehicles in critical situations. In many cases, drivers do not brake hard enough in an emergency situation, wasting valuable meters of braking distance. In order to support the driver in emergency braking situations, the brake assistant measures the velocity with which the driver presses the brake pedal and decides within a fraction of a second whether the driver intends to make an emergency stop. Once the brake assistant has recognized an emergency braking situation, it immediately initiates full braking pressure, regardless of the driver’s brake pedal position. In this way, the brake assistant can usually save several meters of braking distance, and the car is brought to a halt more quickly.

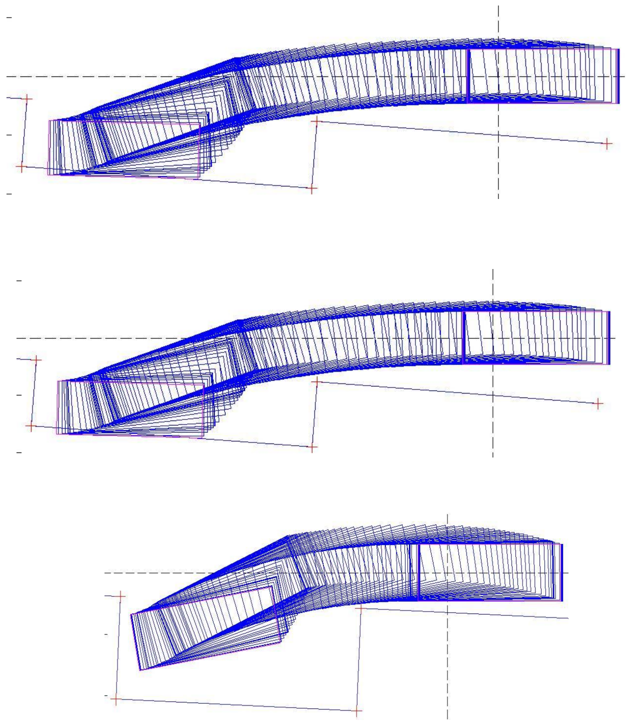


Fig. 3. Three erroneous parking manoeuvres found by the Evolutionary Testing of the automatic parking system: the first two scenarios show a collision of the vehicle with surrounding objects when the initial position of the vehicle is far away from the parking space and the orientation of the vehicle has a certain angle; the third scenario shows a scenario resulting in an inadequate parking position

Brake assistant systems of the second generation make use of environmental sensors in order to continuously monitor the vehicle environment. Thus, information such as the distance to the proceeding vehicle and the closing velocity between the vehicle and the proceeding vehicle, can be used to calculate optimal deceleration of the vehicle in order to avoid an accident. It is easily understandable that a brake assistant of the second generation is much more complex than the first generation systems.

The brake assistant system should support the driver in critical situations with a high vehicle brake retardation. On the other hand, it should not enhance the driver’s brake momentum in uncritical situations. To find errors in the functional behaviour of the brake assistant system scenarios are of interest where the brake assistant system does not enhance the brake retardation, even though the situation is critical. Furthermore, scenarios are searched for where the brake assistance system does enhance the brake retardation, even though the situation is uncritical.

Individuals of the evolutionary testing of the automatic brake system are representing different scenarios varying for instance the velocities of the two vehicles involved and the distance between the vehicles.

For the testing of the new brake assistant system the objective function is based on two quantities: the time-to-collision and the brake momentum added by the brake assistant system. The time-to-collision is often considered in the context of collision-critical situations. It describes the time before a collision happens and can be seen as a degree of how critical a braking situation is. For the brake assistant system the calculation of the time-to-collision is based on the closing velocity and the distance between the proceeding vehicle and the own vehicle. The brake momentum is the brake momentum added to the driver's brake momentum by the brake assistance system. For the objective function the added momentum multiplied with the time-to-collision is integrated over the time of the braking manoeuvre [8]. The objective function can be maximized in order to find scenarios with long time-to-collision but nevertheless a high brake momentum added, and it can be minimized to find critical situations with short time-to-collision but a low brake momentum added.

Again, the tests were performed in a simulation environment, and scenarios where found fully automatically where the behaviour of the brake assistant system could be improved. The erroneous situations found were unlikely to be detected with conventional testing techniques [8].

4 Evolutionary Structural Testing

Structural testing is widespread in industrial practice and stipulated in many software development standards. The idea behind structural testing is that for a thorough test of a system all internal program structures shall be executed at least once. Depending on the structural testing criteria chosen, the internal program structures to be covered by the test differ. Statement, branch, and condition testing are common examples aiming at the execution of every program statement, or every program branch, or at evaluating every program condition at least once as True and False.

The aim of applying evolutionary testing for the automation of structural testing is the generation of a quantity of test scenarios, leading to the highest possible coverage for the selected structural testing criterion. The objective function definitions for structural testing can be generalized.

In order to apply evolutionary testing to the automation of structural testing, the test is split up into partial aims. The identification of the partial aims is based on the control-flow graph of the system under test. Each partial aim represents a program structure that needs to be executed to achieve full coverage, e.g. a statement, a branch, or a condition with its possible logical values. For each partial aim an individual objective function is formulated and a separate optimization is performed to search for a test scenario executing the partial aim. The set of test scenarios found for the partial aims then serves as the test scenario set for the coverage of the structure test criterion.

In order to direct the search toward program structures not covered, the objective function computes a distance for each individual that indicates how far away it is from executing the desired program structure. Individuals closer to the execution of the desired program structure are selected as parents and combined to produce offspring individuals. The objective functions of the partial aims consist of two components – the approach level and the branch distance. The strategy in which approach level and branch distance are computed varies slightly according to the coverage type in question. The approach level supplies a figure for an individual that gives the number of branching nodes lying between program structures covered by the individual and the desired program structure. For this computation, only branching nodes which contain an outgoing edge resulting in a lack of the desired program structure are taken into account (see Fig. 4). In addition, the calculation of the branch distance is performed in order to distinguish between different individuals missing the desired program structure at the same program condition (in Fig. 4 this could be the branching node at approach level 2). If during execution an undesired branch is taken – one which deviates from the desired path – the branch distance is computed using an objective function derived from the predicate of the desired, alternative branch. The branch distance describes how “close" the predicate is to being fulfilled in the desired way. Thus, a distance to the execution of the sibling branch is calculated for the individual by means of the branching conditions in the branching node in which the target node is missed, e.g. if a branching condition x==y is always evaluated as False but needs to be evaluated as True to reach the desired partial aim, then the branch distance may be defined as |x-y|.

For the calculation of the objective function values the system under test is instrumented with statements supporting the calculation of approach levels and

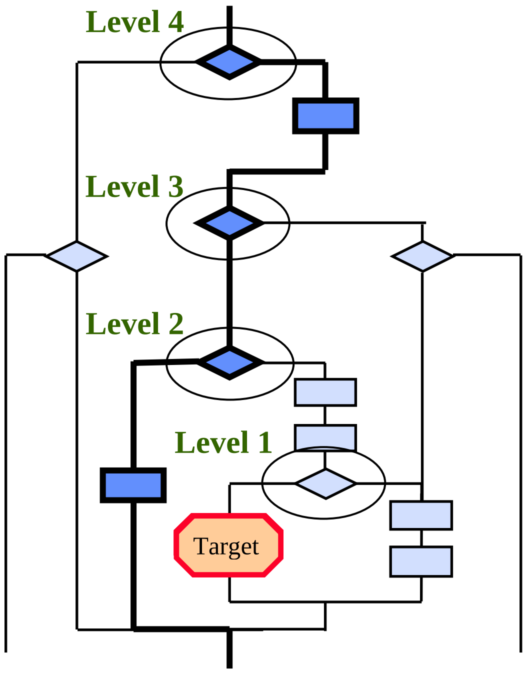


Fig. 4. Objective Function Calculation for Evolutionary Structural Testing: the control-flow executed by a generated individual is shown in bold. The target node is missed at approach level 2. Different individuals with an approach level of 2 will be further distinguished by calculating the branch distance for the branching condition at approach level 2. The branch distance describes how far the generated individual is from reaching the approach level 1.

branch distances. The instrumentation remains the same for all partial aims. The objective function values for the individuals are calculated on the basis of the monitoring results provided by the statements added through the instrumentation. A detailed definition of the objective functions and the test environment developed at DaimlerChrysler to automate structural testing can be found in [9] and [10].

Although only one partial aim after the other is processed by the evolutionary test, the execution of a test scenario usually leads to passing several partial aims. Thus, the test soon focuses on those program structures which are difficult to reach. After the processing of all partial aims, the tester is provided with a minimal amount of test scenarios, leading to an execution of all partial aims reached.

5 Evolutionary Real-Time Testing

Most software-based systems in the automotive industry are subject to temporal requirements. This is due to reasons of operational comfort, e.g. short system reaction times to user commands, or due to the requirements of technical processes that are controlled by the system, e.g. in engine control systems, airbag controllers or vehicle dynamics control systems. Therefore, most software-based systems integrated into the vehicles have to be thoroughly tested not only with regard to their functional behaviour, but also to detect existing deficiencies in fulfilling the systems' real-time requirements.

Existing test methods are unsuitable for the examination of temporal correctness. Even for an experienced tester, it is virtually impossible to find the critical test scenarios with respect to the temporal behaviour of the systems by analysing and testing the behaviour of complex systems manually. Effects of modern processor architectures with pipelining, data caching and instruction caching as well as the influences of system calls, parallelism, and optimizing compilers on the temporal behaviour of a system can hardly be assessed by the tester. Therefore, real-time testing is another important and promising application area for Evolutionary Testing.

When testing the temporal behaviour of systems, the objective is to check whether input situations exist for which the system violates its specified timing constraints. Usually, a violation occurs because outputs are produced too early or their computation takes too long. The task of the tester and therefore of the Evolutionary Test is to find input situations with especially long or short execution times in order to check whether a violation of the real-time requirements can be produced. Evolutionary Testing enables a fully automated search for extreme execution times.

When using Evolutionary Testing for the testing of temporal behaviour, the execution time is measured for every generated test scenario. The fitness evaluation of the individuals is based on the execution times measured for the corresponding test scenario. If one searches for long execution times, individuals with long execution times obtain high objective function values. Conversely, when searching for short execution times, individuals with short execution times obtain high values. Individuals with long or short execution times are selected depending on the objective of the test and are combined in order to obtain test scenarios with even longer or shorter execution times. The test is terminated if a violation of the system's real-time requirements is detected or a previously specified termination criterion is reached. If a violation of the system's predetermined temporal limits is detected, the test was successful and the system has to be corrected, usually by detailed performance analyses and optimizations of the temporal characteristics of the program code causing the deviation.

The application of Evolutionary Testing for the examination of temporal behaviour has been evaluated successfully in many case studies on the task level (e.g. [2], [11], [12], [13], [14]). The performance of the Evolutionary Test was superior to tests with other testing methods such as functional testing, structural testing and random testing.

6 Conclusion and Future Work

Evolutionary Testing is a promising approach to fully automating the testing of various testing objectives. The definition of the objective function depends on the testing objective addressed by the Evolutionary Test. This paper described the application of Evolutionary Testing to functional testing, structural testing and real-time testing. Whereas the definition of the objective function is usually specific for a certain system under test for functional testing, it can be generalized for structural testing and real-time testing.

Evolutionary Testing is based on the idea of searching test scenarios relevant for the testing objective considered in the input domain of the system under test with the help of evolutionary algorithms. Due to the high degree of automation achievable, the system under investigation can be tested with a large number of different input situations. In most cases more than several thousand test scenarios are generated and executed within a few minutes. Evolutionary Testing thus contributes to quality improvement as well as to the reduction of development costs for software-based systems. Effectiveness and efficiency of the test can be increased.

In the future, we intend to expand the application of evolutionary functional tests to further vehicle systems such as the adaptive cruise control or automatic emergency braking systems. We also intend to research the interaction between evolutionary functional tests and structure tests more intensively. This should answer questions such as: Which coverage is achieved with the generated functional tests? Does the seeding of functionally determined test scenarios prove useful for an evolutionary structure test and, on the other hand, does the seeding of structure-oriented test scenarios increase the test quality of the evolutionary functional test?

For the structural testing current and future work aims at further improvements for different program characteristics hindering the successful search for coverage-oriented test scenarios. For example, the occurrence of flags in the branching conditions of a program makes a distance calculation which would support a guidance of the search impossible – the branch distance is one for all individuals reaching the branching node. In cases where the value of a flag is calculated before the branching condition, the application of program transformations that replace a flag by its semantic meaning seems possible ([15], [16]).

A further problem for the test is the short circuit evaluation in C which breaks off the evaluation of atomic predicates if logical dependencies are linked by *&&* or ||, as soon as the value of the entire condition is definite. This leads to an artificial narrowing of the search domain since the linked predicates can only be optimised one after the other. For the condition *A && B && C*, *A* and *B* need to be evaluated as *True* first, before the evaluation of the generated individual with regard to the predicate *C* is possible. The search for individuals that fulfil *A && B* leads to an artificial focusing on the individuals, without taking the predicate *C* into account. The search for individuals that fulfil all three predicates becomes more complicated. One possibility for solving this problem is the parallel evaluation of single predicates. For this, however, one has to ensure that single predicates do not contain any side effects.

By introducing data-flow analyses that determine which input parameters every partial aim relies on, testing efficiency can be further improved. Ideally, it is possible to drastically reduce the size of the search domain for the selected partial aim, which in turn accelerates the search considerably, if only a few input parameters are relevant for the attainment of the partial aim [17].

Future work on real-time testing will focus on the application of Evolutionary Testing at the integration and system test level. Seeding experiments will be performed integrating existing functional and structural test scenarios into the Evolutionary Test. Furthermore, combinations of structural testing and real-time testing will be explored to ensure that all program parts are covered during the temporal behaviour test.

For all testing objectives the consideration of internal states of the system under test would be a significant improvement, since important scenarios might only occur in certain system states [18]. Therefore, it would be useful to incorporate state-knowledge in the objective functions for the different testing objectives.

References

1. Dijkstra, E. W., Dahl, O. J., Hoare, C. A. R.: Structured programming, Academic Press., 1972.
2. Wegener, J., and Grochtmann, M.: Verifying timing constraints of real-time systems by means of Evolutionary Testing. Real-Time, Systems, vol. 15, no. 3, Kluwer Academic Publishers, pp. 275-298, 1998.
3. Jones, B.-F., Sthamer: H.-H., Eyres, D.: Automatic structural testing using genetic algorithms. Software Engineering Journal, vol. 11, no. 5, pp. 299-306, 1996.
4. Sthamer, H.-H.: The automatic generation of software test data using genetic algorithms, PhD Thesis, University of Glamorgan, Pontyprid, Wales, Great Britain, 1996.
5. Tracey, N., Clark, J., Mander, K. and McDermid, J.: An automated framework for structural test-data generation. Proceedings of the 13th IEEE Conference on Automated Software Engineering, Hawaii, USA 1998.
6. Buehler, O. and Wegener, J.: Evolutionary functional testing of an automated parking system. Proceedings of the International Conference on Computer, Communication and Control Technologies and the 9th. International Conference on Information Systems Analysis and Synthesis, Orlando, Florida, USA, 2003.
7. Buehler, O. and Wegener, J.: Automatic testing of an autonomous parking system using evolutionary computation. SAE World Congress, Detroit, USA, 2004.
8. Buehler, O. and Wegener, J.: Evolutionary functional testing of a vehicle brake assistant system. 6th Metaheuristics International Conference, Vienna, Austria, 2005.
9. Wegener, J., Baresel, A.; Sthamer, H.: Evolutionary test environment for automatic structural testing. Special Issue of Information and Software Technology devoted to the Application of Meta-heuristic Algorithms to Problems in Software Engineering 2001.

10. McMinn, P.: Search-based software test data generation: a survey. Software Testing, Verification and Reliability, vol. 14, no. 2, pp. 105-156, 2004.
11. Wegener, J. and Mueller, F.: A comparison of static analysis and evolutionary testing for the verification of timing constraints. Real-Time Systems, vol. 21, no. 3, pp. 241-268, 2001.
12. Puschner, P. and Nossal, R.: Testing the results of static worst-case execution-time analysis. In Proceedings of the 19th IEEE Real-Time Systems Symposium, pp. 134-143, Madrid, Spain, 1998.
13. Tracey, N., Clark, J. and Mander, K.: The way forward for unifying dynamic test-case generation: The optimisation-based approach. International Workshop on Dependable Computing and Its Applications, pp. 169-180, 1998.
14. Gross, H.-G.: Evolutionary testing in component-based real-time system construction. In Proceedings of the Genetic and Evolutionary Computation Conference. Late Breaking Papers, pp. 207-214, New York, USA, 2002.
15. Harman, M., Hu, L., Hierons, R., Baresel, A. and Sthamer, H.: Improving evolutionary testing by flag removal. In Proceedings of the Genetic and Evolutionary Computation Conference, pp. 1359-1366, New York, USA, 2002. Morgan Kaufmann.
16. Baresel, A. and Sthamer, H.: Evolutionary testing of flag conditions. In Proceedings of the Genetic and Evolutionary Computation Conference, Lecture Notes in Computer Science 2724, pp. 2442-2454, Chicago, USA, 2003. Springer-Verlag.
17. Harman, M., Fox, C., Hierons, R., Hu, L., Danicic, S., and Wegener, J.: VADA: A transformation-based system for variable dependence analysis. 2nd IEEE International Workshop on Source Code Analysis and Manipulation, pp. 55-64, Montreal, Canada, 2002.
18. McMinn, P. and Holcombe, M.: The state problem for evolutionary testing. In Proceedings of the Genetic and Evolutionary Computation Conference, Lecture Notes in Computer Science 2274, pages 2488-2497, Chicago, USA, 2003. Springer-Verlag.

Optimal Fuzzy CLOS Guidance Law Design Using Ant Colony Optimization

Hadi Nobahari and Seid H. Pourtakdoust

Sharif University of Technology, P.O. Box: 11365-8639, Tehran, Iran
`nobahari@mehr.sharif.edu`
`pourtak@sharif.edu`

Abstract. The well-known ant colony optimization meta-heuristic is applied to design a new command to line-of-sight guidance law. In this regard, the lately developed continuous ant colony system is used to optimize the parameters of a pre-constructed fuzzy sliding mode controller. The performance of the resulting guidance law is evaluated at different engagement scenarios.

1 Introduction

The principle of Command to Line-of-Sight (CLOS) guidance law is to force the missile to fly as nearly as possible along the instantaneous line joining the ground tracker and the target, called the Line-of-Sight (LOS) [1,2,3,4,5,6,7,8,9]. Theoretically, the missile-target dynamic equations are nonlinear and time-varying, partly because the equations of motion are described in an inertial frame, while aerodynamic forces and moments are represented in the missile and target body frames. Many different control techniques have been used to design different CLOS guidance laws, examples of which are optimal control theory [3,6], feedback linearization [4], polynomial method [5], supervisory control [9], and so on. However, these methods have resulted in rather complicated controllers, and some of them require the knowledge of the maneuvering model of the target.

In recent years CLOS guidance laws based on Fuzzy Logic Control (FLC) have been presented [7,8]. Fuzzy logic was proposed by Professor Lotfi Zadeh in 1965, at first as a way of processing data by allowing partial set membership rather than crisp membership. Soon after, it was proven to be an excellent choice for many control system applications since it mimics human control logic. FLC can model the qualitative aspects of human knowledge and reasoning processes without employing precise quantitative analyses. It also possesses several advantages such as robustness and being a model-free, universal approximation and a rule-based algorithm. However, the stability analysis for general FLC systems is still lacking. To cope with this deficiency, a combination of FLC and the well-known Sliding Mode Control (SMC) has been proposed in recent years, called Fuzzy Sliding Mode Control (FSMC) [10,11,12]. The stability of FSMC can be proved in the Lyapunov sense [12]. This technique has been widely used in many control applications, as well as the CLOS guidance problem [8]. The other advantage of the FSMC is that it has fewer rules than FLC. Moreover, by using

SMC, the system possesses more robustness against parameter variations and external disturbances.

There are some limitations to the development of fuzzy controllers, the most important of which is the knowledge does not always completely exist and the manual tuning of all the base parameters takes time. The lack of portability of the rule bases when the dimensions of the control system change, makes the later difficulty still more serious. To cope with these problems, the learning methods have been introduced. The first attempt was made by Procyk and Mamdani in 1979 , with a "self tuning controller" [13]. The gradient descent method was used by Takagi and Sugeno in 1985 as a learning tool for fuzzy modeling and identification [14]. It was used by Nomura, et al. in 1991 as a self tuning method for fuzzy control [15].

The gradient descent method is appropriate for simple problems and real time learning, since it is fast. But it may be trapped into local minima. Also the calculation of the gradients depends on the shape of membership functions employed, the operators used for fuzzy inferences as well as the selected cost function. In 1998, Siarry and Guely used the well-known Genetic Algorithm to tune the parameters of a Takagi-Sugeno fuzzy rule base [16]. The same problem was solved by Nobahari and Pourtakdoust [17], using Continuous Ant Colony System (CACS) [18], which is an adaptation of the well-known Ant Colony Optimization (ACO) meta-heuristic to continuous optimization problems.

In this paper, CACS is used to optimize the parameters of a FSMC-CLOS guidance law. The optimization problem is to minimize the average tracking error obtained for 10 randomly generated engagement scenarios. In the simulation of these scenarios, the proposed random target maneuver in [3] is used. The performance of the optimal FSMC-CLOS, designed in this way, is then evaluated at some other engagement scenarios, considering both maneuvering and non-maneuvering targets. The simulation results show a good performance in both tracking dynamics and the final miss distance.

2 Problem Formulation

In this section the three-dimensional CLOS guidance is formulated as a non-linear time varying tracking problem. The three-dimensional pursuit situation is depicted in Fig. 1. The origin of the inertial frame is located at the ground tracker. The Z_I axis is vertical upward and the $X_I - Y_I$ plane is tangent to the Earth surface. The origin of the missile body frame is fixed at the missile center of mass with the X_I axis forward along the missile centerline.

Defining the LOS frame as depicted in Fig. 2, the three-dimensional guidance problem can be converted to a tracking problem. According to the definition of CLOS guidance law, reasonable choices for the tracking errors may be $\Delta\sigma = \sigma_t - \sigma_m$ and $\Delta\gamma = \gamma_t - \gamma_m$. The problem involves designing a controller to drive $[\Delta\sigma, \Delta\gamma]^T$ to zero. The same design algorithm will be applied for both azimuth and elevation angle control. In the following the azimuth angle control is chosen as an example.

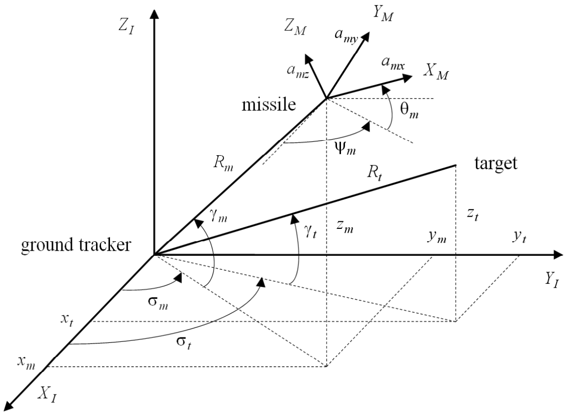


Fig. 1. Three-dimensional pursuit situation

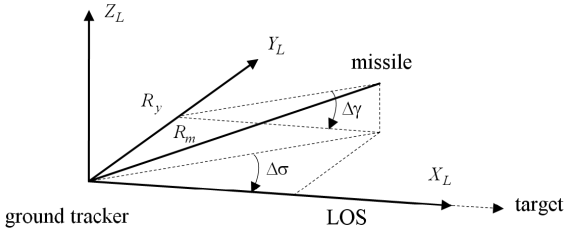


Fig. 2. Definition of the tracking error

Assume that $e(t) = \Delta\sigma$ represents the azimuth loop tracking error. The differential equation of the tracking error can be derived as [9]

$$\ddot{e} = f(\underline{e}, t) + (1/R_m(t))u(t) \tag{1}$$

Where $\underline{e} = [e(t), \dot{e}(t)]^T$ is the error state vector, $R_m(t)$ is missile to tracker range, $u(t) = -a_m(t)$ is the control variable, and

$$f(\underline{e}, t) = -(\ddot{R}_m(t)/R_m(t))e(t) - (2\dot{R}_m(t)/R_m(t))\dot{e}(t) + (1/R_m(t))a_t(t) \tag{2}$$

3 Sliding Mode Control

Let $s(\underline{e}) = 0$ denotes a hyper-surface in the space of the error state, which is called the sliding surface. The purpose of the sliding mode control is to force the error vector $\underline{e}$ approach the sliding surface and then move along it to the origin. If the sliding surface is stable, the error $\underline{e}$ will die out asymptotically. In this regard, let the sliding surface s be defined as follows

$$s(e, \dot{e}) = \left(\frac{d}{dt} + \lambda\right) e = \dot{e} + \lambda e \tag{3}$$

where λ is a positive constant. It is obvious from Eq. (3) that keeping the states of the system on the sliding surface will guarantee the tracking error vector $\underline{e}$ asymptotically approach to zero. The corresponding sliding condition [19] is

$$\frac{1}{2}\frac{d}{dt}s^2 = s\dot{s} \leq 0 \tag{4}$$

The general control structure that satisfies the stability condition of the sliding motion, can be written as [19]

$$u = \hat{u} - Ksgn(s) \tag{5}$$

where $\hat{u}$ is called the equivalent control law that is derived by setting $s = \dot{s} = 0$, and K is a positive constant. The sliding condition can be satisfied as long as K is chosen large enough [19].

3.1 Fuzzy Sliding Mode Control

As mentioned in section 1, the FSMC is a hybrid controller and inherits the advantages of both fuzzy and sliding mode controllers. The FSMC can be regarded as a fuzzy regulator that controls the variable s approach to zero. Let **s** denote the fuzzy variable of the universe of discourse, s. Then some linguistic terms can be defined to describe the fuzzy variable **s**, such as "zero", "positive large", "negative small", etc. Each linguistic term expresses a certain situation in the system. For example, **s** is "zero" means that the state of system is on the sliding surface or is near to the sliding surface. Such linguistic expressions can be used to form fuzzy control rules as below

Rule 1:	If **s** is NB, then **u** is PB	
Rule 2:	If **s** is NM, then **u** is PM	
Rule 3:	If **s** is ZO, then **u** is ZO	(6)
Rule 4:	If **s** is PM, then **u** is NM	
Rule 5:	If **s** is PB, then **u** is NB	

where **u** denotes the fuzzy variable of the control signal u, NB denotes "Negative Big", NM denotes "Negative Mid", ZO denotes "Zero", PM denotes "Positive Mid", and PB denotes "Positive Big". Each linguistic term is described by an associated membership function as shown in Fig. 3. In conventional fuzzy controller design, these membership functions are tuned by a trial-and-error procedure, based on certain physical sense or designer's experiences.

3.2 Definition of FSMC Design as an Optimization Problem

The FSMC considered here, involves a SISO fuzzy inference system with the variable **s** as the input, and the variable **u** as the output. The design problem is defined as finding the optimum values of the membership functions parameters. The more fuzzy terms are defined, the more fuzzy rules will be requested for

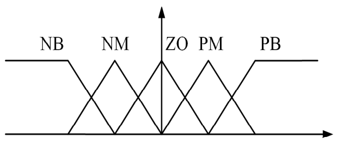


Fig. 3. Definition of fuzzy membership functions

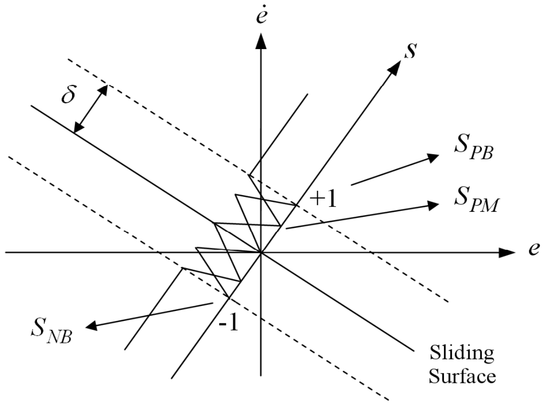


Fig. 4. Fuzzy sliding surface of a second order system

completeness. In this paper, five fuzzy terms are proposed for the measure of **s**, as follows

$$\mathbf{s} = \{\text{NB, NM, ZO, PM, PB}\} \tag{7}$$

One may choose more fuzzy terms if required. The associated five fuzzy terms for control energy are defined in a similar way as

$$\mathbf{u} = \{\text{NB, NM, ZO, PM, PB}\} \tag{8}$$

A scaling factor will be used to normalize the fuzzy set. The fuzzy sliding surface of a second order system is shown in Fig. 4, where δ represents the scaling factor. For normalized fuzzy sets, $S_{\text{NB}} = -1$, $S_{\text{PB}} = 1$ and $S_{\text{ZO}} = 0$ can be defined. The maximum control energy $U_{\max}$ is also bounded by physical limitations, that is $U_{\text{NB}} = -1$ and $U_{\text{PB}} = 1$ can also be defined. If the symmetry of fuzzy terms corresponding to **s** and **u** is assumed, the remaining design factors of the fuzzy system are the values of S_{PM}, U_{PM}, and the scaling factor δ. The optimal design problem of the FSMC then can be formulated as: to determine S_{PM}, U_{PM}, δ and the positive constant λ such that a given cost function is optimized.

4 Ant Colony Optimization

Ant algorithms were inspired by the observation of real ant colonies. An important and interesting behavior of ant colonies is their foraging behavior, and in particular, how ants can find the shortest path without using visual cues. While walking from the food sources to the nest and vice versa, ants deposit on the ground a chemical substance called *pheromone* which makes a pheromone trail. Ants use pheromone trails as a medium to communicate with each other. They can smell pheromone and when they choose their way, they tend to choose paths with more pheromone. The pheromone trail allows the ants to find their way back to the food source or to the nest. Also, the other ants can use it to find the location of the food sources, which are previously found by their nest mates.

4.1 Ant Colony System

Ant Colony System (ACS) is one of the first discrete algorithms proposed based on ACO. At first it was applied to the well-known Traveling Salesman Problem (TSP) which is a discrete optimization problem. In this part we will shortly review the basic idea of ACS. Then in the subsequent part, the continuous version of ACS will be presented.

Ant Colony System uses a graph representation, like as the cities and the connections between them in TSP. In addition to the cost measure, each edge has also a desirability measure, called *pheromone intensity.* To solve the problem, each ant generates a complete tour by choosing the nodes according to a so called *pseudo-random-proportional state transition rule*, which has two major features. Ants prefer to move to the nodes, which are connected by the edges with a high amount of pheromone, while in some instances, their selection may be completely random. The first feature is called *exploitation* and the second one is a kind of *exploration.* While constructing a tour, ants also modify the amount of pheromone on the visited edges by applying a *local updating rule.* It concurrently simulates the *evaporation* of the previous pheromone and the *accumulation* of the new pheromone deposited by the ants while they are building their solutions. Once all the ants have completed their tours, the amount of pheromone is modified again, by applying a *global updating rule.* Again a part of pheromone evaporates and all edges that belong to the global best tour, receive additional pheromone conversely proportional to their length.

4.2 Continuous Ant Colony System

In this part, the lately developed Continuous Ant Colony System (CACS) is introduced. The interested readers can refer to [17,18] to find more details.

A continuous optimization problem is defined as finding the absolute minimum of a positive non-zero continuous cost function $f(x)$, within a given interval $[a, b]$, in which the minimum occurs at a point x_s. In general f can be a multi-variable function, defined on a subset of $\mathbb{R}^n$ delimited by n intervals $[a_i, b_i]$, $i = 1, ..., n$.

Continuous Ant Colony System (CACS) has all the major features of ACS, but certainly in a continuous frame. These are a pheromone distribution model, a state transition rule, and a pheromone updating rule. In the following these features are introduced.

Continuous Pheromone Model. Although pheromone distribution has been first modeled over discrete sets, like the edges of TSP, in the case of real ants, pheromone deposition occurs over a continuous space. The ants aggregation around the food source causes the most pheromone intensity to occur at the food source position. Then increasing the distance of a sample point from the food source will countinuously decreases its pheromone intensity. CACS models the pheromone intensity, in the form of a normal Probability Distribution Function (PDF):

$$\tau(x) = e^{-\dfrac{(x - x_{\min})^2}{2\sigma^2}} \tag{9}$$

where $x_{\min}$ is the best point found within the interval $[a, b]$ from the beginning of the trial and σ is an index of the ants aggregation around the current minimum.

State Transition Rule. In CACS, pheromone intensity is modeled using a normal PDF, the center of which is the last best global solution and its variance depends on the aggregation of the promising areas around the best one. So it contains exploitation behavior. In the other hand, a normal PDF permits all points of the search space to be chosen, either close to or far from the current solution. So it also contains exploration behavior. It means that ants can use a random generator with a normal PDF as the state transition rule to choose the next point to move to.

Pheromone Update. During each iteration, ants choose their destinations through the probabilistic strategy of Eq. (9). At the first iteration, there is no knowledge about the minimum point and the ants choose their destinations only by exploration. It means that they must use a high value of σ, associated with an arbitrary $x_{\min}$, to approximately model a uniform distribution function. During each iteration pheromone distribution over the search space will be updated using the acquired knowledge of the evaluated points by the ants. This process gradually increases the exploitation behavior of the algorithm, while its exploration behavior will decrease. Pheromone updating can be stated as follows: The value of objective function is evaluated for the new selected points by the ants. Then, the best point found from the beginning of the trial is assigned to $x_{\min}$. Also the value of σ is updated based on the evaluated points during the last iteration and the aggregation of those points around $x_{\min}$. To satisfy simultaneously the fitness and aggregation criteria, a concept of weighted variance is defined as follows:

$$\sigma^2 = \frac{\displaystyle\sum_{j=1}^{k} \frac{1}{f_j - f_{\min}} (x_j - x_{\min})^2}{\displaystyle\sum_{j=1}^{k} \frac{1}{f_j - f_{\min}}} \tag{10}$$

where k is the number of ants. This strategy means that the center of region discovered during the subsequent iteration is the last best point and the narrowness of its width depends on the aggregation of the other competitors around the best one. The closer the better solutions get (during the last iteration) to the best one, the smaller σ is assigned to the next iteration.

During each iteration, the height of pheromone distribution function increases with respect to the previous iteration and its narrowness decreases. So this strategy concurrently simulates pheromone accumulation over the promising regions and pheromone evaporation from the others, which are the two major characteristics of ACS pheromone updating rule.

5 Numerical Results

In this section CACS is applied to optimize the parameters of a FSMC-CLOS guidance law, and the performance of the designed optimal guidance law is evaluated through different engagement scenarios. Ten different randomly generated engagement scenarios are used to evaluate each design point discovered by the ants. The cost function is defined as the average of the normalized tracking errors over the considered engagement scenarios. The normalized tracking error is defined as follows

$$r_n = \frac{1}{t_f}\int_0^{t_f} r(t)dt \tag{11}$$

where $r(t)$ is the distance between the missile and the LOS at time t. The average of the normalized tracking errors is defined as follows

$$y = \left(\frac{1}{10}(r_{n_1}^2 + r_{n_2}^2 + ... + r_{n_{10}}^2)\right)^{\frac{1}{2}} \tag{12}$$

5.1 Mathematical Model of Missile and Target

The proposed equations of motion in [4] are used to simulate the behavior of missile and target. The acceleration limits of missile and target are $20(g)$ and $5(g)$, respectively. Other data used in simulations are the same as those in [4].

A random target maneuver similar to that proposed in [3], is utilized in simulations. It is assumed that target maneuvers about the LOS in a random fashion defined by RMS and bandwidth of the target acceleration. A stochastic representation of this maneuver will be generated by passing white noise, n_t, through a third order Butter-worth filter (Fig. 5). The values of K_t and ω_t, used in simulations, are $500(m/s^2)$ and $1(rad/s)$, respectively.

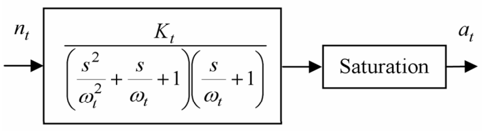

n_t
$\frac{K_t}{\left(\frac{s^2}{\omega_t^2}+\frac{s}{\omega_t}+1\right)\left(\frac{s}{\omega_t}+1\right)}$
Saturation
a_t

Fig. 5. Stochastic target maneuver model

5.2 Generation of the Random Engagement Scenarios

The cost function is defined based on the average performance obtained over 10 randomly generated engagement scenarios. These scenarios are pre-generated because the same situations are needed to evaluate different design points. In the generation of these primary scenarios, the following constraints are made

$$\begin{array}{cc}
3000 \leq R_{0t} \leq 5000 & R_{0m} = 50 \quad m \\
-180^\circ \leq \sigma_{0t} \leq 180^\circ & -5^\circ \leq \sigma_{0m} - \sigma_{0t} \leq 5^\circ \\
20^\circ \leq \gamma_{0t} \leq 70^\circ & -5^\circ \leq \gamma_{0m} - \gamma_{0t} \leq 5^\circ \\
300 \leq V_{0t} \leq 500 & V_{0m} = 150 m/s \\
-45^\circ \leq \psi_{0t} \leq 45^\circ & \psi_{0m} = \sigma_{0m} \\
-20^\circ \leq \theta_{0t} \leq 20^\circ & \theta_{0m} = \gamma_{0m}
\end{array}$$

In addition to the pre-generated initial conditions, for each scenario, two long lists of normalized zero-mean Gaussian random numbers are also generated and stored as the inputs of the target maneuver model, corresponding to a_{ty} and a_{tz}.

5.3 Optimization Results

The proposed CACS works based on the search and evaluation of different points in the solution space in a stochastic intelligent manner. The evaluation is done through the simulation of the missile-target engagement at the primary scenarios. The optimization problem can be defined as: find the values of S_{PM}, U_{PM}, δ, and λ such that the cost function, y, is minimized. The boundaries of the search space are defined as follows

$$0 < S_{\text{PM}} < 1, 0 < U_{\text{PM}} < 1, 0 < \delta < 1, 0 < \lambda < 10$$

Fig. 6 shows the history of y for different number of ants. The best results have obtained using 10 ants which is consistent with our previous results in

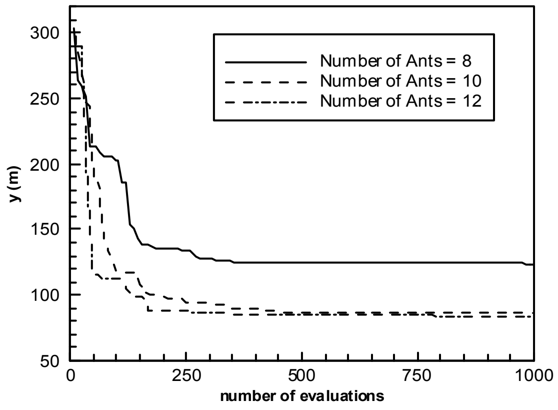


Fig. 6. History of the cost function as shown for different number of ants

[17,18]. The optimum values of the parameters obtained after 1000 evaluations are: $S_{\mathrm{PM}} = 0.0074$, $U_{\mathrm{PM}} = 0.9937$, $\delta = 0.3029$, and $\lambda = 2.74$.

5.4 Evaluation of the Optimal Design

The optimal set of parameters obtained based on the primary engagement scenarios, is again evaluated over two other scenarios. The two scenarios proposed in [4,8,9] are considered here. The first one represents a non-maneuvering target, while in the second one the target maneuvers with $a_{ty} = 5g$ and $a_{tz} = -g$ for the first 2.5 sec., and then $a_{ty} = -5g$ and $a_{tz} = 5g$ until interception. The initial condition data used to simulate these scenarios is given in table 1.

Figures 7 and 8 show the dynamic simulation results. It is clear that the new optimal FSMC-CLOS guidance law, designed using CACS, successfully drives the tracking error and as a result, the miss distance to zero. The obtained values

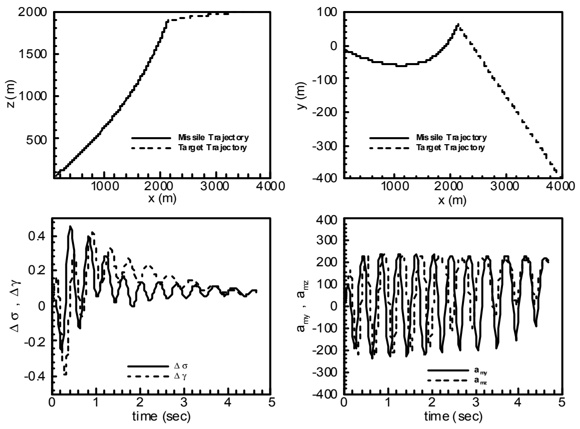


Fig. 7. Simulation results for the first scenario

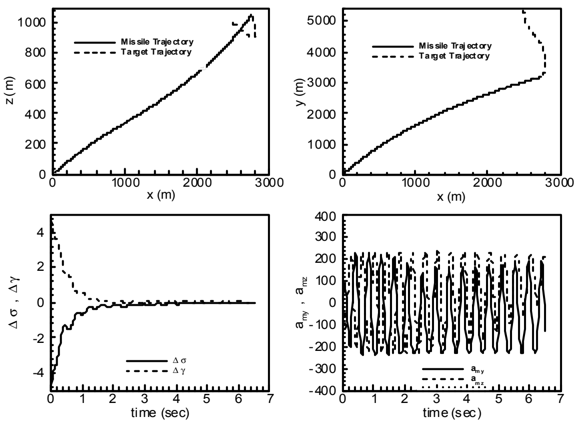


Fig. 8. Simulation results for the second scenario

Table 1. The initial condition used for different flight scenarios

Parameter	Unit	Scenario 1	Scenario 2
$x_t(0), y_t(0), z_t(0)$	m	$4000, -400, 2000$	$2500, 5361.9, 1000$
$\dot{x}_t(0), \dot{y}_t(0), \dot{z}_t(0)$	m/s	$-400, 100, 0$	$0, -340, 0$
$\psi_t(0), \theta_t(0)$	deg	$165.96, 0$	$-90, 0$
$x_m(0), y_m(0), z_m(0)$	m	$100, -10, 50$	$14.32, 39.34, 3.36$
$\dot{x}_m(0), \dot{y}_m(0), \dot{z}_m(0)$	m/s	$100, -10, 50$	$70.84, 151.92, 28.32$
$\psi_m(0), \theta_m(0)$	deg	$-5.71, 26.56$	$65, 9.59$
$\Delta\sigma(0), \Delta\gamma(0)$	deg	$0, 0$	$-5, 5$

of miss distance for these two scenarios are 5.40 m and 3.96 m, respectively. The obtained results verify the ability of CACS to solve practical optimization problems such as guidance and control systems design.

6 Conclusion

In this paper the Continuous Ant Colony System (CACS), which is based on the well-known Ant Colony Optimization meta-heuristic was applied to design an optimal FSMC-CLOS guidance law. The optimization was done for different number of ants. The evaluation of each discovered point within the design space was done through the simulation of missile-target engagement at a number of randomly generated scenarios. The cost function was defined as the average of normalized tracking errors, corresponding to each scenario. Then the performance of the resulting optimal FSMC-CLOS guidance law was studied through some other new scenarios. Simulation results indicate a good performance for the new guidance law. This again shows the ability of CACS to solve practical optimization problems. The main advantage of CACS with respect to the other meta-heuristics such as Genetic Algorithm is its simplicity which is mainly due to its simple structure. CACS has only one control parameter which is the number of ants. This makes the parameter setting easier than many other optimization methods.

References

1. Garnell, P.: Guided Weapon Control Systems, 2^{nd} Edition, Pergamon Press, Oxford, England, UK. (1980) chap. 7
2. Zarchan, P.: Tactical and Strategic Missile Guidance, 3^{rd} Edition, AIAA Education Series. **176** (1997) 193–205
3. Kain, J. E., Yost, D. J.: Command to Line-of-Sight Guidance: A Stochastic Optimal Control problem. AIAA Guidance and Control Conference Proceedings (1976) 356–364
4. Ha, I. J., Chong, S.: Design of a CLOS Guidance Law via Feedback Linearization. IEEE Transactions on Aerospace and Electronic Systems **28** (1) (1992) 51–62

5. Parkes, N. E., Roberts, A. P.: Application of Polynomial Methods to Design of Controllers for CLOS Guidance. IEEE Conference on Control Applications **2** (1994) 1453–1458
6. Pourtakdoust, S. H., Nobahari, H.: Optimization of LOS Guidance for Surface-to-Air Missiles. Iranian Aerospace Organization Conference **2** (2000) 245–257
7. Arvan, M. R., Moshiri, B.: Optimal Fuzzy Controller Design for an Anti-Tank Missile. International Conference on Intelligent and Cognitive Systems (1996) 123–128
8. Lin, C. M., Hsu, C. F.: Guidance Law Design by Adaptive Fuzzy Sliding Mode Control. Journal of Guidance, Control and Dynamics **25** (2) (2002) 248–256
9. Nobahari, H., Alasty, A., Pourtakdoust, S. H.: Design of a Supervisory Controller for CLOS Guidance with Lead Angle. AIAA Guidance, Navigation and Control Conference, AIAA-2005-6156 (2005)
10. Palm, R.: Robust Control by Fuzzy Sliding Mode. Automatica **30** (9) (1994) 1429–1437
11. Chen, C. L., Chang M. H.: Optimal Design of Fuzzy Sliding-Mode Control: A Comparative Study. Fuzzy Sets and Systems **93** (1998) 37–48
12. Choi, B. J., Kwak, S. W., Kim, B. K.: Design of a Single-Input Fuzzy Logic Controller and Its Properties. Fuzzy Sets and Systems **106** (3) (1999) 299–308
13. Procyk, T. J., Mamdani, E. H.: A linguistic self-organizing process controller. Automatica, IFAC **15** (1979) 15–30
14. Takagi, T., Sugeno, M.: Fuzzy identification of systems and its application to modeling and control. IEEE Trans. Systems, Man and Cybernetics **15** (1985) 116–132
15. Nomura, H., Hayashi, I., Wakami, N.: A Self Tuning Method of Fuzzy Control by Descent Method. Proceedings of the International Fuzzy Systems Association, IFSA91, Bruxelles (1991) 155–158
16. Siarry, P., Guely, F.: A Genetic Algorithm for Optimizing Takagi-Sugeno Fuzzy Rule Bases. Fuzzy Sets and Systems **99** (1998) 37–47
17. Nobahari, H., Pourtakdoust, S. H.: Optimization of Fuzzy Rule Bases Using Continuous Ant Colony System. Proceeding of the First International Conference on Modeling, Simulation and Applied Optimization, Sharjah, U.A.E., ICMSAO-243 (2005) 1–6
18. Pourtakdoust, S. H., Nobahari, H.: An Extension of Ant Colony System to continuous optimization problems. Lecture Notes in Computer Science **3172** (2004) 294–301
19. Slotine, J. J. E., Li, W.: Applied Nonlinear Control, Prentice-Hall, Upper Saddle River, NJ (1991) chap. 7

On Some Bounds on the Size of Branching Programs (A Survey)[⋆]

Elizaveta A. Okol'nishnikova

Sobolev Institute of Mathematics
of Siberian Branch of Russian Academy of Sciences,
prospect acad. Koptjuga, 4, Novosibirsk 630090, Russia
okoln@math.nsc.ru

Abstract. Previously it was shown by the author that it is possible to reduce obtaining of lower bounds on the complexity of Boolean functions for branching programs without restriction to obtaining of lower bounds on the complexity of minorants of the considered function for branching programs with restriction on the number of occurrences of a variable in a path (read-k-times branching programs). Theorems that reduce the problem of nonlinear lower bounds on the complexity of Boolean functions for branching programs to the problem of lower bounds on the complexity of covering of the set of "ones" of a Boolean function by functions of the defined type or to the problem of obtaining the upper bounds on the number of "ones" of a Boolean function in i-faces of a cube of the defined dimension are presented. A survey of bounds obtained by this method is given.

1 Introduction

The problem of finding of nontrivial lower bounds on the complexity of well-defined Boolean functions is one of important problems in the complexity theory. In the paper the computation of Boolean functions by nondeterministic branching programs is considered. Methods of obtaining exponential lower bounds on the complexity of Boolean functions for read-k-times branching programs and nonlinear lower bounds on the complexity for branching programs without restrictions are presented. These methods permit to obtain $\Omega(n \log / \log\log n)$ lower bound on the complexity of characteristic functions of Reed–Muller codes and BCH codes for nondeterministic branching programs.

The best known lower bound on the complexity of Boolean functions for nondeterministic branching programs is the bound $\Omega(\frac{n^{3/2}}{\log n})$ obtained by P. Pudlák [11] with the use of Nečiporuk's method [5]. Nečiporuk's method is an universal method that is based on the cardinality approach. It can be applied to a number of types of circuits. It gives nontrivial lower bounds on complexity of Boolean

⋆ This research was supported by the Russian foundation for Basic Researches (Grant 03-01-00634) and President program for support of Leading Scientific schools (Grant 313.2003.1).

functions from a special class. Therefore the problem is to suggest methods of obtaining nonlinear lower bounds for other classes of functions. Besides the lower bound $\Omega(n \log\log\log^* n)$ for some symmetric functions (including the Majority function) follows from the bound of A. A. Razborov [12] for the contact–rectifier circuits. The lower bound $\Omega(n\log n/\log\log n)$ for the characteristic function of Reed–Muller codes was obtained by E. A. Okol'nishnikova [8,9]. The similar bound is valid for the characteristic function of BCH codes (Theorem 10).

The best known lower bound on the complexity of Boolean functions for deterministic branching programs is the bound $\Omega(\frac{n^2}{\log^2 n})$ obtained by P. Pudlák [11] with the use of Nečiporuk's method. For some symmetric Boolean functions, in particular for the Majority function, P. Pudlák [10] obtained a lower bound

$$\mathrm{BP}(\mathrm{MAJ}_n) \geq \Omega(n\log\log n/\log\log\log n)$$

for deterministic branching programs. This bound was subsequently improved to the bound $\Omega(n\log n/\log\log n)$ on the complexity of some symmetric Boolean functions, including the Majority function MAJ_n and the elementary symmetric function $E_{\lfloor n/2\rfloor}$ in n variables [1]. E. A. Okol'nishnikova [6] obtained lower bounds on the complexity of characteristic functions of binary codes with a large number of code nodes and growing (with respect to n) code distance for deterministic branching programs. In particular, the lower bound $\Omega(n\log n/\log\log n)$ for characteristic functions of BCH codes with a code distance $\log n/\log\log n$ was obtained.

The bounds for characteristic functions of codes in [6,8] were obtained by the same method. This method reduces obtaining of lower bounds on the complexity of Boolean functions for branching programs without restriction to obtaining of lower bounds on the complexity of minorants of the considered function for branching programs with restriction on the number of occurrences of a variable in a path (read-k-times branching programs).

At present two methods [3,6] of obtaining lower bounds on the complexity of functions for read-k-times branching programs are known. Exponential lower bounds on the complexity of Boolean functions for deterministic read-k-times branching programs ($k(n) = O(\log n/\log\log n)$) were obtained in [6], and for nondeterministic branching programs in [8]. Exponential lower bounds on the complexity of Boolean functions for nondeterministic read-k-times branching programs for $k(n) = O(\log n)$ were obtained in [3]. In [9] methods both from [6,8] and [3,16] were used for obtaining lower bounds on the complexity of minorants of characteristic functions of Reed–Muller codes.

The direct application of the above-mentioned methods does not permit to obtain exponential lower bounds for a lot of functions. The modification of the method from [6] was suggested in [7]. This version of the method can be applied for functions that are defined on the set of variables that corresponds to the edges of graphs, hypergraphs and other multi-dimensional objects. The Boolean function $F_{n,s}$ was introduced in [7], and an exponential lower bound on the complexity of this function for nondeterministic branching programs was obtained.

An extension of ideas of these approaches was made in [2,14,15,16] in order to obtain exponential lower bounds on the size of randomized read-k-times branching programs.

Each function can be computed by read-k-times branching programs with different values of k. In a number of papers it is shown that the complexity of computing of a function by read-1-times branching programs can exponentially (with respect to the number of variables) exceed the complexity of computing of the same function by read-2-times branching programs.

It is shown in [7] that the complexity of computing of $F_{n,s}$ by read-k-times branching programs can exponentially exceed the complexity of computing the $F_{n,s}$ by read-$\lceil k \ln k / \ln 2 + C \rceil$-times branching programs, where C is a constant not depending on k. Later J. S. Thathachar proved (constructively) that the complexity of computing of some functions in n variables by read-k-times branching programs can exponentially (with respect to n) exceed the complexity of computing of the same functions by read-$(k+1)$-times branching programs.

Exponential lower bounds for schemes with different restrictions were obtained in a lot of papers. But the method from [6,8] is the first one that allow us to use lower bounds for schemes with restrictions for obtaining of nontrivial lower bounds for schemes without restrictions.

The problem of nonlinear lower bounds on the complexity of Boolean functions for branching programs is reduced to the problem of lower bounds on the complexity of covering of the set of "ones" of a Boolean function by functions of the defined type or to the problem of obtaining the upper bounds on the number of "ones" of a Boolean function in i-faces of a cube of the defined dimension (Theorems 4 and 5).

We use traditional definitions of nondeterministic and deterministic branching programs (see, e.g. [13]). A nondeterministic branching program is said to be read-k-times branching program if and only if for any i, $1 \le i \le n$, edges with x_i-labels occur at most k times in any computation path (whether or not consistent). By NBP($\mathcal{P}$) (BP($\mathcal{P}$)) denote the size of a nondeterministic (deterministic) branching program $\mathcal{P}$. By NBP(f) (BP(f)) denote the size of the minimal nondeterministic (deterministic) branching program that computes a Boolean function f.

By NBPk(f) (BPk(f)) denote the size of the minimal nondeterministic (deterministic) read-k-times program that computes a Boolean function f.

2 Lower Bounds on the Size of Read-k-Times Branching Programs

Note that methods of obtaining high lower bounds on the complexity of Boolean functions for read-k-times branching programs in [3,6,7,16] are similar. Let $\mathcal{P}$ be a branching program computing the Boolean function f in n variables. To each "one" of a Boolean function f we can assign a path in $\mathcal{P}$. This path is divided into "equal" parts. A separating set for these parts is a subset of nodes [6,7] or a subset of edges [3,16] of the branching program. The cardinality of this set depends only on parameters selected beforehand and is considerably less than the path length. The function f_i is assigned to the chosen subset of nodes or edges of the program $\mathcal{P}$. This function depends only on this subset of nodes or edges and does not depend on paths to which these subsets belong. Thus

$$f = \vee f_i, \tag{1}$$

i. e. the functions f_i cover the set of "ones" of the function f. If the number of "ones" of each function f_i is not large, and the number of "ones" of f is large, then the number of different subsets that corresponds to units of the Boolean function is large. It allows us to obtain lower bound on the number of nodes (or edges) of the branching program.

In [6] subsets of nodes of the branching program correspond to "ones" of the function. We need to transform a branching program to uniform form in this case. Recall (see [6,8]) that nondeterministic read-k-times branching program is said to be *uniform* if for every node a of this program and for every i, $1 \le i \le n$, the number of edges with x_i-labels, in each path from the source to a node a depends only on i and a. Moreover, for every i, $1 \le i \le n$, the number of edges labelled with $x_i = d$, in each path from the source to the exit is equal to k.

This transformation slightly increases the size of a program, but it makes possible to consider the generalized parts of the path. It gives the possibility to assign to a path not all nodes that were chosen as a separating set of the path, but only a part of them. Sometimes in this a way it is possible to receive bounds that are better than bounds obtained by the application of a method from [3], especially when read-k-times branching programs are used for obtaining of lower bounds on the complexity for branching programs without restrictions (see Corollary 1).

In [3] subsets of edges of the branching program correspond to units of the function. On the one hand, there is no need to transform a branching program to a uniform form in this case, and on the other hand it does not allow to combine parts of the path, i. e., it is necessary to assign to a path all edges that are the separators of the parts. In order to obtain lower bounds on the complexity for branching programs by this method it is necessary to extract the root of greater degree than in [6] from the cardinality of the obtained covering of set of "ones" of the function from (1).

Let's consider all possible representations of the function $f(Y)$, $|Y| = n$, of the form

$$f(Y) = \bigvee_{j=1}^{A} f_1^j(Y_1^j \cup Y_0^j) \wedge f_2^j(Y_2^j \cup Y_0^j), \tag{2}$$

where Y_1^j, Y_2^j, and Y_0^j are nonintersecting sets; $Y = Y_1^j \cup Y_2^j \cup Y_0^j$; $|Y| = n$; $|Y_1^j| \ge m_1$; $|Y_2^j| \ge m_2$; $|Y_0^j| = n - |Y_1^j| - |Y_2^j|$.

By $A(f; n, m_1, m_2)$ denote the minimal number of disjunctive terms in the representation (2).

I. We formulate the result from [8] for a particular case when $p = k$ and $t = k^2 + k$ (see lemma 3(1) from [8]). We have

Theorem 1 ([8], Theorem 3). *Let f be a Boolean function essentially depending on n variables, $n \geq 16$. Then*

$$\mathrm{NBP}k(f) \geq \max\left\{n; \frac{1}{8\sqrt{k}} \cdot (A(f; n, m_1, m_2))^{1/(4k)}\right\},$$

where $m_1 = \left\lceil n \Big/ \left(2(ke)^k\right)\right\rceil$, $m_2 = \lceil n/(k+1)\rceil$.

II. In the same terms results from [3,16] can be formulate in the following form.

Theorem 2. *Let f be a Boolean function essentially depending on n variables. Then*

$$\mathrm{NBP}k(f) \geq \max\left\{n;\ \frac{1}{2} \cdot (A(f; n, m_1, m_2))^{1/(144k^2 \cdot 2^k)}\right\},$$

where $m_1 = \lceil (2/3)n/2^k \rceil$, $m_2 = \lceil (2/3)n/2^k \rceil$.

3 Reduction of Lower Bounds on the Complexity for Programs Without Restrictions to Lower Bounds on the Complexity for Read-k-Times Programs

The idea of a method of obtaining of nonlinear lower bounds for branching programs without restrictions is the same as in [6]. Let $\mathcal{P}$ be a branching program that computes a Boolean function $f(x_1, x_2, \ldots, x_n)$. Let the number of occurrences of a variable x_i on a path from the input node to the output node exceeds $k(n)$, where $k(n) \to \infty$ as $n \to \infty$, and the number of such variables is not small. Then the size of $\mathcal{P}$ can not be small too. If the number of such variables is small, we substitute constants for these variables. It allows us to transform the program $\mathcal{P}$ to a program $\mathcal{P}'$ with restrictions on the number of occurrences of a variable in a path, i. e. to consider the computation of a minorant of the function f by read-$k(n)$-times branching programs.

This approach allows us to reduce obtaining of lower bounds on the complexity of a Boolean function for branching programs without restrictions to obtaining of lower bounds on the complexity of minorant functions of the considered Boolean function for programs with restrictions, namely, for read-$k(n)$-times branching programs.

Let $f(x_1, x_2, \ldots, x_n)$ be a Boolean function, $X' = \{x_{i_1}, \ldots, x_{i_m}\}$ be a subset of the set $\{x_1, x_2, \ldots, x_n\}$, and $\alpha = \{\alpha_{i_1}, \ldots, \alpha_{i_m}\}$ be a set of constants. By $f\big|_{X'=\alpha}$ denote the function that is obtained from f by substitution of constants from α instead of variables from X'.

Theorem 3 ([8], Theorem 1). *Let $g(X)$ be a Boolean function, C, $0 < C < 1$, be a constant, $\psi(n)$ be a growing function. Let X_0 be a subset of variables, i. e., $X_0 \subseteq X$, and let $|X_0| = \lfloor Cn \rfloor$. If for each X_0 there exists a substitution of constants α in the set X_0 such that* $\mathrm{NBP}k(n)\left(g\big|_{X_0=\alpha}(X \setminus X_0)\right) \geq n\psi(n)$, *then* $\mathrm{NBP}(g) \geq \min\{Cnk(n), n\psi(n)\}$.

4 Lower Bounds on the Complexity for Branching Programs Without Restrictions

From Theorems 1, 2, and 3 it is easy to formulate theorems that reduce the problem of nonlinear lower bounds on the complexity of Boolean functions for branching programs to estimating of the value $A(f; n, m_1, m_2)$, i.e. to the problem of lower bounds on the complexity of covering of the set of "ones" of a Boolean function by functions of the defined type.

There are some ways to obtain lower bounds on $A(f; n, m_1, m_2)$. We use the following one. (The same way was used in [8]).

By $H_i(f)$ denote the maximal number of "ones" of a Boolean function f belonging to i-faces of a cube.

It is easy to prove the following lemma.

Lemma 1. *The value $A(f; n, m_1, m_2)$ satisfies an inequality*

$$A(f; n, m_1, m_2) \geq \frac{|f^{-1}(1)|}{2^{n-m_1-m_2} H_{m_1}(f) H_{m_2}(f)}.$$

Combining Lemma 1, Theorem 1, 2, and 3 we obtain the following theorems.

Theorem 4 ([9], Theorem 6). *Let $g_n(X_n)$, $|X_n| = n$, be a sequence of Boolean functions; $k(n)$ be a growing function; C, $0 < C < 1$, be a constant. Then the complexity* $\mathrm{NBP}(g_n)$ *of the function $g_n(X_n)$ for nondeterministic branching programs without restrictions satisfies an inequality*

$$\mathrm{NBP}(g_n) \geq \min \left\{ Cnk(n), \frac{1}{8\sqrt{k}} \cdot \left(\frac{|g_n^{-1}(1)|}{2^{n-m_1-m_2} H_{m_1}(g_n) H_{m_2}(g_n)} \right)^{1/(4k)} \right\},$$

where $m_1 = \left\lceil \frac{\lceil (1-C)n \rceil}{2(ke)^k} \right\rceil$, $m_2 = \left\lceil \frac{\lceil (1-C)n \rceil}{k+1} \right\rceil$.

Theorem 5 ([9], Theorem 7). *Let $g_n(X_n)$, $|X_n| = n$, be a sequence of Boolean functions; $k(n)$ be a growing function; C, $0 < C < 1$, be a constant. Then the complexity* $\mathrm{NBP}(g_n)$ *of the function $g_n(X_n)$ for nondeterministic branching programs without restrictions satisfies an inequality*

$$\mathrm{NBP}(g_n) \geq \min \left\{ Cnk(n), \frac{1}{2} \cdot \left(\frac{|g_n^{-1}(1)|}{2^{n-m_1-m_2} H_{m_1}(g_n) H_{m_2}(g_n)} \right)^{1/(144k^2 \cdot 2^k)} \right\},$$

where $m_1 = m_2 = \left\lceil (2/3) \frac{\lceil (1-C)n \rceil}{2^k} \right\rceil$.

5 Lower Bounds on the Complexity of Characteristic Functions of Binary Codes for Nondeterministic Branching Programs

We use the results from [17] for generalized Hamming weights and the weight hierarchy for linear codes to estimate the maximum number of the code nodes belonging to i-faces, i.e. to obtain the upper bound on H_i for the characteristic function of Reed–Muller codes. For these codes we shall use the notation from [4,17]. By $\mathcal{R}(u, m)$ denote the mth Reed–Muller code of order u. $\mathcal{R}(u, m)$ is considered as a linear space composing of all Boolean polynomials of degree u or less in m variables $v_1, v_2, \ldots, v_m$.

It is known that the number of code nodes in $\mathcal{R}(u, m)$ is equal to

$$2^{1+\binom{m}{1}+\binom{m}{2}+\ldots+\binom{m}{u}}; \tag{3}$$

the length of code words is equal to

$$n = 2^m; \tag{4}$$

the minimal code distance is

$$d = 2^{m-u}.$$

Theorem 6 ([8], Theorem 7). *Let φ be a natural number not exceeding m. Then a $2^m/2^\varphi$-face of the cube contains no more than $2^{\binom{m-\varphi}{0}+\binom{m-\varphi}{1}+\ldots+\binom{m-\varphi}{u-\varphi}}$ code nodes of the code $\mathcal{R}(u, m)$.*

From this theorem and Theorems 4 and 5 we obtain the following bounds on the complexity of characteristic functions of Reed–Muller codes.

Theorem 7 ([8], Theorem 8). *Let $\frac{m}{m-u_m} \to \infty$ as $m \to \infty$ and $m - u \geq 3$. Then*

$$\mathrm{NBP}(\mathcal{R}(\mathrm{u_m}, \mathrm{m})) = \Omega\left(2^m \frac{m}{m-u_m} \Big/ \ln \frac{m}{m-u_m}\right).$$

The proof of the following theorem is similar to that of Theorem 7.

Theorem 8 ([9], Theorem 10). *Let $\frac{m}{m-u_m} \to \infty$ as $m \to \infty$ and $m - u \geq 3$. Then*

$$\mathrm{NBP}(\mathcal{R}(u_m, m)) \geq 2^{m-1} \min\left\{\frac{m}{8(m-u_m)}, \frac{m-u_m}{4} \log_2 \frac{m}{m-u_m}\right\}.$$

Combining Theorems 7 and 8 we have

Theorem 9 ([9], Theorem 11). *Let $\frac{m}{m-u_m} \to \infty$ as $m \to \infty$ and $m - u_m \geq 3$. Then*

$$\mathrm{NBP}(\mathcal{R}(u_m, m)) \geq \frac{2^m}{2} \max\left\{\frac{m}{4(m-u_m)} \Big/ \log_2 \frac{m}{m-u_m},\right.$$
$$\left.\min\left\{\frac{m}{8(m-u_m)}, \frac{m-u_m}{4} \log_2 \frac{m}{m-u_m}\right\}\right\}.$$

Corollary 1. *Let* $\frac{m}{m-u_m} \to \infty$ *as* $m \to \infty$ *and* $m - u_m \geq 3$. *Then*

$$\mathrm{NBP}(\mathcal{R}(u_m, m)) \succeq \begin{cases} \frac{2^m m}{(m-u_m)} \Big/ \log_2 \frac{m}{m-u_m}, & \text{if } m - u_m \leq h \\ & \text{(by Theorem 7)}; \\ 2^m \sqrt{m}, & \text{if } h \leq m - u_m \leq \frac{\sqrt{m}}{\log_2 \sqrt{m}} \\ & \text{(by Theorem 8)}; \\ \frac{2^m (m-u_m)}{8} \log_2 \frac{m}{m-u_m}, & \text{if } \frac{\sqrt{m}}{\log_2 \sqrt{m}} \leq m - u_m \leq h_1 \\ & \text{(by Theorem 8)}; \\ 2^m \sqrt{m \log_2 m}, & \text{if } h_1 \leq m - u \leq \frac{\sqrt{m}}{\sqrt{\log_2 (m)}} \\ & \text{(by Theorem 8)}; \\ \frac{2^m m}{16(m-u_m)}, & \text{if } \frac{\sqrt{m}}{\sqrt{\log_2 m}} \leq m - u_m \\ & \text{(by Theorem 8)}. \end{cases}$$

$$\text{where } h = \frac{2\sqrt{m}\left(1 - \frac{2\log_2 \log_2 \sqrt{m}}{\log_2 m}\right)}{\log_2 m},$$

$$h_1 = \frac{\sqrt{m}\left(1 - \frac{2\log_2 \log_2 m}{\log_2 m}\right)}{\sqrt{\log_2 (m)}}.$$

From this corollary it follows that there exist Boolean functions such that lower bounds of the complexity of these functions obtained by the use of Theorem 7 are better than similar ones obtained by the use of Theorem 8, and vise versa.

Reed–Muller code is a linear code, and the number of checking symbols of this code is equal to $2^{1+\binom{m}{1}+\binom{m}{2}+\ldots+\binom{m}{m-u-1}}$. The complexity of a linear function (in n variable) for deterministic branching program is not more than $2n$. Therefore

$$\mathrm{NBP}(\mathcal{R}(u_m, m)) \leq 2^{m+1+1+\binom{m}{1}+\binom{m}{2}+\ldots+\binom{m}{m-u-1}}.$$

From this fact and Theorem 9 we have

Corollary 2. *Let* $u_m = m - C^0$, *where* $C^0, C^0 \geq 3$, *is a constant, then*

$$n \log n / \log\log n \preceq \mathrm{NBP}(\mathcal{R}(u_m, m)) \preceq n \log^{C_0 - 1} n,$$

where n *is a number of variables of the characteristic function of* $\mathcal{R}(u_m, m)$.

There is a slight difference between statement of Theorem 1 for nondeterministic and deterministic branching programs. Bounds

$$n \ln n / \ln\ln n \preceq \mathrm{BP}(H_r) \preceq n \ln^2 n / \ln\ln n,$$

on the complexity of the characteristic function of BCH code H_r ($r = \ln n / \ln\ln n$) were obtained for deterministic branching programs (Corollary 3 from [6]). It is easy to obtain similar bounds for nondeterministic branching programs.

Theorem 10. *Let B_r be a characteristic function of BCH code, where $r = \ln n / \ln\ln n$, then*

$$n \ln n / \ln\ln n \preceq \mathrm{NBP}(H_r) \preceq n \ln^2 n / \ln\ln n.$$

References

1. Babai L., Pudlák P., Rödl V., Szemerédi M. Lower bounds to the complexity of symmetric Boolean functions. *Theoretical Computer Science*, 74 (1990) 313–324.
2. Bollig B., Sauerhoff M., Wegener I. On the nonappoximability on boolean functions by OBDD and read-k-times branching programs. *Information and Computation*, 178 (2002) 263–278.
3. Borodin A., Razborov A., Smolensky R. On lower bounds for read-k-times branching programs. *Computational Complexity*, 3(1) (1993) 1–18.
4. MacWilliams F.J., Sloane N.J.F. The theory of Error-Correcting Codes. Amsterdam: North-Holland, (1977).
5. Nečiporuk E. On a Boolean function. *Soviet Math. Doklady*, 7 (1966) 999-1000.
6. Okol′nishnikova E. A. Lower bounds on complexity for the realization of characteristic functions of binary codes by binary programs. *Metody Diskret. Anal.*, 51 (1991) 61–83 (in Russian) (see also: *Siberian Adv. Math.,* 3(1) (1993) 152–166).
7. Okol'nishnikova E. A. On the hierarchy of nondeterministic branching k-programs. In *Proc. of FCT 97, Lecture Notes in Comput. Sci.*, 1279 (1997) 376–387.
8. Okol'nishnikova E. A. On one method of obtaining of lower bounds of Boolean functions for nondeterministic branching programs. *Diskretn. Anal. Issled. Oper. Ser. 1*, 8(4) (2001) 76–102 (in Russian). (see also: ECCC TR02-020,2002, available at http://www.eccc.uni-tri.de/eccc/ (in English))
9. Okol'nishnikova E. A. On the complexity of nondeterministic branching programs for characteristic functions of Reed–Muller codes. *Diskretn. Anal. Issled. Oper. Ser. 1*, 10(3) (2003) 67–81 (in Russian).
10. Pudlák P. A lower bound on complexity of branching programs. *Lecture Notes in Comput. Sci.*, 176 (1984) 480–489.
11. Pudlák P. The hierarchy of Boolean circuits. *Comput. Artificial Intelligence*, 6(5) (1987) 449–468.
12. Razborov A. A. Lower bounds on the complexity of symmetric Boolean functions by switching-rectifier circuits. *Matemat. zametki*, 48(6) (1990) 79-90.
13. Razborov A. A. Lower bounds for deterministic and nondeterministic branching program. *Lecture Notes in Comput. Sci.*, 529 (1991) 47–61.
14. Sauerhoff M. Randomness versus nondeterminism for read-once and read-k branching programs. *Lecture Notes in Comput. Sci.*, 1373 (1998) 105–115.
15. Sauerhoff M. Randomness versus nondeterminism for read-once and read-k branching programs. *Lecture Notes in Comput. Sci.*, 2607 (2003) 307–318.
16. Thathachar J. S. On separating the read-k-times program hierarchy. In *Proc. of the 30th annual ACM Symposium on theory of computing*, (1998) 652–662.
17. Wei V.K. Generalized Hamming weights for linear codes. *IEEE Trans. on Inform. Theory*, 37(5) (1991) 1412–1418.

Two Metaheuristics for Multiobjective Stochastic Combinatorial Optimization

Walter J. Gutjahr

Dept. of Statistics and Decision Support Systems,
University of Vienna

Abstract. Two general-purpose metaheuristic algorithms for solving multiobjective stochastic combinatorial optimization problems are introduced: SP-ACO (based on the Ant Colony Optimization paradigm) which combines the previously developed algorithms S-ACO and P-ACO, and SPSA, which extends Pareto Simulated Annealing to the stochastic case. Both approaches are tested on random instances of a TSP with time windows and stochastic service times.

Keywords: Ant colony optimization, combinatorial optimization, multiobjective decision analysis, simulated annealing, stochastic optimization.

1 Introduction

Recently, two branches of combinatorial optimization have undergone a particularly dynamic evolution due to a strong application pull on the one hand, a technology push triggered by increased computer power on the other hand. The one of these branches is *multiobjective* combinatorial optimization (MOCO) pursuing the aim to support decision makers in the choice among a finite, but large number of alternatives without imposing the necessity of an *a priori* assignment of weights to different objectives (see, e.g., Ehrgott and Gandibleux [6]). Especially the solution of MOCO problems by *metaheuristics* has recently attracted the attention of many researchers [8]. The other of the mentioned branches is *stochastic* combinatorial optimization (SCO), which promises to support decision making under a type of uncertainty that can be represented by some suitable stochastic model. In many cases, simulation is used as an auxiliary tool for solving SCO problems. Efficient methods of diverse kind have been developed in the past for SCO and simulation-supported optimization (for a recent survey, see Fu [7]). Also in this area, metaheuristic techniques are gaining importance.

Interestingly enough, despite the large body of literature in both the MOCO and the SCO area, there are only few papers combining both features, although it would be desirable to be able to cope with problems that incorporate both multiple objectives *and* uncertainty. The scarcity of literature in this intersection has also been noted in [7] and in [1]. *Very* few articles deal with problems of this combined type by *metaheuristic* techniques. Baesler and Sepulveda [1] report on a multiobjective simulation-optimization problem in layout and scheduling, which they solve by a modification of a genetic algorithm (GA); their approach relies on goal programming. Also Hughes [11] uses a GA approach, but he refers to the paradigm of finding nondominated (Pareto-optimal) solutions. He adapts ranking procedures applied in standard multiobjective GAs to the situation where function evaluations are subject to random noise.

The aim of this paper is to present two general-purpose metaheuristic solution algorithms SP-ACO and SPSA, determining approximations to the Pareto-optimal set for instances from a large class of MOSCO (multiobjective stochastic combinatorial optimization) problems. In Section 2, the considered type of MOSCO problems is defined. Section 3 presents the new algorithms SP-ACO and SPSA. The first, SP-ACO, is based on the ant colony optimization (ACO) paradigm (see [5]), whereas the second, SPSA, an extension of the PSA algorithm by Czyzak and Jaszkiewicz [2], draws on the well-known simulated annealing metaheuristic. In Section 4, we outline the application of both approaches to randomly generated instances of a bi-objective stochastic travelling salesperson problem with time windows and stochastic service times, and report on the obtained experimental results. Section 5 contains conclusions.

2 MOSCO Problem Formulation and Objective Function Estimation

Both approaches are designed for the heuristic solution of MOSCO problems of the following very general form:

$$\text{Minimize}\,(F_1(x),\ldots,F_R(x)) \quad \text{subject to} \quad x \in S \tag{1}$$

with $F_r(x) = \mathrm{E}\,(f_r(x,\omega))$ $(r = 1,\ldots,R)$. Therein, x is the decision variable, f_r is the r-th cost function, R is the number of objectives (cost functions), ω denotes the influence of randomness, E denotes the mathematical expectation, and S is a finite set of feasible decisions.

A solution $x' \in S$ *dominates* a solution $x \in S$, if $F_r(x') \leq F_r(x)$ for all $r = 1,\ldots,R$, and if there is at least an index r such that $F_r(x') < F_r(x)$. A solution $x \in S$ is called dominated resp. nondominated by a set $S' \subseteq S$ of solutions, if there is an $x' \in S'$ such that x' dominates x, resp. if there is no such $x' \in S'$. A solution x is called *Pareto-optimal* if it is nondominated by S. The *Pareto-optimal set* is the set of Pareto-optimal solutions. As an *exact* solution of (1), we consider the Pareto-optimal set defined by the MOCO problem (1). Since the proposed algorithms are heuristics, it cannot be expected that they will in general produce the Pareto-optimal set. They output *approximations* to this set. Concerning quality evaluation of these approximations, we refer the reader to Section 4.

In our context, it is not necessary that $\mathrm{E}\,(f_r(x,\omega))$ can be computed numerically. Instead, *sampling* is used for estimating this quantity: For this purpose, draw N random *scenarios* $\omega_1,\ldots,\omega_N$ independently from each other. A *sample estimate* of $F_r(x) = \mathrm{E}\,(f_r(x,\omega))$ is given by

$$\mathcal{E}F_r(x) \;=\; \frac{1}{N}\sum_{\nu=1}^{N} f_r(x,\omega_\nu) \;\approx \mathrm{E}\,(f_r(x,\omega)). \tag{2}$$

3 Algorithms

3.1 The SP-ACO Algorithm

There exits several articles extending the ACO metaheuristic to multiobjective *or* to stochastic problems; see Dorigo and Stützle [5] for a survey. For our combined approach, we rely on the following formerly developed basic techniques: In [9], [10], an algorithm S-ACO for the heuristic solution of *single-objective* stochastic combinatorial problems has been proposed. The algorithm SP-ACO (Stochastic Pareto Ant Colony Optimization) presented here is an extension of S-ACO to the multiobjective case, combining it with the P-ACO algorithm developed for MOCO problems in [3], [4]. S-ACO and SP-ACO work based on the encoding of a given problem instance as a *construction graph* $\mathcal{C}$, a directed graph with a distinguished start node. The stepwise construction of a solution is represented by a self-avoiding random walk in $\mathcal{C}$, beginning in the start node. There may be additional rules defining particular nodes as infeasible after a certain partial walk has been traversed. When there is no feasible unvisited successor node anymore, the walk stops and is decoded as a complete solution for the problem. The conceptual unit performing such a walk is called an *ant.*

The encoding must assign exactly one feasible solution to each feasible walk. Vice versa, to each feasible solution *at least* one feasible walk (possibly more) must correspond. Given that the indicated condition is satisfied, we may consider a walk as a solution, denote it again by the symbol x and consider S as the set of feasible walks.

The probability p_{kl} that an ant goes from a node k to a feasible successor node l is chosen as proportional to $\tau_{kl} \cdot \eta_{kl}(u)$, where τ_{kl} is the so-called *pheromone value*, a memory value storing how good step (k, l) has been in previous runs, and $\eta_{kl}(u)$ is the so-called *visibility*, a pre-evaluation of how good step (k, l) will presumably be, based on some problem-specific heuristic. $\eta_{kl}(u)$ is allowed to depend on the partial walk u performed so far. In the experimental investigations in this paper, we did not use nontrivial visibility values, setting $\eta_{kl}(u) = 1$ in each case. For this reason, the role of the visibility (which can improve solution quality) will not be discussed here. For details, we refer the reader to [5].

Whether a continuation (k, l) of a partial walk u ending with node k is feasible or not is defined in accordance with the condition above that node l is not yet contained in u, and that none of the (eventual) additional rules specifies l as infeasible after u has been traversed.

In a loop, a predefined number Γ of random walks of ants according to the procedure above are performed sequentially. These Γ walks form together a *round* of the process. The single-objective heuristic S-ACO determines in each round a *round-winner.* This is done by comparing all walks that have been performed in this round on one random scenario ω, drawn specifically for this round. In the multiobjective context of SP-ACO, the determination of a round winner necessitates that a *unique* objective function for ranking the solutions produced by the walks of the ants is defined for this round. We do this by taking a *weighted average* of the cost functions $f_1, \ldots, f_R$. The weights $w_1, \ldots, w_R$ are drawn ran-

```
Procedure SP-ACO
  $\tau_{kl}^{(r)} := 1$ for all $(k,l)$ and for all $r = 1,\dots,R$;
  initialize the solution set $X$ as the empty set;
  for period $\pi = 1$ to $\Pi$ {
    draw weights $w_1,\dots,w_R$ randomly;
    $\tau := \sum_{r=1}^{R} w_r \tau^{(r)}$;
    for round $m = 1$ to $M$ {
      for ant $\gamma = 1,\dots,\Gamma$ {
        set position $k$ equal to start node of $\mathcal{C}$;
        set $u$ equal to the empty list;
        while (a feasible continuation $(k,l)$ of $u$ exists) {
          select successor node $l$ with probability
            $p_{kl} = \begin{cases} 0, \text{ if } (k,l) \text{ is infeasible}, \\ \tau_{kl} \cdot \eta_{kl}(u) \;/\; \left(\sum_{(k,r)} \tau_{kr} \cdot \eta_{kr}(u)\right), \text{ else}, \end{cases}$
          the sum being over all feasible $(k,r)$;
          set $k := l$, and append $l$ to $u$; }
        $x_\gamma := u$; }
      based on one random scenario $\omega$ and objective function
        $f(x,\omega) = \sum_{r=1}^{R} w_r f_r(x,\omega)$,
      select the best walk $x$ out of $x_1,\dots,x_\Gamma$;
      if $(m = 1)$ set $\hat{x} := x$; // candidate for best solution in period
      else {
        based on random scenarios $\omega_1,\dots,\omega_{N_m}$, compute sample estimate
          $\mathcal{E}(F(x) - F(\hat{x})) = \frac{1}{N_m} \sum_{\nu=1}^{N_m} \sum_{r=1}^{R} w_r (f_r(x,\omega_\nu) - f_r(\hat{x},\omega_\nu))$;
        if $(\mathcal{E}(F(x) - F(\hat{x})) < 0)$ set $\hat{x} := x$; }
      evaporation: $\tau^{(r)} := (1-\rho)\,\tau^{(r)}$ for all $r$;
      global-best reinforcement: $\tau_{kl}^{(r)} := \tau_{kl}^{(r)} + c_1 w_r$ for all $(k,l) \in \hat{x}$ and all $r$;
      round-best reinforcement: $\tau_{kl}^{(r)} := \tau_{kl}^{(r)} + c_2 w_r$ for all $(k,l) \in x$ and all $r$;
      $\tau := \sum_{r=1}^{R} w_r \tau^{(r)}$;
      based on a sample of size $N^{(c)}$, evaluate estimates $\mathcal{E}F_1(\hat{x}),\dots,\mathcal{E}F_r(\hat{x})$;
      if ($\hat{x}$ nondominated by $X$ according to computed estimates of size $N^{(c)}$)
        add $\hat{x}$ to $X$ and remove dominated elements from $X$; } }
```

Fig. 1. Pseudocode SP-ACO

domly at the beginning of a process phase called *period*. A period contains several rounds in which solutions are gradually improved w.r.t. the current weights. In the next period, new weights are drawn; this process is iterated.

In an ACO implementation for a deterministic problem, it is customary to store the best solution seen so far in a special variable. A crucial difference to the deterministic case is that in the stochastic context, it is not possible anymore to decide with certainty whether a current solution x is better than the solution currently considered as the best found, $\hat{x}$, or not. To make a tentative decision by sampling, we perform a *tournament*: After a current round-winner x has been determined, x is compared with the solution considered as the overall best solution so far in this period, $\hat{x}$. For evaluating the solutions, a weighted average F of the objective functions $F_1,\dots,F_R$ with the current weights $w_1,\dots,w_R$ is used, and estimates for the values of the functions F_r are determined from a sample consisting of N_m randomly drawn scenarios ω_ν which are used by both solutions. Also these scenarios are round-specific, i.e., in the next round, new scenarios will be drawn. The larger N_m, the more reliable is the decision. The winner of the comparison is stored as the new "global-best" $\hat{x}$. In [9] it is shown that the sample size N_m should be increased as a linear function of the round number m to enable a convergence result for the single-objective case.

Next, the solution $\hat{x}$ considered so far as the best of the current period as well as the current round-winner are reinforced on each of their arcs by pheromone increments, after a certain fraction ρ ("evaporation factor") of pheromone has been removed from each arc. The parameters $c_1 > 0$ and $c_2 > 0$ in the algorithm determine the amount of pheromone increment on global-best and round-best walks, respectively.

We take account of the different objective functions $F_1, \ldots, F_R$ by assigning a *separate* pheromone matrix $\tau^{(r)} = (\tau_{kl}^{(r)})$ to each objective r. Global-best and round-best reinforcement is done in each of these pheromone matrices with the current respective weight w_r. For the transition from a node k to a feasible successor node l, the guiding pheromone values τ_{kl} must be computed as a weighted mean of the objective-specific pheromone values $\tau_{kl}^{(r)}$. As weights we take again the current values w_r.

After reinforcement, it is checked whether the current global winner solution $\hat{x}$ can be added to the current set X of candidates for the approximation to the Pareto-optimal set. For this purpose, an estimation of the objective function values $F_1(\hat{x}), \ldots, F_R(\hat{x})$ based on a random sample of constant size $N^{(c)}$, where $N^{(c)}$ is comparably large, is performed. If $\hat{x}$ turns out as nondominated by X according to these estimates, $\hat{x}$ is added to X, and solutions in X dominated by $\hat{x}$ are removed from X. The objective function estimates obtained from the sample of size $N^{(c)}$ are assigned to $\hat{x}$ in the list of the elements of X for future dominance comparisons with new candidates for the solution set X.

3.2 The SPSA Algorithm

In this subsection, we present an extension of the PSA (Pareto Simulated Annealing) algorithm by Czyzak and Jaszkiewicz [2], designed for solving MOCO problems, to an algorithm SPSA (Stochastic Pareto Simulated Annealing) for the solution of MOSCO problems. Since PSA is already a well-established technique, we keep the description short, focusing on the points by which SPSA extends PSA. The pseudocode of SPSA is given in Fig. 2.

PSA uses a search set (here denoted by Θ) exploring the solution space governed by a mechanism that (i) drives the search points towards the Pareto-optimal set, and (ii) favors diversification by forcing points that lie close to each other in the solution space to "specialize" on different objectives. The last effect is achieved by a suitable modification of the weights assigned to the objective functions: the weights are increased for those objectives for which a current search point x is better than a near-by search point x', and decreased for the others.

```
Procedure SPSA
  initialize the search set $\Theta$ by $s$ random feasible solutions;
  initialize the solution set $X$ as the empty set;
  initialize $N^{(2)}$ by $N_{init}$;
  for $i = 1$ to $s$
    if ($i$th solution $x_i$ in $\Theta$ is nondominated by $X$, based on sample size $N^{(1)}$)
      add $x_i$ to $X$ and remove dominated elements from $X$;
  initialize temperature parameter $T$;
  repeat until (termination criterion is met) {
    for $l = 1$ to $L$ {
      for $i = 1$ to $s$ {
        for $r = 1$ to $R$
          compute sample estimate $\mathcal{E}F_r(x_i)$ based on sample size $N^{(3)}$;
        construct a random feasible neighbor solution $y$ to $x_i$;
        select $x' \in \Theta$ nondominated by $x_i$ with minimum distance to $x_i$;
        if (first run or $x'$ not found) {
          for $r = 1$ to $R$
            draw random weight $w_{ir}$;
          normalize weights $w_{ir}$ to $\sum_r w_{ir} = 1$; }
        else {
          for $r = 1$ to $R$ {
            compute sample estimate $\mathcal{E}F_r(x')$ based on sample size $N^{(3)}$;
            if ($\mathcal{E}F_r(x_i) < \mathcal{E}F_r(x')$) $w_{ir} := aw_{ir}$; else $w_{ir} := w_{ir}/a$;
            normalize weights $w_{ir}$ to $\sum_r w_{ir} = 1$; } }
        for $r = 1$ to $R$
          compute sample estimate $\mathcal{E}(F_r(x_i) - F_r(y))$ based on sample size $N^{(2)}$;
          with probability $\min(1, \exp(\sum_r w_{ir}\mathcal{E}(F_r(x_i) - F_r(y))/T)$ {
            $x_i := y$;
            if ($T < T_c$ and $y$ nondominated by $X$, based on sample of size $N^{(1)}$)
              add $y$ to $X$ and remove dominated elements from $X$; } } }
    $T := bT$;
    increase $N^{(2)}$ by $N_{inc}$; }
```

Fig. 2. Pseudocode SPSA

Basically, our extension SPSA works as PSA, with the exception that at any time when an objective function evaluation is necessary, an estimation based on sampling is done. For the sample estimate, we use the notation of (2). Three sample size parameters $N^{(1)}$, $N^{(2)}$ and $N^{(3)}$ are used. $N^{(1)}$ and $N^{(3)}$ are constants, $N^{(2)}$ is a variable. $N^{(1)}$ corresponds to the sample size $N^{(c)}$ in the SP-ACO algorithm in that it is applied for deciding whether a candidate solution is (presumably) nondominated by the current elements of the solution set X. As $N^{(c)}$ in the case of SP-ACO, $N^{(1)}$ must be high, because the estimates are not revised anymore at a later time. In the experiments, it turned out that better results were achieved by considering insertion into X only after the temperature parameter T has fallen below some threshold T_c. Sample size $N^{(2)}$ is used for deciding whether or not a neighbor solution is to be accepted. Since by the simulated annealing philosophy, in a late phase (low temparature), neighbor solutions that are not better than the current solution should be rejected with a high probability, it is important that the estimation accuracy for the difference of the objective function values is gradually increased during the process, similarly as we gradually increase N_m in the SP-ACO algorithm. The third sample size, $N^{(3)}$, is used for getting estimates of the objective function values of a current solution $x \in \Theta$ compared to those of the closest neighbor x' of x in Θ.

4 Experimental Results on a TSPTW-SST

For first computational experiments with the described algorithms, we used a bi-objective TSP with Time Windows and Stochastic Service Times (TSPTW-SST). A set of customers $\{1, \ldots, n\}$ and a distance matrix $D = (d_{ij})$ are given. Distances are interpreted as driving times. Let us imagine that the travelling person is a service engineer. To each customer i, a time window $[a_i, b_i]$ can be assigned, indicating that customer i requests a visit by the service engineer starting at time t_i with $a_i \leq t_i \leq b_i$. If the service engineer arrives at customer i at a time t_i before time a_i, (s)he must wait until time a_i. If (s)he arrives after time b_i, a *tardiness* of amount $t_i - b_i$ is registered. Not every customer needs to have a time window for the visit. The service at customer i takes some time Y_i, where Y_i is a random variable with known distribution. After finishing the service at customer i, the service engineer drives to the next customer on the list given by the chosen permutation x of customers. The aim is to minimize the exepected values of two objectives: (i) the sum of total driving time and total waiting time, (ii) the sum of the tardiness values $(t_i - b_i)^+$.

Besides SP-ACO and SPSA, we also implemented a complete enumeration procedure combined with brute force simulation (CE/BFS, sample size 10^6) to evaluate proposed solutions, and a random search (RS) procedure with constant sample size per solution.

We generated 12 different problem instances at random. The problem size was chosen as $n = 9$ in all these instances, which was the largest number of customers for which we were able to compute the Pareto-optimal set by the CE/BFS approach within reasonable time. (On a PC Pentium 2.4 GHz, this took about 5 hours per test instance; the sample size was chosen as high as 10^6 to reach a sufficient accuracy of the objective function evaluations.) Test instances with $n = 9$ may seem as very small, but we would like to emphasize that in the combined setting of two different objectives combined with simulation-based objective function determination, the problem is highly nontrivial already for this instance size.

In the case of each problem instance, $n = 9$ customer points were selected uniformly at random from a square. Distances were computed as Euclidean distances between these points. It was assumed that traversing an edge of the square takes 10 time units. For each customer, a random decision was made on wether or not to assign a time window, using a fixed probability p_{TW} for the existence of a time window. If a time window was assigned, its length was selected uniformly at random between 6 and 60 time units, and its start time was selected uniformly at random between time 0 and the maximum time such that the whole time window was still contained in an interval of 120 time units. The service time distributions were chosen as uniform distributions on the interval between two values σ_{min} and σ_{max}.

Experiments were carried out for the following combinations of parameter values: For p_{TW}, we considered the cases $p_{TW} = 0.2$ and $p_{TW} = 0.3$, and for the intervals $[\sigma_{min}, \sigma_{max}]$, we chose the cases $[0, 20]$, $[0, 40]$ and $[10, 30]$. For each of the 6 parameter combinations, 2 random test instances were generated, such that 12 test instances in total were produced. Fig. 3 shows a typical Pareto front (Pareto-optimal set in objective space).

To have a fair comparison, each of the heuristic algorithms SP-ACO, SPSA and RS was given 20 seconds on a PC Pentium 2.4 GHz for a single run. The parameters of each heuristic were tuned to best possible results within these 20 seconds at the first test instance for the combination $p_{TW} = 0.2$, $[\sigma_{min}, \sigma_{max}] = [0, 40]$. After that, 100 runs for each of the three heuristics were performed on each test instance and evaluated with the help of the results obtained by CE/BFS.

The tuning yielded the following parameter values: (a) SP-ACO: $M = 10$, $\Gamma = 200$, $\rho = 0.00005$, $N^{(c)} = 2000$, $c_1 = 0.0005$, $c_2 = 0.000005$. (b) SPSA: $s = 10$, $N^{(1)} = 500$, $N^{(2)}$: initial value 1, increment 1, $N^{(3)} = 1$, $a = 1.1$, $b = 0.9$. A neighbor solution y was determined by m_n random 2-opt moves, where m_n was chosen uniformly between 1 and 4. (c) RS: sample size $N_{rs} = 200$ per randomly generated solution. Concerning the SP-ACO parameters, we remark that the value ρ may seem rather low compared to usual ACO implementations for single-objective deterministic problems. However, it should be noted that pheromone is not re-initialized at the beginning of each period, such that also low values of ρ effect distinct differences in the pheromone values towards the

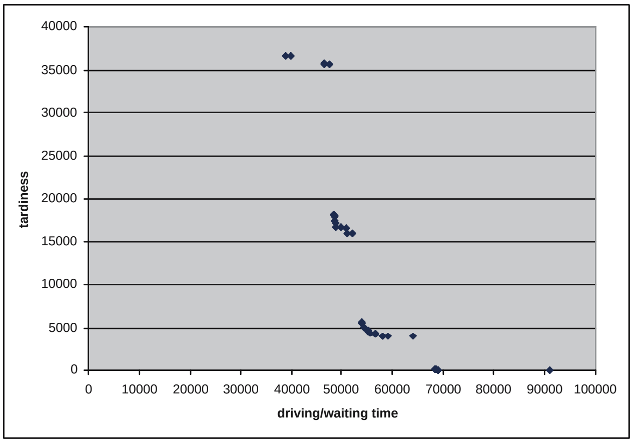


Fig. 3. Pareto front for the first instance of $p_{TW} = 0.2$, $[\sigma_{min}, \sigma_{max}] = [0, 40]$, time units multiplied by 1000

Table 1. Results for the 12 test instances (including 100 runs per instance and method)

		average ratio of found sol.			significantly better		
test instance	no. sol.	aco	sa	rs	aco : sa	aco : rs	sa : rs
0.2, [0,20] / 1	22	0.337	0.327	0.184	n	aco (s^{**})	sa (s^{**})
0.2, [0,20] / 2	17	0.663	0.571	0.444	aco (s^{**})	aco (s^{**})	sa (s^{**})
0.2, [0,40] / 1	41	0.421	0.285	0.218	aco (s^{**})	aco (s^{**})	sa (s^{**})
0.2, [0,40] / 2	34	0.450	0.450	0.343	n	aco (s^{**})	sa (s^{**})
0.2, [10,30]/ 1	25	0.428	0.305	0.222	aco (s^{**})	aco (s^{**})	sa (s^{**})
0.2, [10,30]/ 2	16	0.727	0.720	0.511	n	aco (s^{**})	sa (s^{**})
0.3, [0,20] / 1	21	0.701	0.561	0.366	aco (s^{**})	aco (s^{**})	sa (s^{**})
0.3, [0,20] / 2	19	0.748	0.652	0.448	aco (s^{**})	aco (s^{**})	sa (s^{**})
0.3, [0,40] / 1	23	0.620	0.662	0.515	sa (s^{**})	aco (s^{**})	sa (s^{**})
0.3, [0,40] / 2	27	0.432	0.480	0.329	sa (s^{**})	aco (s^{**})	sa (s^{**})
0.3, [10,30]/ 1	20	0.532	0.582	0.459	sa (s^{**})	aco (s^{*})	sa (s^{**})
0.3, [10,30]/ 2	11	0.550	0.451	0.338	aco (s^{**})	aco (s^{**})	sa (s^{**})

end of the procedure. Moreover, it can be observed that in the parameter choice above, the global pheromone increment c_1 is 100 times as high as the local pheromone increment c_2, which turned out as advantageous.

In the literature, many measures have been described for evaluating the results of multiobjective optimization heuristics (see, e.g., Jaszkiewicz [12]). Since our present results are only intended as a first experimental test, we used only one, rather basic evaluation metric, namely the ratio of Pareto-optimal solutions, as determined by CE/BFS, that are covered by one of the solutions in the output set of a considered heuristic algorithm. This is the first evaluation metric Q_1 for MOCO heuristics suggested in [12]; it goes back to Ulugu et al. [13].

In Table 1, we list the average Q_1 value over 100 test runs for each problem instance and each heuristic. In addition, for each problem instance and each pair of heuristics, we performed a (two-sided) Wilcoxon test to decide whether or not the difference between the indicated average Q_1 values for the two heuristics is statistically significant or not. Column 2 of Table 1 shows the total number of Pareto-optimal solutions, columns 3 – 5 the average Q_1 values over 100 runs, and columns 6 – 8 the results of the significance tests for the pairwise comparisons (n: no significance; s, s^* and s^{**}: significance at level $\alpha = 0.05$, 0.01 and 0.001, respectively). If a significant difference was found, the heuristic with the better results in the pairwise comparison was named. As it can be seen, both SP-ACO and SPSA outperformed RS in all cases with high significance. SP-ACO outperformed SPSA in 6 cases with high significance and was outperformed by SPSA in 3 cases, also with high significance. In the 3 remaining cases, there was no statistically significant difference between the results of SP-ACO and SPSA.

5 Conclusions

Two metaheuristics, SP-ACO and SPSA, for solving MOSCO problems have been developed and compared on a TSPTW with stochastic service times. The results do not indicate a consistent superiority of SP-ACO over SPSA or vice versa in the evaluation metric Q_1, but a random search approach is clearly outperformed by both. Future work should deal with more comprehensive outcome evaluations on extended test instance sets, and Q_1 should be supplemented by other metrics. Furthermore, extensions of other MOCO metaheuristics to the stochastic case should be included into the comparison.

References

1. Baesler, F.F., Sepúlveda, J.A., "Multi-objective simulation optimization for a cancer treatment center", *Proc. WSC 2001*, pp. 1405-1411 (2001).
2. Czyzak, P., Jaszkiewicz, A., "Pareto simulated annealing — a metaheuristic technique for multiple-objective combinatorial optimization", *J. of Multi-Criteria Decision Analysis* 7, pp. 34-47 (1998).
3. Doerner, K., Gutjahr, W.J., Hartl, R.F., Strauss, C., Stummer, C., "Ant Colony Optimization in Multiobjective Portfolio Selection", *Proc. 4th Metaheuristics International Conference*, pp. 243-248 (2001).
4. Doerner, K., Gutjahr, W.J., Hartl, R.F., Strauss, C., Stummer, C., "Pareto Ant Colony Optimization: A metaheuristic approach to multiobjective portfolio selection", *Annals of Operations Research* 131, pp. 79-99 (2004).
5. Dorigo, M., Stützle, T., *Ant Colony Optimization*, MIT Press (2004).
6. Ehrgott, M., Gandibleux, X., "A Survey and Annotated Bibliography of Multiobjective Combinatorial Optimization", *OR Spektrum* 22, pp. 425-460 (2000).
7. Fu, M.C., "Optimization for simulation: theory vs. practice", *INFORMS J. on Computing* 14, pp. 192-215 (2002).
8. Gandibleu, X., Sevaux, M., Sörensen, K., T'kindt, V. (Eds.), *Metaheuristics for Multiobjective Optimization*, Springer, Berlin-Heidelberg (2004).
9. Gutjahr, W.J., "A converging ACO algorithm for stochastic combinatorial optimization", *Proc. SAGA 2003* (Stochastic Algorithms: Foundations and Applications), Springer LNCS 2827, pp. 10-25 (2003).
10. Gutjahr, W.J., "S-ACO: An ant-based approach to combinatorial optimization under uncertainty", *Proc. ANTS 2004* (4th International Workshop on Ant Colony Optimization and Swarm Intelligence), Springer LNCS 3172, pp. 238-249 (2004).
11. Hughes, E.J., "Evolutionary Multi-objective Ranking with Uncertainty and Noise", in: E. Zitzler, K. Deb, L. Thiele, C.A. Coello Coello, and D. Corne (eds.), *First International Conference on Evolutionary Multi-Criterion Optimization*, pp. 329-343, Springer LNCS No. 1993 (2001).
12. Jaszkiewicz, A., "Evaluation of multiple objective metaheuristics", in: Gandibleu, X., Sevaux, M., Sörensen, K., T'kindt, V. (Eds.), *Metaheuristics for Multiobjective Optimization*, Springer, Berlin-Heidelberg, pp. 65-89 (2004).
13. Ulugu, E.L., Teghem, J., Fortemps, Ph., Tuyttens, D., "MOSA method: a tool for solving multiobjective combinatorial optimization problems", *J. of Multi-Criteria Decision Analysis* 8, pp. 221-236 (1999).

Self-replication, Evolvability and Asynchronicity in Stochastic Worlds

Chrystopher L. Nehaniv

Adaptive Systems, Algorithms, and BioComputation Research Groups,
School of Computer Science & Science and Technology Research Institute,
University of Hertfordshire, Hatfield, Hertfordshire AL10 9AB, United Kingdom
C.L.Nehaniv@herts.ac.uk

In Memory of John Maynard Smith and Victor Varshavsky

Abstract. We consider temporal aspects of self-replication and evolvability – in particular, the massively asynchronous parallel and distributed nature of living systems. Formal views of self-reproduction and time are surveyed, and a general asynchronization construction for automata networks is presented. Evolution and evolvability are distinguished, and the evolvability characteristics of natural and artificial examples are overviewed. Minimal implemented evolvable systems achieving (1) asynchronous self-replication and evolution, as well as (2) protocultural transmission and evolution, are presented and analyzed for evolvability. Developmental genetic regulatory networks (DGRNs) are suggested as a novel paradigm for massive asynchronous computation and evolvability. An appendix classifies modes of life (with different degrees of aliveness) for natural and artificial living systems and possible transitions between them.

1 Models of Time: Logical vs. Physical Time

We consider time in discrete dynamical systems. St. Augustine considered time as something intuitively graspable yet ineffable. Varshavsky distinguished two kinds of time: Time as a *logical variable* in a system defined by events vs. time as an independent *physical variable* [96], and studied self-timing and asynchrony theory for computing devices as the problem of reconciling the two types of time via design of system timing for the appropriate functioning asynchronous devices interacting with external environments.

For a single observer or location, we can consider three main views of the (logical) time:

1.1 Partial Orders as Models of Time

Aristotle considered events in time via ordering related to casuality (and motion), and time as defined by differences between states before and after (thus change is required for the passage of time).

1.2 Time as a Random Variable

Another view is to regard logical events, such as a discrete event clock-tick, as embedded in physical time but where a random variable takes values *event* or *no event* according to some distribution at successive discrete moments of physical time. (Instead of just one type of *event* more generally different particular events might be generated.) Here the passage of logical time, if used to increment a measuring counter, is monotonically but not deterministically related to the passage of physical time.

1.3 Algebras of Time: Semigroups as Models of Time

Following J. L. Rhodes (who refers to Aristotle), we can describe time algebraically. If α, β, and γ are each sequences of events in time, and the composite sequence β then γ is preceded by α, this is exactly the same as when β follows α and after both γ occurs. That is, the *associative law*

$$(\alpha\beta)\gamma = \alpha(\beta\gamma)$$

is a grammatical statement about sequences of events in time. The study of associative structures (semigroups) is thus the study of models of time.

As a simple application, we have algebraically classified all models of time allowing for only a single repeated event (or "clock-tick").[1]

Semigroups are intimately connected with deterministic automata, as sequences of inputs induce mappings of the set of states of the automaton to itself; these induced mappings thus comprise a semigroup (under the associative operation of function composition) which serves as a model of time in the automaton.

To pass to a nondeterministic or probabilistic automaton, there are several methods. A very general one, related to the construction of minimal automata, applies to the more general case of observations or measurements of any phenomenon at all. Observations of a given stochastic phenomenon can be treated via Crutchfield's ϵ-*machines*: from observations of an, in general stochastic, process one constructs a deterministic automaton in which transitions are single observations and in which states are equivalence classes of past histories for each member of which the probability distribution over the future histories is the identical. In other words, in a given state the future is conditionally independent of the past [15]. The semigroup of the ϵ-machine then serves as an algebraic invariant and model of the temporal dynamics of the given phenomenon.

2 Evolution and Evolvability

Evolution viewed as stochastic synchronous or asynchronous algorithm or temporal process is described here. Evolvability describes the capacity to which a

[1] The possible single event models are cyclic groups, the positive natural numbers under addition, and thresholded cyclic groups – in which the event can be repeated some number of times whereupon one enters a cyclic group [57].

particular evolutionary process is successful in generating adaptive individuals and will be discussed in detail later. After defining Darwinian evolution, we survey non-biological examples and the other evolution-like phenomena. Evolvability is then discussed in detail for these examples.

2.1 Definition of Darwinian Evolution

Evolution *is any* **dynamical population process** *[17,16] with the following characteristics, which one can regard as the semi-formal* **Darwin-Wallace axioms***:*

(1) **Heritability:** *Individuals have inherited information or material (genotype) from parent(s) that makes them similar to their parent(s) in some traits.*
(2) **Variability:** *Offspring may differ from their parents in their heritable material (genotype) and in other respects (phenotype).*
(3) **Differential Reproductive Success** [**selection**]**:** *depending on phenotype, which must depend at least in part on inherited traits, some individuals are more likely to have any (or more) offspring than others.*
(4) **Finite Resources and Turn-over of Generations:** *Lifespans of individuals are finite and the existence of only a limited number of individuals can be supported in the population at any moment.*

The above axioms yield a creative engine via a "*struggle for existence*" driving "*descent with modification*". Persistence and increase in distribution of heritable successful traits follow by (1) and (3), and creativity arises via (2). Competition for existence is due to (4). Note that a presupposition of (1) is that the population consists of well-defined individuals.

2.2 Stochasticity of Evolution

Stochasticity impinges on evolution usually (1) via the mechanism of genotypic variability, whereby inherited information is perturbed, but also (2) in phenotypic variability whereby the environment or constrained aspects of development lead to differences between parent and progeny. Differential reproductive success (3) refers to the probability of success at producing progeny depending on inherited information is therefore generally modeled as stochastic in nature.[2]

Evolution can thus be regarded as a very general class of stochastic algorithm with many instances occurring in nature, culture, and artificial systems.

2.3 Instances of Evolution *in Silico*

Genetic Algorithms and Evolutionary Computation. Genetic Algorithms (GAs) and allied methods are population processes for artificial evolution in com-

[2] Nevertheless, evolution is also possible in completely deterministic systems, e.g. the synchronous evoloop system [77] (see below), or any non-interactive genetic algorithm running with a given 'seed' for generating random numbers.

puters and have been introduce in many variants: *genetic algorithms* [27], *evolutionary strategies* [70,80], *evolutionary algorithms* [24], and others. The "vanilla" genetic algorithm in the style of Holland [27] is described here:
1. Create a population of fixed finite size of fixed-length bit-strings encoding candidate solutions to an optimization problem (initialized randomly or with domain knowledge)
2. Evaluate each against an objective function ("fitness function")
3. Copy individuals that do better with higher probability into next generation (a new population, with same population size) [selection]
4. Apply variability operators: mutation random bit-flips, crossover: recombination of substrings in individuals subtrees at random nodes between two individuals, and others.
5. Iterate 1-3 until satisfied.

Genetic Programming: Variant of GA. Genetic Programming (GP) is a variant of GAs introduced in the early 1990s by John Koza [30]. It has the following structure:
1. Do GA, but on populations of programs, not bit-strings (e.g. Lisp S-expressions, or parse trees in any programming language). Individuals are syntactically correct programs over some chosen set of basic operations, and terminals (constants and variables).
2. Behaviour or output of each program in population is evaluated against objective function ("fitness function")
3. Copy individuals that do better with higher probability into next generation (a new population, with same population size) [selection]
4. Apply variability operators respecting syntax: mutation replaces a subtree by random one (of the same type); crossover: exchange subtrees (of the same type) at random nodes between two individuals. New individuals are syntactically correct since the operators respect node typing.

Later variants of GP also introduced explicit support for modularity, named functions (so-called automatically defined functions (ADFs)) that can be called by the main result producing branch of the program.

Digital Organisms. Digital organisms were introduced by tropical evolutionary biologist and computer scientist T.S. Ray around 1989. Individuals in a finite computer memory are self-replicating programs running on a Darwinian operating systems (one in which mutations in data and flaws in computational operations occur with certain low probabilities). There is no objective function, so we have an instance of *natural selection.* The motivation is not optimization but artificial life as a generalization of biology. Several systems for the evolution of digital organisms have been developed: `Tierra` [69], the first one, gives rise to rich ecologies (parasites, obligatorily social hyperparasites, etc.). Insights that make it evolvable (as compared to random mutation of computer code) were inspired by biology and include (1) template recognition and matching (recognition based on "shape"); (2) all strings are syntactically correct assembler-like programs, and (3) small language size - no numerical constants are permitted in statements (but must be constructed in an organism's digital processor if needed). Space in computer memory and processor cycles are the fundamental resources for digital organisms; these and interaction amongst digital organisms determines their reproductive success in an emergent manner. `NetTierra` is an internet wide version with multithreading (an analogue of multicellularity), sensing by digital organisms of other sites over the network, and migration between computers. The `Avida` system added CPU cycle rewards for some computations and is currently most widely used [2], especially as a model for bacterial and an experimental test-bed for population genetics theory. `Physis` [23] is new system for studying evolvability of digital organisms in which organisms carry not only the code for their self-replication, but also code specifying the processor that will run it and, moreover, code specifying the language they will run on it: evolvable processors. This latter system allows the study of the evolvability of self-replication, including phenomena analogous to evolution of the genetic code for protein biosynthesis via translation to amino acid sequences from sequences of codons in very long oligonucleotides DNA/RNA.

3 Not-Quite-Darwinian Evolution

Several cases of what looks like evolution (and is often called evolution) fail to meet the Darwinian axioms. Generally, in such cases, there are dynamical similarities, but the problem is that one cannot identify well-defined individuals. An analogue of producing progeny in such cases is *persistence* [52], usually eventually with modification (and hence variability).

3.1 Software Evolution

In Software Engineering, the costs of so-called 'software maintenance' and 'software evolution', i.e. costs of modifying and adapting already released software, amount to billions of dollars annually (50-95% of all software costs [83,35]). Software is static, fragile and inflexible (except where adaptation need has been foreseen), but its context and environments of use change, hence requirements change. Software evolution has been characterized as managing change – see the work of Lehman, Goguen, and Berners-Lee (e.g. [35,25,26,8]). Persistence and reuse of software is an analogue to heritability [52]. If software code is regarded as heritable information, the severe problem of requirements change shows the need for software that possesses *phenotypic versatility* and *robustness to perturbation*, both of which are related to evolvability.

Software growth has been studied as a dynamical system with system-level, positive and negative growth laws [103]. There are no clear individuals, no population. But there is persistence and growth, and descent with modification. Are there principles in common with those biological evolvability? The answer seems to be yes, but they are not well-understood yet. Any software carries with it unbounded number of assumptions, which are progressively violated as time passes and context of use – and hence requirements – change [35]. A design principle similar to biological evolvability: attempt to be future-proof, robust to likely sources of change (see [8] on future-proofing and world wide web data and mark-up languages).[3]

3.2 Cultural Evolution

Other examples of evolution without a readily identifiable individuals in populations are the evolution of artifacts, and the evolution of behavior or cultural (memetic) evolution.

4 Evolvability

The evolution of life on earth has undergone several major transitions. Major transitions in evolution are studied in [11,45]: Free Replicators to replicators in compartments; RNA as gene/enzyme to DNA and protein (genetic code); prokaryotes to eukaryotes; asexual clones to sexual populations; protists to differentiated multicellular life (esp. [11,47]); solitary individuals to colonies with non-reproductive castes [45].

All of them involve transitions in the way information is used and most of them involve the advent of new types of individuality and thus new units of selection in populations of these new individuals.

Nothing like the complexity and creative power of organic evolution has been realized in artificially constructed evolutionary systems. Why is this the case? Computer scientists using evolutionary computation techniques quickly discovered that in some cases evolution was better able to find solutions than in others. Sometimes evolution completely failed as an optimization method, other times it worked well. Biologists had tacitly assumed that evolution by itself was sufficient to generate open-ended adaptivity and complexity of the kind they observed in nature (e.g. flowering plans, animals with complex body plans, etc.). But the frustration of computer scientists in some cases showed clearly that some systems were obviously more evolvable than others.

4.1 Krohn-Rhodes Complexity and Open-Ended Evolution

This leads to a constructive challenge problem.

Open Problem 1. (Open-Ended Evolution) Build a system that exhibits open-ended evolution. One in which complexity can grow arbitrarily large and new innovation and complex traits continue to arise.

Krohn-Rhodes complexity in algebraic automata theory using semigroups as models of time (or, e.g. Kolmogorov complexity) can be used to formalize

[3] The problem with being "future-proof" is that evolution by itself is a historical process that cannot predict anything about the future. In biology, robustness to likely sources of change appears to be achieved via *lineage selection*, i.e. lineages robust to the kind of change that has historically occurred are more likely to continue than others when changes of the same type reoccur in the future. In software, human design as well as such lineage selection may be operate. See also the discussion in the sections of this paper on GP code bloat and on the evolution of evolvability.

the notion of unbounded complexity growth, and explicit bounds on complexity increase in the course of smooth evolution can be computed [60]. Duplication-and-divergence is one generic method of maximizing jumps in complexity [62].

4.2 Origin vs. Fate of Variation

Most evolutionary theory (e.g. nearly all of population genetics) has been concerned with the fate rather than origin of variation [101]. Variability is the only source of creativity in the evolution axioms, and its generation must therefore be one of the keys to evolvability.

4.3 Definition of Evolvability

Evolvability has been characterized in various ways in the literature:

- "the ability of a population to produce variants fitter than any yet existing" (Altenberg [5])
- "genome's ability to produce adaptive variants when acted on by the genetic system" (Wagner & Altenberg [101])
- "the capacity to generate heritable phenotypic variation" (Kirschner & Gerhart [29])
- characterized by evolutionary watersheds opening the floodgates of evolution, such as with the advent of segmentation and body plans (Dawkins [20])

A synthetic definition is formulated here:

Definition. Evolvability *is the capacity of a population to generate adaptive heritable genotypic and phenotypic variation.*

In this definition, "**adaptive**" is understood as *fitter than any currently existing.*[4]

4.4 Genetic Algorithms and Evolutionary Computation: Evolvability Issues

Choice of *encoding* is a crucial issue for evolutionary computation: 'The Representation Problem". Encoding determines the genotype (e.g. bit-string) to phenotype (fitness evaluation) mapping (Genotype-Phenotype Map).

[4] This notion of evolutionary adaptivity is similar in its sense to that used in Altenberg's definition above [5], but is more specific in that it replaces "fitter than any *yet* existing" by "fitter than any *currently* existing". The reason for this is that fitness is a spatio-temporally *local* notion depends of the current organism-environment interactions and niches which are of course subject in general to temporal variation over generations. The production of fitter individuals might first proceed via *neutral evolution*, i.e. the production of new individuals with different genotypes and of equal fitness to those existing; this is known to increase evolvability in many examples (cf. [28,93].)

Genotype-Phenotype Relation

"The genotype-phenotype map is the common theme underlying such varied biological phenomena as genetic canalization, developmental constraints, biological versatility, developmental dissociability, morphological integration, and many more" - G. P. Wagner & L. Altenberg [101]

Variability operators determine the topology (neighborhood relations of genotypes) of the *fitness landscape* (S. Wright 1932 [107]), mapping genotype (or genotype and phenotype via environment interaction) to probability of reproductive success. Smoothness of the objective function on this landscape determines how well GAs can do their stochastic hill-climbing. If there are deep broad valleys between fitness peaks (local optima) that can't be traversed quickly enough, the system is not evolvable. Conversely, uphill paths reachable by a single step from local optima make landscape evolution-friendly.

To improve evolvability, the evolutionary strategies of Rechenberg and Schwefel [70,80] introduce the heritability of locus specific mutation parameters (for the variance of noise applied to numerical parameters under optimization).

Extradimensional bypass [14] is the adding of dimensions to the genetic 'search' space (e.g. by an insertion mutation or by duplication of a gene), in higher dimensional fitness landscapes, local optima often become saddle points; this is observed in protein evolution, and is related to neutral networks and robustness (via mutational buffering). Sometimes it has been used in evolutionary computation, e.g. via growth in genome size or duplication of all or part of the genome, to achieve improved evolutionary performance.

4.5 Genetic Programming and General Evolvability Issues

In GP, an important phenomenon is code bloat: for robustness to crossover, size of programs increases uncontrollably. They are full of junk in order to withstand crossover with lower chance of distribution. Making multiple crossover occurrences more likely for large trees according to their size eliminates this trend [91].

This is a particular instance of a general principal in the evolution of evolvability: *Evolution favors lineages that robustness to disruption from the variability operators experience by the evolving population.* See [66] for a related study on linkage and crossover, and [92] for the neutral evolution of mutational robustness.

Modularity: Automatically Defined Functions (ADFs) [31] are functional modules that can be called from various locations in a program. Using these can measurably increase evolvability [90]. This is related to analogous principles in software engineering for evolvability: code factoring, appropriate modularity, re-use (e.g. [65,52,83]).

4.6 Properties and Mechanisms of Evolvability

What makes an instance of the stochastic algorithm, evolution, evolvable? A list of properties and mechanisms that seem closely related to evolvability is presented here. In many cases it is unclear whether we are examining a prerequisite for, or a consequence of, evolvability, or possibly both (via the circular casuality of the dynamical evolutionary process), or perhaps an incidental property.

1. Developmental Plasticity: Universal responsiveness to interaction with the environment, an incessant, continual coupling throughout life. (lifelong viability; multiple cell types; complex life cycles; multiple developmental pathways/behaviours/morphs; continual self-creation and maintenance in interaction with environment/others). This property is almost unknown in artificial systems, standard population genetics models, the 'new synthesis', unimodal evolutionary models; but see West-Eberhard [104] and also Varela [94].
2. Flexibility/Rigidity of Genotype-Phenotype Relation Robustness (to Heritable and to Developmental/ Environmental Perturbations)
3. Duplication-Divergence: From one, many! (cell types, castes, genetic regulatory networks (GRNs), segmentation, generic complexity increase)
4. Differentiation, Local Adaptation and Control
5. Appropriate Modularity, Compartmentation Potential to Combine Lower Level Units: one from many!
6. Symbiogenesis
7. New Individuality (e.g. Multicellularity, Compartmentation; Linkage)
8. Use of Signaling, Switches, Signal Transduction, & Feedback Control
9. Employment of Evolutionary Dynamics (within individuals!)
10. Redundancy
11. Extradimensional Bypass

4.7 Duplication and Divergence

Gene duplication is remarkably frequent and important in biological evolution [63], and subject to complex evolutionary dynamics [39]. The creation of a full or partial extra copy of a gene (or other component) frees one copy or both copies to specialize functionally, or one copy to acquire a new function. Duplication and divergence in biological evolution [63] is thus a generic mechanism for the generation of variability, of great potential creative power.

Duplication and divergence (Figure 1) is also exemplified by division of labor among cells or tissue types in a body, or castes in a social insect colony. In differentiated multicellularity, growth via cell division and followed by specialization into cell types (e.g. into soma and germ lines) provides an opportunity and mechanism to exploit asynchronous parallel processing by closely related entities to achieve adaptation at a higher level of individuality.

Complexity increase via increase via duplication and divergence, e.g. increase in number and role of cell types [9], or the acquiring genomes [43] (which does not involve duplication and divergence) can apparently realize known, sharp theoretical bounds on the evolution of biological complexity [60].

Differentiation. Differentiation of multiple copies of the same entity as in differentiated multicellular involves the following properties:

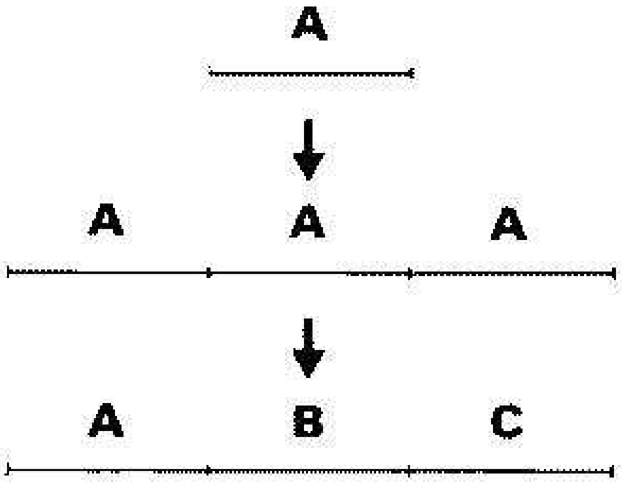


Fig. 1. Duplication-Divergence: A Generic Path toward Complexity Increase and Evolvability (after J. Maynard Smith [44])

- Multiple copies of regulatory mechanism in similar units (e.g. genetic regulatory networks, cells, individuals, etc.)
- Local state
- State inherited by lineage (e.g. Cell types, growth and morphogenesis, epigenetics methylation, etc.)
- Local adaptation to local conditions
- Long- and medium- distance interactions
- Growth from single unit to a differentiated many, with changing topology
- Division of labour

5 Self-production and Reproduction

How is it possible for a mechanistic system to produce something as complex or even more complex than itself? This problem motivated von Neumann to study the physical and logical basis of self-reproduction using automata models. Von Neumann considered automata capable of (1) examining and copying any pattern or specimen given to them, or of (2) production of any object starting from a logical description.[5] Either approach leads to a solution to von Neumann's problem: in the first method, present the automaton with an entity as or more complex than itself; in the second, present it with a logical description of one. Of

[5] See especially notes of von Neumann's fifth lecture "Re-Evaluation of the Problems of Complicated Automata - Problems of Hierarchy and Evolution" in Part I of [99] delivered in December 1949 (and edited and reconstructed by A. W. Burks). In both cases the word "any" must taken as having scope over a particular very large class of bounded structures whose existence possible in the ambient environment.

course for this to work, it is necessary to construct such *universal constructing* automata with these capabilities or to demonstrate their existence.[6]

These two approaches lead to solutions of the problem of self-reproduction by *self-examination* vs. *heritable encoded information* respectively. One presents the universal construction automaton with itself (or a copy of itself), or with a logical description of itself, respectively. In the latter case, a copying component of the automata can be used to copy the logical description (regarded part of the entity), which thus becomes *heritable* genetic information. Von Neumann showed how to construct such an automaton in a synchronous cellular automata network using the second method [99, Part II].

Mutations or errors in the construction process could lead to lethal or non-lethal variant copies and hence provide the variability required for evolution to act. Conceivably, this could therefore lead to the evolution of more and more complex automata. Although von Neumann considered this possibility, so far no one has been able to shown in detail how it could be realized.[7]

It is remarkable that von Neumann's solution used genetic, inherited information in two roles: (1) blindly copied and (2) executed, before the structure of the heritable genetic material in life on earth was uncovered by Watson and Crick's 1953 detailed description of the structure of DNA revealing its essentially *digital* nature with similar dual roles [102]. Thus, von Neumann's work on his automata models even anticipated the important transcription ("blind copying") and translation ("executability") properties of genetic material found for DNA, with the former realized by complementary pairing of bases and the latter via template matching and the genetic code sequentially mapping codons (triplets of "letters" of DNA) to the amino acids in proteins (along with numerous regulatory intricacies).

From the beginnings of the study of self-reproduction in artificial systems initiated by von Neumann already in 1948, the primary formal model has been synchronous cellular automata in which configurations develop that eventually may include an unbounded number of copies of the original. The models constructed by von Neumman and his successors have amply demonstrated that self-reproduction is indeed possible in artificial systems.

The different possibilities for achieving self-reproduction have implications for our understanding of the origin of life, the nature of organic life, and for the possibilities of life as it may exist elsewhere in the universe. Szathmáry [84] offers a classification of replicators applicable to natural and artificial systems along the dimensions of the replication process (*holistic* vs. *modular*, and *genotypic* vs. *phenotypic* (the latter is defined by non-modular copying of functionality)) and of variability (*limited* vs. *unlimited heredity*, where the latter requires that the number of possible variants be much larger than the number of individuals in the population).

[6] Portions of this section are based on the author's paper [55].

[7] To demonstrate this, a suitable rigorous complexity measure would of course be a pre-requisite (cf. [62]).

Self-reproduction is of course a prerequisite for any independent evolutionary process. Sending information, instructions on how to build copies of desired structures using local materials, into an environment rather than sending all necessary materials into that environment represents more economical methods of space exploration and colonization. See the NASA report edited by Freitas and Gilbreath (1980) for further potential examples and applications of self-reproduction to space science, e.g. self-replicating and self-maintaining lunar factories.

Nature abounds with asynchrony. Cells in a multicellular organism or organelles or molecules within a cell apparently have no access to a central clock signal. Can von Neumann's problem still be solved without synchrony? Might the restriction to synchronous update be relaxed?. In building an artificial self-reproducing entity is it really necessary to have a single global synchronization signal that reaches all parts of the entity simultaneously (or at least within a well-defined tolerance)? If local parts of the configuration are ready to change their state, is it realistic and practical to assume that they must wait until all other parts of the cellular space are also ready to update their states?

We can indeed free all cellular automata models of self-reproduction as well as all cellular automata models of evolution, universal computation, and universal construction from the need for synchronous update ([55,53], and below). This is accomplished by an elegant simple mechanism that allows one to construct an asynchronous automata network that is capable of emulating the behavior of a given synchronous automata network. State updates in the asynchronous model may be produced by practically any asynchronous update mechanism whatsoever[8] (e.g. updates may be sequential, occur randomly – locally distributed according a probability distribution, be partially simultaneous, etc., or even synchronous). The result for cellular automata is a special case of a more general theorem for automata networks with inputs due to the author (Theorem 1 below, [56]).

We describe below the construction for making any automata network's computation asynchronously realizable, give examples that illustrate how the use of "local time" frees cellular automata networks from the need for global synchronization, and display asynchronous examples of self-reproduction and evolution in cellular automata in the context of discussing evolvability in natural and artificial systems.

5.1 Models of Self-reproduction

Von Neumann's original constructive demonstration (begun in the 1940s and completed by Burks in the 1960s) of self-reproduction of a configuration of states in the cellular automata network has the properties that the self-reproducing configuration is capable of universal computation (in Turing's sense) and of universal construction – loosely speaking, the ability to fill any compact area in

[8] The only essential restriction is that each local automaton is updated an unbounded number of times, and a given node from the viewpoint of another cannot have been updated infinitely often in the past.

the cellular space with any desired pattern. These properties were included in addition to the ability of the replicator to make a copy of itself, and could also be used to support this ability. Namely, *universal construction* (as the ability to fill any compact region of the cellular space with arbitrary configurations) guarantees that a copy of the self (including its 'instruction tape' which is present in many examples) can be constructed. However, von Neumann's design of a self-reproducing universal computer and constructor was infeasibly large and has never been fully implemented and executed through a reproduction cycle on a computational device.

Langton's (1984) definition requires that a copy is constructed but realizes neither universal computation nor universal construction [33]. Langton implemented and studied the first example of feasible self-reproduction in cellular automata, using an 8-state cellular automaton with an initial configuration of 86 cells, that produces a first offspring after 151 time steps and then proceeds to fill up available space with copies. To avoid trivialities while avoiding the complexity of von Neumann's model, *Langton's criterion* [33,34] was proposed as a necessary condition on self-reproduction and requires that information is treated in the two ways identified above: as instructions that are *executed* ('translation') and as data which are blindly *copied* ('transcription'). These properties are also present in and abstracted from von Neumann's and later Codd's examples [13], and were by that time also known to be characteristic of biological self-reproduction. Encoding of heritable information in the shape of a configuration or using self-inspection represents another feasible mode of encoding heritable variation in self-reproduction (cf. [32,69,50,54]). Subsequent examples of Byl [12] and Reggia et al. (e.g., [71,38]) simplified the self-replicating loop of Langton toward minimality, with fewer states, simpler transition rules, or less cells in the initial configuration. In some cases the simplifications are so severe that it is debatable whether *nontrivial* self-reproduction has been achieved (e.g. according to Langton's criterion).

Subsequently, various researchers kept Langton's requirements for self-reproduction, but have added more and more computational power to the relatively small self-reproducing cellular automata configurations (in comparison to von Neumann's solution). These trends are surveyed by Lohn [37], who also describes the evolution of cellular automata rules that support self-reproduction (see also [38]). An annotated bibliography with some links to various relevant on-line resources can be found at Moshe Sipper's Artificial Self-Replication page [82].

H. Sayama [76,77] has constructed variants of the self-reproducing Langton loop which exhibit self-dissolution once they can no longer reproduce, thus freeing up space for reuse by progeny, and most interestingly, another similar variant called "evoloop" which exhibits heritable variability in loop size and is subject to evolution via interaction among descendants of a common ancestor acting as a selective force ([75,77], and below). Heritability, variability, and differential survival in an environment with limited resources are present in his evoloop when run in finite spaces. Thus evoloop appears to be the first convincing example of an evolutionary process occurring in cellular automata.

5.2 Self-reproduction, Individuality, and the Heritability of Fitness

What constitutes self-reproduction?

The definition is not uncontroversial. We have already mentioned that von Neumann included universal computation and universal construction in order to exclude trivialities, such as the simple example of spreading activation. Langton abstracted the properties of inherited information being both copied and executed.

E. F. Moore [49] defines a configuration C to be *capable of self-reproducing n offspring by time t* if starting from the initial conditions of the entire cellular space at time $t = 0$ such that the set of all non-quiescent cells of the space is an array whose configuration is a copy of C there is a time $t' > t$ such that at time t' the set of all non-quiescent cells will then be contained in an array whose configuration includes at least n copies of C.

Lohn and Reggia [38] give the following definition:

> "A configuration C is self-replicating if the following criteria are met. First, C is a structure comprised of more than one non-quiescent cell and changes its shape during its self-replication process. Second, replicants of C, possibly translated and/or rotated, are created in neighbor-adjacent cells by the structure. Third, there must exist a time t such that C can produce i or more replicants, for any positive integer i, for infinite cellular spaces (Moore's criterion). Fourth, if the self-replication begins at time t, there exists a time $t + \varDelta t$ (for finite $\varDelta t > 1$) such that the first replicant becomes isolated from the parent structure."

The issue of exactness of the copy is problematic since it is not desirable to exclude the possibility of variability. Variability among offspring is certainly present in biological systems, and, as Darwin showed us, is necessary for evolvability. Vitányi [97] introduced sexual reproduction in cellular automata and Sayama [76], mentioned above, has demonstrated variability and (deterministic) evolution occurring in cellular automata.

A discussion of the difficulties in formulating a rigorous definition of self-replicating or self-reproduction is given by Nehaniv and Dautenhahn [58], who point out that even in accepted cellular automata models of self-reproduction there are rarely two copies of the original configuration present at exactly at the same time when reproduction is generally accepted to have occurred (e.g. in the von Neumann or Langton models), and it is certainly not the case when the first offspring has been produced. The various copies of the configuration may be at different stages in their "lifecycles" and not have exactly the same configuration of states. They suggest looser criteria on identity of copies to allow 'species' of non-exact copies to be acknowledged as offspring, and also loosen the restriction on the presence of copies all at the same time (e.g., offspring that have to grow into adults are still regarded as offspring even though they are never in exactly the same state of development as the parent). Adequate formal definitions of "member of the same species" and of "individual" are still lacking in the sciences of the artificial, including the study of self-reproduction in artificial systems. Although these concepts are clearly fundamental to biological evolution, even within biology there is still on-going controversy and current research into appropriate definitions for these concepts [43].

Coming back to Darwin's ideas, some degree of *heritability of fitness* is required for nontrivial evolution to occur. With self-reproduction, the similarity of offspring to the parents and the similarities of the environments in which the replicators find themselves is often enough to account for this. However, beyond the level of simple replicators, heritability of fitness requires more explanation, e.g. in considering multicellular lifeforms with differentiated cell types, subunits which are themselves replicators comprise populations within the body that are themselves potentially subject to evolutionary pressures [11,45,47,48]. For example, cancer is an example in which reproduction and evolution occur at the lower cellular level at the expense of the higher organismal one. Multicellularity can arise (in certain conditions on mutations and cost of defection) where fitness (reproductive success) at the higher, whole organism level emerges in a trade-off against short-term fitness at lower, cellular level. Guaranteeing that the offspring are similar to the parent by suppression of freedom at the lower level in exchange for benefits is the first functionality required of any higher unit of fitness such as a multicellular organism. The latter must employ mechanisms to balance the tendency of the lower level to defect by sufficient benefits from cooperation in the higher level unit, in order to persist over evolutionary time [48,47].

In asynchronous self-reproduction the very fact that the relative synchronization of the entire state of the "organism" is uncertain contributes to this problem of heritability of fitness.

5.3 Self-repair: Biological Methods and Generalizations

Self-reproduction and self-repair (or self-maintenance) are often closely related in biology, and an understanding of self-reproduction can thus contribute to our ability to create self-repairing, self-maintaining hardware and software. von Neumann [98] considers synthesis of reliable organisms from unreliable components through redundancy and degeneracy, but apparently did not extend this during his lifetime to self-repair or relate it directly to self-reproduction. Automata models and circuitry capable of autonomous fault-detection and self-repair is an increasingly important area [40,89,41], both as an means to understand principles of biological organization, and also in technological applications, including robust computation, and especially for mission critical systems in space sciences. The capacity of a system to generate parts and components of its own structure and to establish their organization might obviously be useful in generating and installing a replacement parts in maintaining itself.

5.4 Self-maintaining and Self-creating Systems: Autopoiesis

Such a capacity for production of constituent components in the building and maintenance of them in an organized structure and dynamical process in the face of favorable or unfavorable perturbations (such as damage, production of waste, and entropic decay) is identified, according to biologists F. Varela and H. Maturana, as the key property, *autopoiesis* ("self-production"), defining living systems [95].

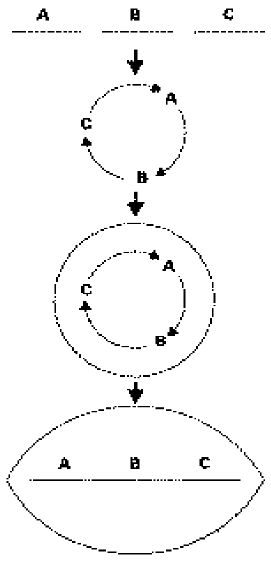


Fig. 2. Compartmentation as Proto-Self-Maintenance: Components of an Autocatalytic Cycle in a Protected Environment with only Constituent Partners present in suitable proportions (After J. Maynard Smith [44])

Neither von Neumann's work on self-reproducing automata, nor the studies following him have addressed via constructive models this aspect of living systems. Langton's work on self-reproducing loops (removing the universal construction and universal computation capacity) and its successors have focused on minimal models of self-reproduction, first by minimizing the size of replicators [12,71,37], and then adding various computational and other abilities [86].

Autocatalysis, Compartmentation, Early Life. An autocatalyst, by definition, promotes its own formation from other materials, and thus is in some sense self-replicating [64]. Autocatalysis implies dynamically cycles, potentially continuing without end. Compartmentation proceeds via isolation of an environment inside a vesicle or membrane (see Figure 2) within which conditions are conducive to the autocatalytic cycle and the production and maintenance of the membrane. Self-replication in early life might thus have arisen as a bifurcation in the dynamics of a self-producing, self-maintaining system resulting in response to some perturbation. For example, due to increase in size or due to accidental damage, an early self-maintaining vesicle is broken into two parts along its membrane; each surviving component repairs itself comprising a new self-producing organization. Any heritable aspect of organization that increases stability following such an event leads to similar descendants, potentially growing exponentially in number.

5.5 Challenge Problems

Work in constructive biology and the theory of self-reproducing automata discussed above leaves several challenges unanswered:

Open Problem 2. Realize construction universality in any computationally feasible, implementable models.

Open Problem 3. Construct an autopoietic self-reproducer whether synchronous or asynchronous in logical, kinematic or physical realization.

Open Problem 4. Solve open problem 3, adding heritable variation to realize evolution in a population of autopoietic self-reproducers.

Genetic Acquisitions. Sex in biology is, by definition, nothing more than the transfer or exchange of genetic material. It occurs, e.g., between homologous chromosomes in meiosis, or in the uptake of DNA from the environment by bacteria. If precious genetic information is lost due to damage to DNA, or if an organism is doing poorly due to heavy environmental stress, recourse to the genetic material from others may save the day by providing an undamaged source – though quite possibly different in content – of relevant genetic information; see [46] for the role of sex in repair.

A more extreme acquisition of genetic material than in sex is the acquisition of entire genomes in *symbiogenesis* (the advent of a merged entity, derived from replicators from two evolving populations, which becomes the unit of selection in an evolutionary process – see Appendix), and resulting speciation [43]. Such processes were involved in the acquisition by eukaryotes of the bacterial ancestors of mitochondria and, in plants, of chloroplasts [42].

Open Problem 5. Construct an evolutionary system in which different populations of autopoietic self-reproducers interact, and in which one species acquired the genome of another, realizing symbiogenesis.

5.6 Evolution of Evolvability

Finally, how can evolvability itself evolve? Lineage selection arguments suggest that lineages will survive that are robust to variational operators acting an evolving population [66,5]. The genetic code and genotype phenotype mapping, and genetic switches have all arisen in organic evolution of life on earth. Evolution of developmental genetic regulatory networks in constructed artificial systems interacting with their environments (see sec. 8) is suggested as one road toward achieving some small reflection of what nature has achieved.

6 Local Time

We adopt here the view of local time as a random variable to approach the asynchronization problem for automata networks. That is, given a synchronously updating network of automata, we want to construct another network of automata, with essentially the same behaviour, but in which at each node logical time is determined by an (unknown) local random variable. It is at first unclear whether this is possible at all, since simple experiments with common cellular automata networks show that the behaviour of the system generally changes radically in a qualitative sense when abandoning synchronous update.

An **automata network** consists a collection of automata $\mathcal{A}_v$ associated to the vertices $v \in V$ of a locally finite directed graph $\Gamma = (V, E)$, and a global input alphabet X and local transition rules δ_v. A state of the network is a choice of state for each component automaton. Given a global input $x \in X$ and a state of the automata, the *next state of the network at node* v is determined by the state of the automaton at v, the states of the automata in the neighborhood of v (i.e. at those nodes w which have an edge $(w, v) \in E$ to node v), and x. Thus the new state of the automaton at node v may be written as

$$q'_v = \delta_v(q_v, q_{N(v)}, x),$$

where q_v and $q_{N(v)}$ are, respectively, the current state at v and the states q_w of all nodes w in the neighborhood of v.

An automata network is *synchronous* if every node advances to its next state simultaneously. Otherwise it is called *asynchronous.*[9]

An automata network is called a *cellular automaton* if it has only only one global input letter (i.e. the alphabet satisfies $|X| = 1$ and its unique letter can be considered a "clock tick"), and the local transition functions, local automata, and neighborhoods at each node are isomorphic. Synchronous cellular automata have been well-studied since they were introduced by S. Ulam and J. von Neumann in the middle of the last century (e.g. [99,13,10,88]).

Definition (Emulation). Let $\mathcal{A}$ be an synchronous automata network over a directed graph $\Gamma = (V, E)$ with global state set Q and $\widehat{\mathcal{A}}$ be an asynchronous automata network with the same input alphabet X, a directed graph $\Gamma' = (V, E')$ with the same set of nodes, and global state set $\widehat{Q}$. Let $\pi : \widehat{Q} \to Q$ be a function from global states of the asynchronous automata network to global states of the synchronous one, such that $\pi^v(\hat{q}) = (\pi(\hat{q}))^v$ depends only on $\hat{q}^v$ for all $\hat{q} \in \widehat{Q}$. Thus we can denote $(\pi(\hat{q}))^v$ by $\pi(\hat{q}^v)$.

Regarding physical time as modeled by non-negative real numbers and logical time in the synchronous automata network as modeled by the natural numbers,

[9] We assume for asynchronous update that there are no delays in state information reaching a node in a local transition and that local updates may be regarded as instantaneous. We do *not* require any particular ordering of updates of nodes, only that, after an update of any given node, each node will still be updated an unbounded number of times in its future. Simultaneity of the update of any two nodes is permitted (but not required), and massive asynchronous parallelism is thus possible.

We may assume an ambient physical time in which stochastic update events occur, i.e. particular subsets of the sets of nodes are updated at discrete moments of physical time; every node is updated an unbounded number of times; and no node is updated an infinite number of times within a bounded interval of physical time.

In our model of asynchronous networks, based solely on a function of its local neighborhood and state information, a local automaton may choose to read or delay reading the next letter in global input sequence. Reading of the global input sequence is thus not synchronized but happens independently at each node.

See [56] or [21, Ch. 7] for more details and proofs of theorems stated here.

we then say that the behavior $\hat{q} : \mathbb{R}^+ \longrightarrow \widehat{Q}$ of $\widehat{\mathcal{A}}$ starting in state $\hat{q}_0$ for update pattern determined by local random variables at each node (as above) and input sequence $x_1, x_2, \ldots$ ($x_i \in X$ for $i \in \mathbb{N}$) *emulates* the behavior $q : \mathbb{N} \longrightarrow Q$ of $\widehat{\mathcal{A}}$ starting in state q_0 with the same input sequence under the projection π if there exists a spatial-temporal covering $\lambda : \mathbb{R}^+ \times V \longrightarrow \mathbb{N}$ such that the following diagram commutes for each $v \in V$:

$$\begin{tikzcd}
\mathbb{R}^+ \arrow[r, "\hat{q}^v"] \arrow[d, "\lambda(-,v)"'] & \widehat{Q}^v \arrow[d, "\pi"] & \text{(asynchronous)} \\
\mathbb{N} \arrow[r, "q^v"] & Q^v & \text{(synchronous)}
\end{tikzcd}$$

That is, $\pi(\hat{q}^v_t) = q^v_{\lambda(t,v)}$, with q^v_n = state in $\mathcal{A}$ of node v at time $n \in \mathbb{N}$ and $\hat{q}^v_t$ = state in $\widehat{\mathcal{A}}$ of node v at time $t \in \mathbb{R}^+$.

Thus the behaviour of $\widehat{\mathcal{A}}$ projects onto and completely determines the behaviour of $\mathcal{A}$.

Theorem 1 (Emulation by Asynchronous Automata Networks [56]). *Let any synchronous automata network $\mathcal{A}$ over a locally finite digraph $\Gamma = (V, E)$ with local automata $\mathcal{A}^v = (Q^v, X^v, \delta^v)$ ($v \in V$) and external input alphabet X be given.*

We construct an asynchronous automata network $\widehat{\mathcal{A}}$ (with the same input alphabet X) such that every possible behavior of $\widehat{\mathcal{A}}$ with input sequence $\{x_n\}_{n>0}$ emulates the (only possible) behavior of $\mathcal{A}$ with input sequence $\{x_n\}_{n>0}$, when $\widehat{\mathcal{A}}$ starts in an initial global state $\hat{q}_0$ depending only on the initial global state q_0 of $\mathcal{A}$.

Moreover, the following hold:

1. *The underlying digraph for $\widehat{\mathcal{A}}$ is the reflexive-symmetric closure of the digraph for $\mathcal{A}$.*
2. *For each vertex v, the local automaton $\widehat{\mathcal{A}}^v$ at vertex v in $\widehat{\mathcal{A}}$ is "not much more complicated" than the local automaton $\mathcal{A}^v$ at v in $\mathcal{A}$. Indeed, $\widehat{\mathcal{A}}^v$ is a direct product of $\mathcal{A}^v$, an identity-reset automaton, and a modulo three counter. In fact, $\mathcal{A}^v$ has state set state set $\widehat{Q}^v = Q^v \times Q^v \times \{0, 1, 2\}$.*
3. *The projection $\pi : \widehat{Q} \longrightarrow Q$ is given locally by $\pi^v(q^v, b^v, r) = q^v$ for $(q^v, b^v, r) \in \widehat{Q}^v$.*
4. *The starting state of $\widehat{\mathcal{A}}$ is given by $\hat{q}^v_0 = (q^v_0, q^v_0, 0)$ for all $v \in V$.*
5. *Furthermore, the spatial-temporal covering of the emulation satisfies*

$$|\lambda(t, v) - \lambda(t, v')| \leq \lfloor \frac{d(v, v') + 2}{3} \rfloor.$$

Note that updates of local states in the constructed emulating automaton are essentially arbitrary.

We call $\lambda(t, v)$ the *local time* of the synchronous automaton $\mathcal{A}$ at vertex v for time t in the emulating asynchronous automaton $\widehat{\mathcal{A}}$. Of course, λ depends in general on the update pattern for the particular behavior of $\widehat{\mathcal{A}}$. Thus (5.) above says that the difference in local time at two nodes in the emulating asynchronous automata network is bounded above by approximately one third of the distance between them.

Brief Sketch of Proof and Construction: Let $N(v)$ denote the set of neighbors of node v, and let $\widehat{N}(v)$ denote the neighbors of v in the reflexive-symmetric closure of Γ, which gives the topology of the emulating asynchronous automaton $\widehat{\mathcal{A}}$. The local update function in $\widehat{\mathcal{A}}$ is defined as follows, where φ^v and $\widehat{\varphi^v}$ give the action at vertex v in $\mathcal{A}$ and $\widehat{\mathcal{A}}$, respectively, as a function of their arguments, depending only on local state in the neighborhood and global input letter:

$$\widehat{\delta}^v((q^v, b^v, r^v), \widehat{\varphi}^v(\hat{q}, x)) =$$
$$\begin{cases} (q^v, b^v, r^v) & \text{if } r^w = r^v - 1 \bmod 3 \text{ for some } w \in \widehat{N}(v) \\ (q^v, b^v, r^v + 1 \bmod 3) & \text{if } r^w \neq r^v - 1 \bmod 3 \text{ for all } w \in \widehat{N}(v) \\ & \quad \text{and } r^v \neq 0 \\ (q^v \cdot \varphi^v(c, x), q^v, 1) & \text{otherwise,} \end{cases}$$

where c be an arbitrary state of $\mathcal{A}$ such that for each $w \in N(v)$,

$$c^w = \begin{cases} q^w \text{ if } r^w = 0 \\ b^w \text{ if } r^w = 1. \end{cases}$$

Note each r^w must lie in $\{0, 1\}$ in determining c^w of the third case, as necessarily $r^v = 0$ in third case and $w \in N(v) \subseteq \widehat{N}(v)$ implies $r^w \neq 2 \mod 3$.

Thus, in the emulating automata the neighboring nodes carry a copy of "current state" and "old state" in case a neighbor needs to read either one. The third component of state carries a modulo 3 value. The neighbors of any node v can be shown inductively to receive the same number of increments modulo 3 as node v, plus or minus one. Thus neighboring nodes differ by at most 1 modulo 3 in this component. In computing its local update, a node can check whether each of its neighbors is in the past, future, or in sync with it. If any neighbor is in the past, no update is performed (and the global input letter is not read). Otherwise, we increment the modulo 3 counter and on every third counter increment, copy current state to old, and update the current state according to the update rule of $\mathcal{A}$ and the global input letter. In the latter case, every neighbor must be in sync or in the future relative to the node in question, so the appropriate state of the neighbor node in $\mathcal{A}$ can be determined from the current or past state component of the corresponding neighbor in $\widehat{\mathcal{A}}$.

Using the fact that nodes differ by at most one in the number of increments they receive in the third component and using local finiteness another lemma shows *freedom from deadlocks* – a node can only be waiting for one that has received one less such increment and only finitely many can have occurred, so any chain of waiting ends when the automata at its end (with fewest increments so far) receives an update. Induction then shows that behavior of the synchronous automata network can be recovered uniquely from any behaviour of the asynchronous one by a *spatial-temporal section* $\lambda(t, v)$ equal to the ceiling of the one-third of two plus the number of counter increments at node v. (See [56] for full details.) □

A special case of essentially this construction was found independently and presented by K. Nakamura [51], the author [55,53], and T. Toffoli [87,88], with full rigorous proof of its correctness given in [56]:

Corollary 1 (Asynchronous Emulation of Cellular Automata Networks Theorem). *If $\mathcal{A}$ is a synchronous cellular automaton then there is an emulating asynchronous cellular automaton $\widehat{\mathcal{A}}$.* □

Open Problem 7. Prove an analogue of the Asynchronous Emulation Theorem for Automata Networks that may dynamically change their topology and number of component automata. (Or, more weakly, prove such an analogue for cellular automata networks.)

6.1 Temporal Waves, Asynchronous Game of Life and Universal Computation

Temporal Waves. From what we saw in the last section, it follows that local time in the asynchronous emulating network for nodes at distance d differs by at most about a third of the distance between them. [10] Since the values of the modulo 3 synchronization counter differs by at most 1 between neighbors in the asynchronous emulating network, this spatial continuity of the modulo 3 counter state entails that updates corresponding to simultaneous ones in the synchronous network move as temporal waves across the space of the asynchronous network.

Asynchronous Game of Life. This phenomenon is illustrated here with an asynchronous version of John Conway's famous synchronous cellular automata network, "The Game of Life".

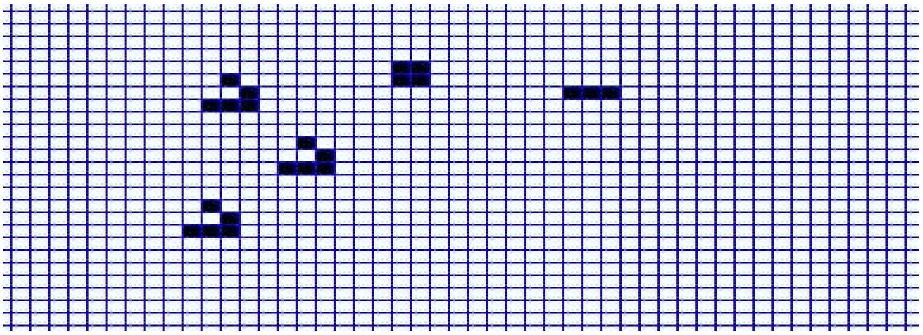
Let us apply the construction to Conway's (synchronous) Game of Life. A local automaton in synchronous Life has two possible states (quiescent (0) or alive (1)) and the following transition function: if a cell is quiescent and has exactly 3 neighbors that are alive, its next state is alive. If a cell is alive, and it has either 2 or 3 live neighbors (not including itself) then it stays alive, otherwise it becomes quiescent. It is well-known that, in principle, universal computation can be implemented in a infinite two-dimensional (synchronous) cellular automaton running Conway's rule (for an enjoyable yet highly readable and detailed overview see chapter 1 of [81]).

Figure 3 (top panel) shows an initial configuration of some well-known structures in Conway's Game of Life as an initial configuration for the corresponding asynchronous cellular automaton: Three gliders which move across the space, a stable 2×2 box, and a blinker (a row of 3 cells, that becomes a column of 3 cells, then a row of 3 cells, and so on).

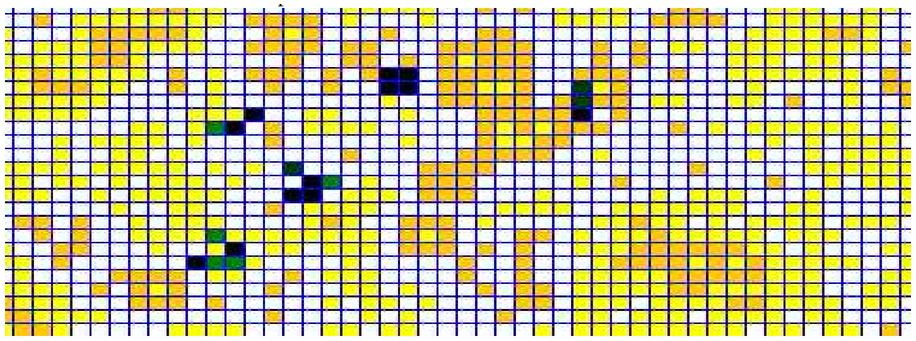
The next panel shows the state of the world a few time steps later, the shading indicates the synchronization state of the cell in the space, while the darker cells

[10] Disconnected components are of course at infinite distance, and so the temporal disparity between them can be arbitrarily large.

Asynchronous Game of Life
Initial State:



Progress of Gliders in Asynchronous Life. Note that the upper left hand glider is not recognizable as one due to small local temporal variation in its cells:



Further Progress of Gliders in Asynchronous Life. All their parts are nearly in the same spatial-temporal section; all three gliders are now recognizable again:

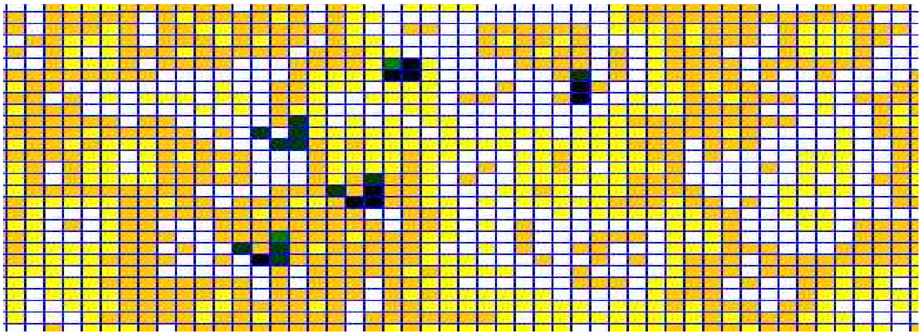


Fig. 3. Temporal Waves and Progression of 3 Gliders, with Box, and Blinker in Asynchronous Game of Life. Contiguous regions of the same shade are "temporal wavefronts" that represent the same moment in a spatio-temporal section giving the global state of the corresponding synchronous cellular automaton [55]. Shade is determined by value of modulo 3 counter at a given node. Neighbor nodes differ by at most one time unit with respect corresponding nodes in the synchronous model.

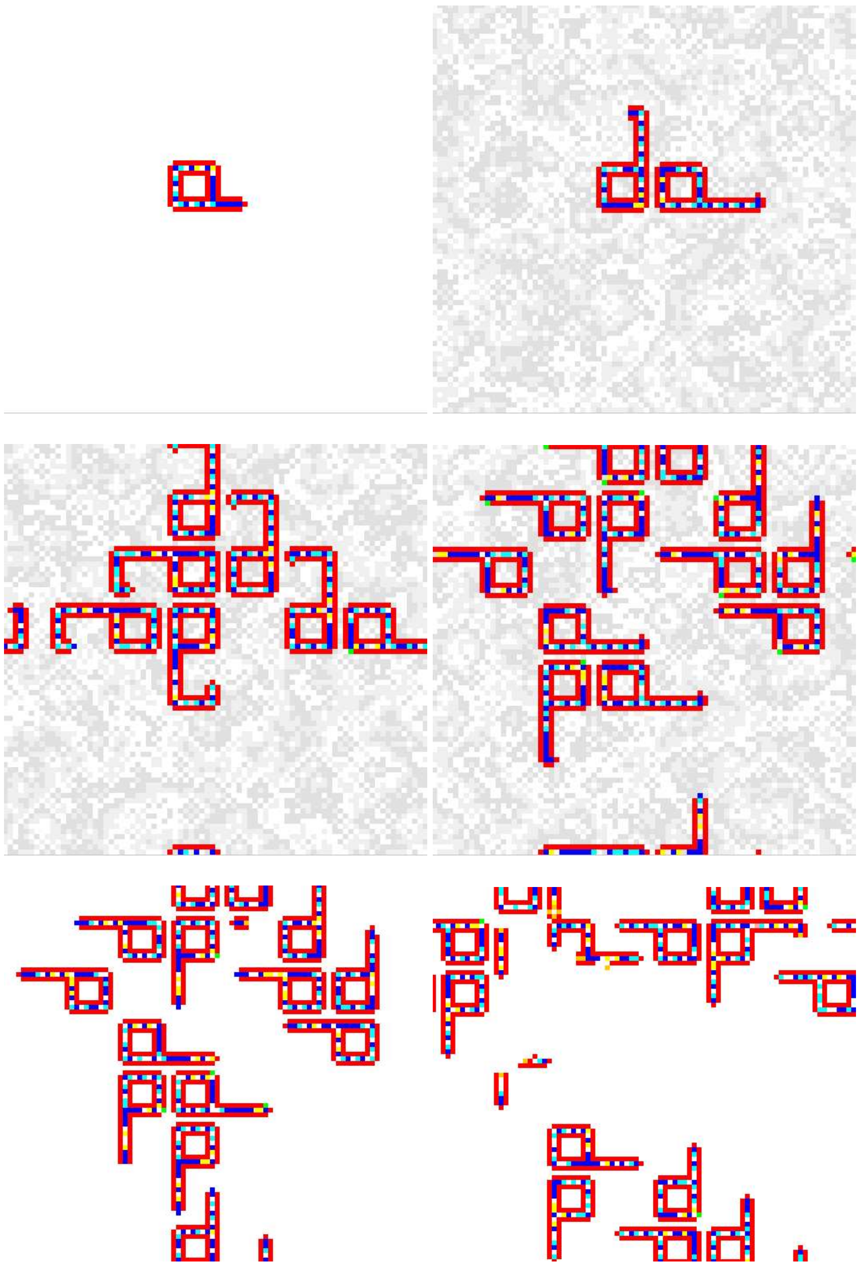


Fig. 4. Asynchronous Version of Sayama's Structurally Dissolvable Self-Reproducing Loop. Space is liberated by "programmed cell death" and can be reused by descendents of the original loop (6 snapshots of a single run; toroidal topology). Differences in shading (shown only in the first four panels) correspond to differences in the synchronization component of local state (cf. discussion of temporal waves).

of various shades are live cells in various stages of temporal synchronization. Contiguous cells of the same shade are in sync and reflect the same instant of time in the synchronous cellular automaton. The third panel down shows the state of the system a little later.

Asynchronous Universal Computation. The possibility of implementation of Conway's Game of Life in an asynchronous cellular automaton as illustrated here entails that universal computation is possible in a two-dimensional asynchronous cellular automata running the modified rules (see [81] for a lively exposition).

Of course, a Turing machine can be regarded as a synchronous 1-dimensional cellular automata where all state transitions are trivial except in the vicinity of the read-write head. Thus applying Corollary 1 to a universal Turing machine also yields the result.

6.2 Asynchronous Self-replicators

Applying the construction of the theorem to Langton's self-reproducing loop, and numerous self-reproducers including those of Byl, Reggia et al., Sayama and others mentioned above, we implemented the first asynchronous self-reproduction in cellular automata [55]. Figure 4 illustrates asynchronous replication of a structurally dissolvable loop capable of programmed cell death.

7 Minimal Evolvable Systems

To better understand evolvability we considered some open-ended evolutionary systems. Now we examine two (more or less) minimal evolvable systems to study how evolvability, and in particular variability, can arise.

7.1 Minimal Example 1: Asynchronous Evoloop

This example is due to Sayama-Nehaniv [77,53] by combining their techniques. Here a population of self-replicating loops in finite space is implemented (asynchronous cellular automata; physics: changes according to deterministic rules depending on local neighborhoods; asynchronous version of Sayama-Langton evoloop [53]).

Sayama [76], extending Langton's construction, introduced apoptosis. Apoptosis ("programmed cell death"), locally started, is triggered by local rules in response to stagnant or unexpected configurations (tending to indicate non-viability) generating a suicide signal, which propagates over to contiguous local automata that are non-quiescent [76]. This results in resource freeing and makes possible the turn over of generations required by evolution.

A further synchronous variant, evoloop, allows evolution in a cellular automata network to be realized [77]. By careful design of the update rules, ancestral self-reproducing loops are robust to some interactions (collisions) with others in space. They might either recover from a collision with another loop, undergo an apoptosis chain reaction, or survive in a changed form. The latter may or may not have the same circulating genome determining its construction. If a changed loop produces viable, reproductive offspring, then variation is inherited, so variability has been introduced in evolution.

Applying the asynchronous emulation theorem yields the first implemented example of an asynchronous cellular automata network with the capacity for Darwinian evolution (minimal evolvability), including heredity, variability, differential reproductive success, finite resources and turn-over of generations [53].

Over evolutionary time, loops of different sizes arise; smaller loops can replicate more quickly and are less likely to collide than large ones; the population generally evolves smaller and smaller loops until no further reduction in size is possible. See Figure 5.

7.2 Sources of Variability: Interaction

Interactions during lifetime are the major selective force but also the source of variation. There is only limited potential for variability (rotation of genetic core; loop and genome size).

7.3 Minimal Example 2: Cultural Evolution in Alissandrakis' Imitating Robotic Arms

Another instance of evolution occurs in human and non-human culture with the transmission of patterns of behavior (or *memes* [19]). Imitation broadly construed is the transmission mechanism for memes.[11]

7.4 Imitation, Social Learning, and Cultural Evolution

Learning behaviours from others, with cultural variations between populations that are not explainable simply due to differences in local ecological context, has been established not only for humans, but also in some other animal species, including cetaceans [72] and in chimpanzees [105].

Cultural evolution is based on transmitted patterns of behavior, and is exhibited by humans and some other animal species. Social learning, imitation, and/or instruction allow an organism to learn from the experience of others, which facilitates the accumulation of cultural practice and obviate much, often dangerous, trial-and-error individual learning. Social learning can also be combined with individual learning to exploit creative variability. In several realms (behaviors, technology artifacts, language) cultural evolution can be open-ended.

[11] There is a unsettled debate on whether a meme should be regarded as an unobservable (at least until now) pattern of information in the brain or as an observable expressed pattern of behaviour. The former seems more "genotypic" (as an informational pattern, but it probably is incapable of ever being directly copied from one individual to another) and the latter more "phenotypic" (as an effect of such a pattern).

Moreover, it is unclear what constitute an *individual* meme in the population dynamics of memetic evolution, or when two memes are "the same" (either at the neuronal or behavioral level).

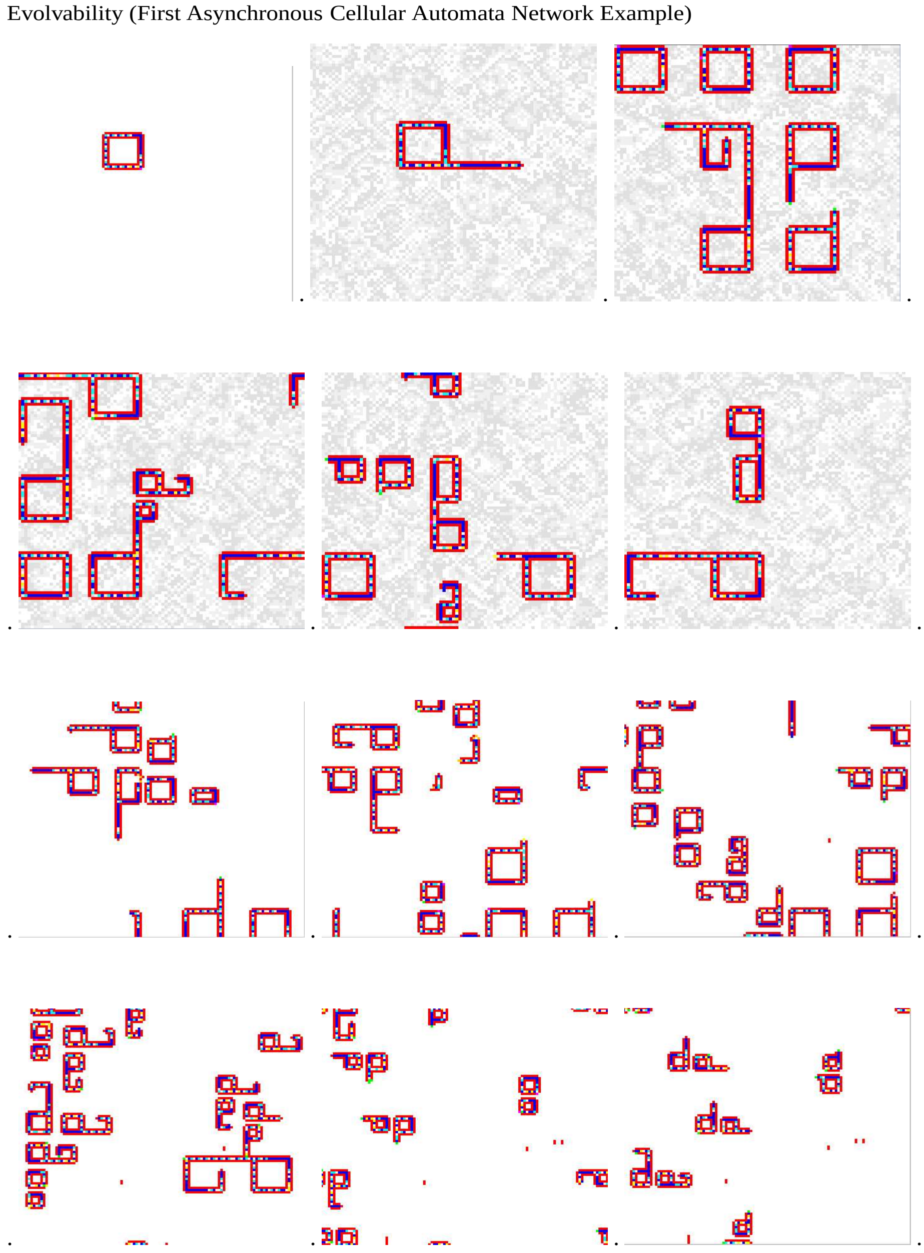
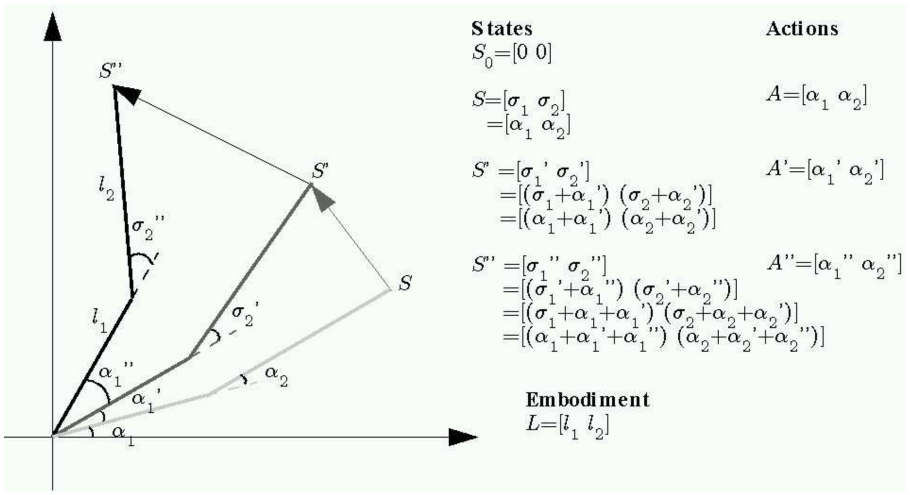

Evolvability (First Asynchronous Cellular Automata Network Example)

Fig. 5. Evolution in Asynchronous Cellular Automata: Asynchronous Version of Self-Reproducing Evoloop. (12 snapshots (not all from the same run); toroidal topologies). Heritable variability of characteristics of individuals (e.g. loop size) entails that this is an evolutionary system. Evolution leads to small, fast-replicating loops that are less likely to collide then larger ones. Temporal waves shading is shown in first six snapshots [53].



S''
l_2
σ_2''
S'
S
l_1
σ_2'
α_1''
α_1'
α_2
α_1
States
$S_0=[0\ 0]$
$S=[\sigma_1\ \sigma_2]$
$=[\alpha_1\ \alpha_2]$
$S'=[\sigma_1'\ \sigma_2']$
$=[(\sigma_1+\alpha_1')\ (\sigma_2+\alpha_2')]$
$=[(\alpha_1+\alpha_1')\ (\alpha_2+\alpha_2')]$
$S''=[\sigma_1''\ \sigma_2'']$
$=[(\sigma_1'+\alpha_1'')\ (\sigma_2'+\alpha_2'')]$
$=[(\sigma_1+\alpha_1+\alpha_1')\ (\sigma_2+\alpha_2+\alpha_2')]$
$=[(\alpha_1+\alpha_1'+\alpha_1'')\ (\alpha_2+\alpha_2'+\alpha_2'')]$
Actions
$A=[\alpha_1\ \alpha_2]$
$A'=[\alpha_1'\ \alpha_2']$
$A''=[\alpha_1''\ \alpha_2'']$
Embodiment
$L=[l_1\ l_2]$

Fig. 6. An example robotic arm imitating agent. A two-joint robotic arm, with segments of length ℓ_1 and ℓ_2, moving from *state* S_0 (arm completely outstretched along the horizontal axis) to state S to state S' to state S'', as it sequentially performs *actions* A, A', and A''. Effects of these actions (marked trails) are shown as the arrows that join the tips of the arm as it moves.

Cultural (memetic) evolution is possible in artificial societies and in the future might find application e.g. in factories populated by autonomous robots of various types who acquire and transmit skills and task knowledge through social learning. These robots could acquire skills and competencies by observing others (e.g. human demonstrators, or industrial robotic arms with different sizes, kinds and numbers of joints) and pass them to newcomer robots of as yet unknown type when they join the population.

A simple robotic population model illustrates this potentiality [3,4]: Simulated robotic arms are used, with differing lengths of segments, differing numbers of joints, but all with fixed base about which they can rotate (Figures 6 & 7).

The robot arms carry out behaviors in a two-dimensional workspace and engage in social learning via imitation. The agent embodiment can be described as the vector $L = [\ell_1, \ell_2, \ell_3 \cdots, \ell_n]$, where ℓ_i is the length of the i^{th} joint. Each robot builds "correspondence library" to imitate another, possibly dissimilar one (using various metrics of similarity to reinforce success). A robot arm observes another one (with possibly different embodiment) and attempts to match its behaviour (according to some metric such as posture, end-effector position, or angle changes at the joints). In turn, a third robot arm observes the imitator and attempts to imitate it (again using its own possibly different embodiment, using some metric), but does not observe the first robot, and so on. Behaviors can thus be culturally transmitted through a chain of robots.

The example illustrated in Fig. 8 demonstrates such (horizontal) transmission of a behavioral pattern via social learning in a chain of imitating agents.

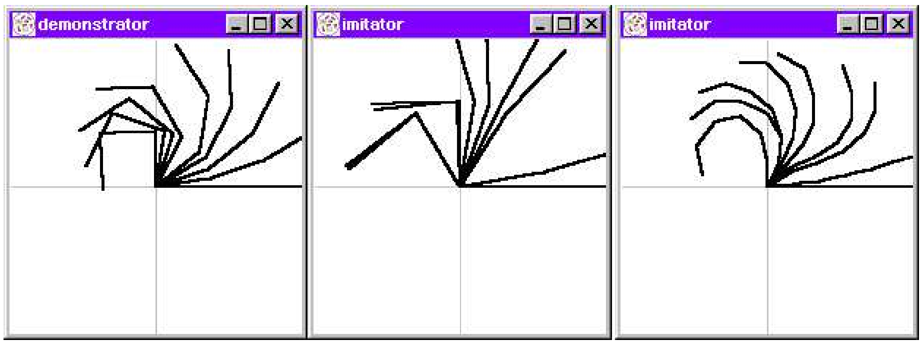


Fig. 7. Solving a correspondence problem for matched behaviour between different embodiments. A demonstrator behaviour consists of a model folding its 3 joints counter-clockwise (left). Imitation attempts to match the position of the end point are shown for a 2-joint imitator (center) and a 6-joint imitator (right).

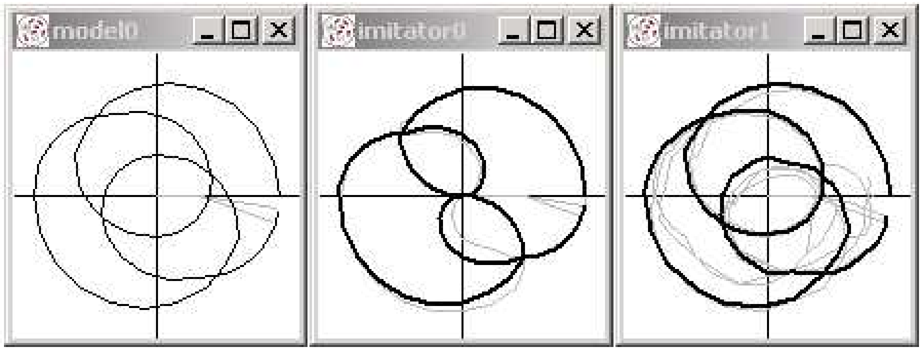


Fig. 8. An example of social transmission. Trail left by end-effector tips during behaviors by 3 robot arms are visualized. The original model `model0` ($L = [20, 20, 20]$) is shown to the left. In the middle, a two-joint `imitator0` ($L = [30, 30]$) acts also as a model for `imitator1` on the right ($L = [20, 20, 20]$)). Due to the different embodiment of the agent `imitator0`, the replication of the model pattern is similar, but not exact. `imitator1` has the same embodiment as the original model `model0` and, although indirectly transmitted, the resulting pattern is closer to that of the original model than is the behavior of the intermediate agent `imitator 0` used as a model by this second imitator [3].

The original model with three joints is shown in (Fig. 8, left). It is imitated by a two-joint robotic arm (Fig. 8, center), which in turn is imitated by another imitator (Fig. 8, right) with the same embodiment as the original model, but which only perceives the behavior of the two-joint agent. After transmission through the intermediary, the behavioral pattern that has been acquired by the second imitator in (Fig. 8, right) is quite similar to the original despite differences in embodiment in the chain of transmission. This example illustrates transmission of a behavioral pattern through a chain of robotic agents, despite differences in embodiment of agents involved. This simple example serves as proof of the concept that by using social learning and imitation, rudimentary cultural transmission with variability is possible among robots, even heterogeneous ones.

Evolutionary emergence of shared behavior and rudimentary ‘proto-culture’ in populations of robotic arms is discussed in [4]. Figure 9 shows imitators, with different embodiments arranged in a circle each learning by imitating its neighbor, and the resulting emergence of shared behavior.

Synchronization (via resetting to a fixed initial posture) before each demonstration has been shown to generally result in much faster and more accurate behavioral transmission [3].

7.5 Sources of Variability: Embodiment Differences

The behaviors of the arms are the selectable entity for a Darwinian evolutionary process: Imitation is the replication mechanism for these behaviors. Resources are finite since there are finitely many arms and each arm can only perform one behavior at a time.

Variability arises from several sources: (1) Errors in observation and noise in production of a behavior can introduce variability in a behaviour, which an observing robot might match, learn and pass on. (2) Embodiment differences may constrain what an imitator can do. For instance, a complex folding up by a six-segment arm could not be matched exactly angle-for-angle by a three-segment arm. Conversely, the six-segment arm imitating a two-segment arm might vary the position of various joints in many ways and still achieve a satisfactory imitative behavior; further down the chain of transmission an observer of this six-segment robot arm might acquire some aspects of its behavior not present in the original behavior of the two-segment robot.

The first source of variability is very closely analogous to mutation at the level of copying errors and is not particularly novel. The second source of variability, *differences in embodiment*, is unlike what we know from biological or evolutionary computation examples.

Replication in the robot arm example is based on *interaction* (like in prions – proteins that can inherit a conformation from interaction with a variant protein – but with vastly more variability).[12]

8 Developmental Genetic Regulatory Networks (DGRNs)

A particular paradigm from nature realizes all of the above properties discussed for differentiation, duplication and divergence: developmental genetic regulatory

[12] This is in contrast to evolving population of self-reproducing loops in example 1. There interaction provided the basis, not for replication, but for variability.

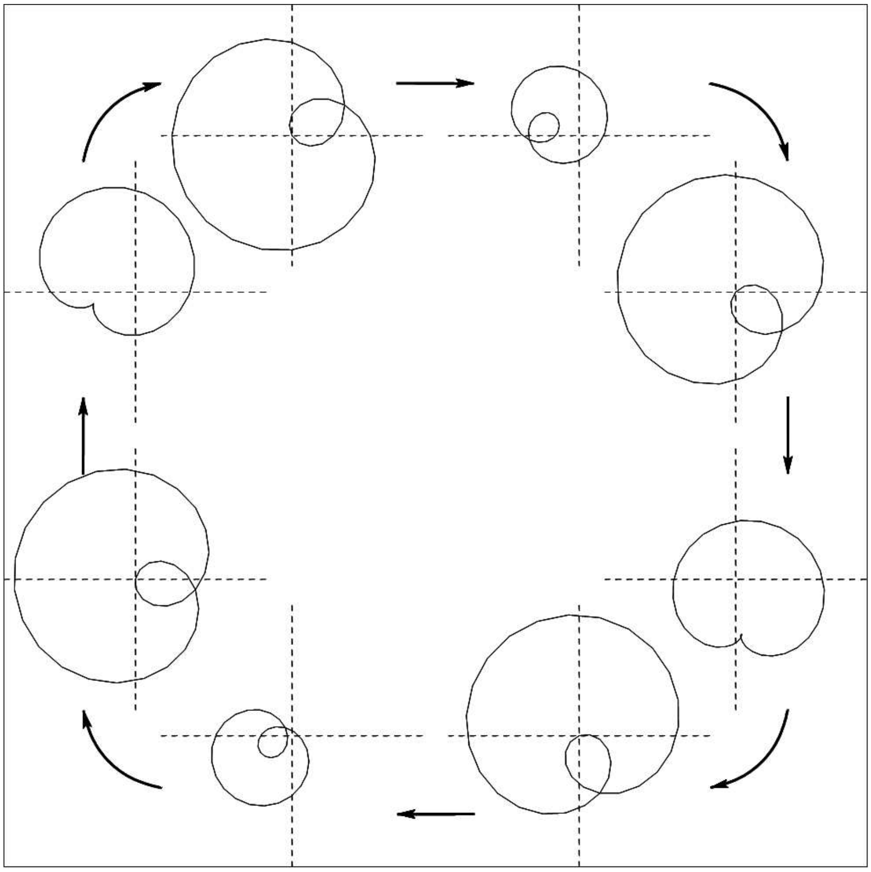


Fig. 9. Emergence of proto-culture among eight heterogeneous agents. Arm robots have alternating 2 and 4 joint embodiments (overall arm length remains constant). Starting with no initial "seed" model, and imitating each other clock-wise using a metric on actions, the figure shows an example of two stable repeated single-action variant behaviours emerging in the population: the agents with two joints move both joints anti-clockwise (by 10 degrees), while the agents with four joints freeze their first and the third joint, moving only the second and third joint. The different effect patterns shown result from the different states the agents are in when stable imitative behaviour is established.

networks (DGRNs). The most complex systems known to humankind are differentiated multicellular organisms. They may consist of e.g. on the order of between 10^{13} and 10^{14} cells in the human body; in multicellular organisms genetically identical cells differentiate, spending on the species, into between two or tens, or hundreds, of cell types [9] with each cell each capable of taking astronomically many states. The component cells themselves are each living entities each in itself already more complex than anything ever constructed by a human being. The organism maintains coherence as an individual while growing from a single cell into this huge number through asynchronous divisions, with a dynamically changing topology of interactivity between this changing number of cells including long, medium and short range interactions regulating patterning, global metabolism and the essential processes of life (see e.g. [106,6]).

In nature genetic regulatory networks with development/differentiation are constantly engaged in interaction (within each cell, with the local environment of the cell, or at organismal level with the external, ecological, and possibly social environment). This kind of incessant activity in the control systems within each cell and their coherent integrated activity has been called *universal responsiveness* by West-Eberhard and lies at the basis of phenotypic and developmental plasticity [104]. Selection for robustness in development (due, for example, to pressures for fitness to be heritable from parent to offspring) could have as a non-selected by-product the following properties which enhance evolvability:

(1) phenotypic variation becomes tolerated and possible;
(2) particular phenotypic variations becomes heritable, since similar genes in a similar environment yield similar development; later canalized, and
(3) developmental versatility leads to increased phenotypic variability (along the "right" dimensions of variation) serving as fodder for the next "rounds" of evolution.

Very little in Artificial Life has been achieved in the two modes of life involving self-reproducing autopoietic entities or symbiogenesis (see Appendix), but they can be approached on the foundations of current work. Genetic regulatory networks (GRNs) in cells are an essential component of how nature is able to grow developing, living systems [18]. GRNs are universally responsive control systems within biological cells. In multicellular organisms, GRNs are duplicated and diverge in functionality as organisms grow, in response to local conditions, the environment, and via signaling. They appear to provide essential properties for evolvability [1], and the continual, universal responsiveness and plasticity of living systems [104].

Operating continually in close connection with their environment through signaling channels, while actively maintaining internal dynamics, artificial GRNs easily allow for heritable digital genetic encoding, and provide a model analogous to that of a single cell (although presently without its replication capabilities). Unlike most other present-day computational models it is natural to apply them in an continually active and responsive mode [67]. Moreover, they exhibit very flexible evolvable, expressive dynamics similar to key biological regulatory phenomena useful in achieving a variety of control and computational dynamics [7,67]. The evolvability properties of GRNs and DGRNs are being analyzed mathematically as dynamical systems using techniques, e.g. of [62,61,60], in efforts to develop a predictive theory of their evolution and application in novel computation, as well as Artificial Life. The genetic regulatory network in a developing organism is duplicated in each cell, which carries its own differing state (in cytoplasm, structural and epigenetic marking). Each cell has the same genetic network and responds to local conditions. Multicellular living organisms use DGRNs to control for growth and differentiation, as well as for incessant active control while growing from a single zygote (or "seed") to adult by cell division. The desirable dynamical systems properties of GRNs might be combined with development to allow flexible, responsive control, continually coupled to the environment in organisms consisting of even astronomical numbers of different cells. Massively parallel distributed, adaptive, robust, fault-tolerant, self-repairing control and computation is a hallmark of DGRNs in living organisms but very unlike what we find in conventional software engineering and von Neumann computation, but the potential of DGRNs for novel computation and the simulation and synthesis of life is only now beginning to be explored.

Reaction-diffusion, cellular signaling and positional information could be set up using tools available and being developed in evolving multicellular systems as a natural method for computational morphogenesis and novel, developmental computation.

Development of NetBuilder, a test-bed for modeling the dynamics of multicellular genetic regulatory networks for biologists [78], is currently supported by a grant of the Wellcome Trust to Maria Schilstra and the author. It turns out that this platform can also be used to model artificial developmental genetic regulatory networks. Schilstra and Nehaniv [79] discuss the computational modelling of gene regulation in genetic regulatory networks, and current work is exploring the use of such computational networks to reverse engineer genetic regulatory control given gene expression data.

The evolutionary approach to understanding DGRNs is the most natural and would help characterize and evaluate aspects of their evolvability properties and developmental plasticity in different contexts.

Study of artificial versions of developmental genetic regulatory networks, comprising multicellular individuals, in evolving populations of such multicellular individuals is also a natural approach to addressing the essential questions of defining possible modes of life (see Appendix). Such evolving populations of DGRNs embodied in different environments may be rich enough to study (1) heritability of characters at higher levels, and (2) regulation of conflicts with the constituent cellular level (guided by some predicative theory from [47]), (3) emergence of self-maintenance at various levels, as well as (4) differentiation and modification of regulatory dynamics, and genetic, development and phenotypic plasticity.

Coupled with replication capability, evolving these artificial DGRNs (in software or in artificial proto-cells) could lead to systems showing more or all of the properties of life in its various modes. It would also be interesting to study (5) the induction of symbiogenesis in such systems, perhaps leading to artificial organelles and the degeneration of properties of life if capacity for independent maintenance of pattern integrity and replication is lost (mode 3). The genetic network in each cell of a differentiated multicellular organism is duplicated and divergence from its progenitors. Evolved differentiated multicellular organism possess a dynamic topology of interacting, developing genetic regulatory networks within their cells. These DGRNs have the following properties that could also be realized in implementations of artificial versions:

– Multiple copies of the same regulatory mechanism in similar units, with lineage structure
– Expressive and robust dynamical systems with parameters tunable by transcription factor (TF) binding strengths, concentrations, co-factors,etc.
– Layering of combinatorial logic on activation/inhibition of transcription ("biologic")
– Duplication-divergence via sensitivity of dynamics to epigenetic marking, development, environment, timing, cell-type, external signals

There are some open questions for GRNs and DGRNs:
(1) What is the degree of smoothness of their evolutionary dynamics?
(2) What is the relative importance of variability operators yielding regulatory changes vs. operators yielding gene product changes?
(3) What organizations of development yield what evolvability properties?
(4) What is the role of development and ecology in their evolvability (evo-devo-eco) regulatory changes?
(5) As biological DGRNs are naturally asynchronous, with no global clock coordinating their action, developmental and organismal time and timing must rely on local mechanisms to achieve coordination. The dynamic topology and asynchronous nature of DGRNs thus make them a promising test-bed for the studying the evolutionary dynamics and emergence of asynchronous temporal coordination.

9 Conclusion and Major Challenges

Evolution have been presented as a powerful and general class of stochastic algorithms. Response to interactivity (phenotypic plasticity) with environment/ others may be fundamental to evolvability. Interaction can play a selective and/or reproductive role in the capacity to evolve (as shown in two minimal examples exhibiting evolvability). Interaction can modulate duplication-divergence: Genetic regulatory networks, lifelong engagement, and differentiation/development appear to have important evolvability properties and consequences that deserve to be better studied. Interaction also plays a important role in the arising of multicellularity. Culture arises via social transmission of behavior, knowledge and skills, and is possible for constructed agents (e.g. robots on shop floor). In different example minimal evolutionary systems, interaction and embodiment can serve as a sources of variability. Developmental Genetic Regulatory Networks (DGRNs) are proposed as a paradigm for novel computation and the study of evolvability. Evolvability of autopoietic self-replicators, open-ended evolution, and feasible universal construction are open problems. Asynchrony is present in the most complex natural systems such as differentiated multicellular life, but synchronous automata network models can be made asynchronous using a uniform method by which emulation of behaviour of the synchronous system is mathematically guaranteed.

Acknowledgments. Many thanks for their encouragement to Andreas Albrecht and the organizers of SAGA'05, and for discussions on particular areas to Aris Alissandrakis (variability in social transmission; proto-culture in artifacts), Attila Egri-Nagy (evolvability of genotype-phenotype relation in populations of self-replicating programs), R. E. Michod (discussions of individuality, sex, and fitness), John L. Rhodes (algebra and time), and Günter P. Wagner, Lee Altenberg, the late Michael Conrad (evolvability), Hiroki Sayama and Barry McMullin (self-replication and evolvability), Maria Schilstra, Wolfgang Marwan, René te Boekhorst, Tom Ray, Hans Meinhardt, Julian F. Miller, and members of the EPSRC SEEDS research cluster. Except where stated, the ideas and results expressed in this paper, together with any shortcomings, are the author's own. Thanks to Algorithms and Adaptive Systems Research Groups, University of Hertfordshire, and the Engineering and Physical Sciences Research Council (EPSRC) for supporting the network on Evolvability in Biological and Software Systems. The biological modelling work involving NetBuilder is supported in part by an grant from the Wellcome Trust.

References

1. *BioSystems 69,* 2-3. Special issue on Evolvability, C. L. Nehaniv (ed.) (2003).
2. Adami, C., Ofria, C., and Collier, T. C. Evolution of biological complexity. *Proc. Natl. Acad. Sci. U.S.A. 97* (2000), 4463–4468.
3. Alissandrakis, A., Nehaniv, C. L., and Dautenhahn, K. Synchrony and perception in robotic imitation across embodiments. In *Proc. IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA '03)* (2003), pp. 923–930.
4. Alissandrakis, A., Nehaniv, C. L., and Dautenhahn, K. Towards robot cultures? - learning to imitate in a robotic arm test-bed with dissimilarly embodied agents. *Interaction Studies 5*, 1 (2004), 3–44.
5. Altenberg, L. The evolution of evolvability in genetic programming. In *Advances in Genetic Programming*, K. E. Kinnear, Ed. MIT Press, 1994, pp. 47–74.
6. Arthur, W. *The Origin of Animal Body Plans: A Study in Evolutionary Developmental Biology*, 1st paperback edition ed. Cambridge, 2000.
7. Banzhaf, W. Artificial regulatory networks and genetic programming. In *Genetic Programming - Theory and Applications.* Kluwer, 2003, pp. 43–61.
8. Berners-Lee, T. Evolvability. In *7th International WWW Conference, Brisbane, Australia* (15 April 1998). keynote address: slides on-line at `http://www.w3.org/Talks/1998/0415-Evolvability/`.
9. Bonner, J. T. *The Evolution of Complexity, by Means of Natural Selection.* Princeton University Press, 1988.
10. Burks, A. W. *Essays on Cellular Automata.* University of Illinois Press, Urbana, Illinois, 1970.
11. Buss, L. W. *The Evolution of Individuality.* Princeton University Press, 1987.
12. Byl, J. Self-reproduction in small cellular automata. *Physica D 34* (1989), 295–299.
13. Codd, E. F. *Cellular Automata.* Academic Press, New York, 1968.
14. Conrad, M. The geometry of evolution. *BioSystems 24*, 2 (1990), 61–81.

15. Crutchfield, J. P. Observing complexity and the complexity of observation. In *Inside versus Outside*, H. Atmanspacher, Ed. Springer, Berlin, 1993, pp. 235–272.
16. Darwin, C. *The Origin of Species by Means of Natural Selection*, 1st ed. John Murray, London, 1859.
17. Darwin, C., and Wallace, A. On the tendency of species to form varieties; and on the perpetuation of varieties and species by natural means of selection. *Journal of the Proceedings of the Linnean Society, Zoology 3* (20 August 1858), 45–62.
18. Davidson, E. H. *Genomic Regulatory Systems: Development and Evolution.* Academic Press, 2001.
19. Dawkins, R. *The Selfish Gene.* Oxford, 1976.
20. Dawkins, R. The evolution of evolvability. In *Artificial Life*, C. Langton, Ed. Addison Wesley, 1989.
21. Dömösi, P., and Nehaniv, C. L. *Algebraic Theory of Finite Automata Networks: An Introduction (SIAM Monographs on Discrete Mathematics and Applications, Vol. 11).* Society for Industrial and Applied Mathematics, Philadelphia, 2005.
22. Edmundson, A. C. *A Fuller Explanation: The Synergistic Geometry of R. Buckminster Fuller.* Birkhäuser, 1987.
23. Egri-Nagy, A., and Nehaniv, C. L. Evolvability of the genotype-phenotype relation in populations of self-replicating digital organisms in a tierra-like system. In *Proc. European Conference on Artificial Life (ECAL'03), September 14-17, 2003 Dortmund, Germany* (2003), vol. Springer Lecture Notes in Artificial Intelligence Vol. 2801, pp. 2380–247.
24. Fogel, L. J., Owen, A. J., and Walsh, M. J. *Artificial Intelligence Through Simulated Evolution.* John Wiley, 1966.
25. Goguen, J. Requirements engineering as the reconciliation of technical and social issues. In *Requirements Engineering: Social and Technical Issues*, M. Jirotka and J. Goguen, Eds. Academic Press, 1994, pp. 165–199.
26. Goguen, J. Formality and informality in requirements engineering. In *Proceedings, Fourth International Conference on Requirements Engineering* (April 1996), IEEE Computer Society, pp. 102–108.
27. Holland, J. *Adaptation in Natural and Artificial Systems.* MIT Press, 1975.
28. Kimura, M. *The neutral theory of molecular evolution.* Cambridge Univ. Press, 1983.
29. Kirschner, M., and Gerhart, J. Evolvability. *Proc. Natl. Acad. Sci. USA 95* (1998), 8420–8427.
30. Koza, J. R. Evolution of subsumption. In *Genetic Programming: On the Programming of Computers by Means of Natural Selection.* MIT Press, 1992, ch. 13.
31. Koza, J. R. *Genetic Programming II: The Next Generation.* MIT Press, 1994.
32. Laing, R. Automaton models of reproduction by self-inspection. *Journal of Theoretical Biology 66* (1977), 437–456.
33. Langton, C. G. Self-reproduction in cellular automata. *Physica D 10* (1984), 135–144.
34. Langton, C. G. Studying artificial life with cellular automata. *Physica D 22* (1986), 120–149.
35. Lehman, M. M. The role and impact of assumptions in software development, maintenance and evolution. In *IEEE International Workshop on Software Evolvability* (2005), IEEE Computer Society Press, p. (in press).
36. Leyser, O., and Day, S. *Mechanisms in Plant Development.* Blackwell, 2003.

37. Lohn, J. D. Self-replicating systems in cellular space models. In *Mathematical and Computational Biology: Computational Morphogenesis, Hierarchical Complexity, and Digital Evolution.* Vol. 26 in Lectures on Mathematics in the Life Sciences (Providence, Rhode Island, 1999), C. L. Nehaniv, Ed., American Mathematical Society, pp. 11–30.
38. Lohn, J. D., and Reggia, J. A. Automatic discovery of self replicating structures in cellular automata. *IEEE Transactions on Evolutionary Computation 1*, 3 (1997), 165–178.
39. Lynch, M., and Katju, V. The altered evolutionary trajectories of gene duplicates. *Trends in Genetics 11* (2004), 544–549.
40. Macias, N. J., and Durbeck, L. J. K. Adaptive methods for growing electronic circuits on an imperfect synthetic matrix. *BioSystems 73* (2004), 173–204.
41. Mange, D., Sanchez, E., Stauffer, A., Tempesti, G., Marchal, P., and Piguet, C. Embryonics: A new methodology for designing field-programmable gate arrays with self-repair and self-replicating properties. In *Readings in Hardware/Software Co-Design*, G. D. Micheli, R. Ernst, and W. Wolf, Eds. Morgan Kaufmann, San Francisco, 2002, pp. 643–655.
42. Margulis, L. *Symbiosis in Cell Evolution.* W. H. Freeman & Co., 1981.
43. Margulis, L., and Sagan, D. *Acquiring Genomes: A Theory of the Origins of Species.* Basic Books, 2002. Foreword by Ernst Mayr.
44. Maynard Smith, J. A darwinian view of symbiosis. In *Symbiosis as a Source of Evolutionary Innovation*, L. Margulis and R. Fester, Eds. MIT Press, 1991, pp. 26–39.
45. Maynard Smith, J., and Szathmáry, E. *The Major Transitions in Evolution.* W. H. Freeman, 1995.
46. Michod, R. E. *Eros and Evolution: A Natural Philosophy of Sex.* Addison-Wesley, 1995.
47. Michod, R. E. *Darwinian Dynamics: Evolutionary Transitions in Fitness and Individuality.* Princeton, 1999.
48. Michod, R. E., and Roze, D. Cooperation and conflict in the evolution of individuality. III. transitions in the unit of fitness. In *Mathematical and Computational Biology: Computational Morphogenesis, Hierarchical Complexity, and Digital Evolution.* Vol. 26 in Lectures on Mathematics in the Life Sciences (Providence, Rhode Island, 1999), American Mathematical Society, pp. 47–91.
49. Moore, E. F. Machine models of self-reproduction. In *Proceedings of the Fourteenth Symposium on Applied Mathematics* (1962, (Reprinted in A. W. Burks (ed.), 1968)), American Mathematical Society, pp. 17–33.
50. Morita, K., and Imai, K. A simple self-reproducing cellular automaton with shape-encoding mechanism. In *Artificial Life V* (1997), C. G. Langton and K. Shimohara, Eds., MIT Press, pp. 489–496.
51. Nakamura, K. Asynchronous cellular automata and their computational ability. *Systems, Computers, Controls 5*, 5 (1974), 58–66. translated from Japanese, *Denshi Tsushin Gakkai Ronbunshi*, **57-D**, No. 10, pp. 573–580, October 1974.
52. Nehaniv, C. L. Evolvability in biology, artifacts, and software systems. In *Proceedings of the Evolvability Workshop at the the Seventh International Conference on the Simulation and Synthesis of Living Systems (Artificial Life 7) 1-2 August 2000, Reed College, Portland, Oregon, USA* (2000). On-line at: `http://homepages.feis.herts.ac.uk/ nehaniv/al7ev/`.
53. Nehaniv, C. L. Evolution in asynchronous cellular automata. In *Artificial Life VIII: Proc. 8th Intl. Conf. on Artificial Life* (2002), R. K. Standish, M. A. Bedau, and H. A. Abbass, Eds., MIT Press, pp. 65–73.

54. Nehaniv, C. L. Internal constraints and ecology in evolution: A case study in tierra. In *Proceedings of the Fifth German Workshop on Artificial Life (GWAL V)* (2002), pp. 243–252.
55. Nehaniv, C. L. Self-reproduction in asynchronous cellular automata. In *Proc. 2002 NASA/DoD Conference on Evolvable Hardware (15-18 July 2002 – Alexandria, Virginia)* (2002), IEEE Computer Society Press, pp. 201–209.
56. Nehaniv, C. L. Asynchronous automata networks can emulate any synchronous automata network. *International Journal of Algebra & Computation 14*, 5 & 6 (2004), 719–739. Presented at International Workshop on Semigroups, Automata, and Formal Languages (June 2002 – Crema, Italy).
57. Nehaniv, C. L. The algebra of time. In *Proc. National Conf. of the Japan Society for Industrial and Applied Mathematics* (September 1993), pp. 127–128.
58. Nehaniv, C. L., and Dautenhahn, K. Self-replication and reproduction: Considerations and obstacles for rigorous definitions. In *Proceedings of the Third German Workshop on Artificial Life (GWAL III)* (1998), pp. 283–290.
59. Nehaniv, C. L., and Dautenhahn, K. *Artificial Life Fundamentals: The Simulation and Synthesis of Living Systems.* Springer Verlag, in prep.
60. Nehaniv, C. L., and Rhodes, J. L. On the manner in which biological complexity may grow. *Lectures on Mathematics in the Life Sciences 26* (1999), 93–102.
61. Nehaniv, C. L., and Rhodes, J. L. Axioms for biological complexity and mathematically rigorous measures of computational capacity: Applications to evolution of computation in cells. In *Proc. Computation in Cells: An EPSRC Emergent Computing Workshop (17-18 April 2000)* (2000), H. Bolouri and R. Paton, Eds., University of Hertfordshire, U.K., pp. 71–76.
62. Nehaniv, C. L., and Rhodes, J. L. The evolution and understanding of biological complexity from an algebraic perspective. *Artificial Life 6*, 1 (2000), 45–67.
63. Ohno, S. *Evolution by Gene Duplication.* Springer Verlag, 1970.
64. Orgel, L. E. Molecular replication. *Nature 358* (1992), 203–209.
65. Parnas, D. On the criteria to be used in decomposing systems into modules. *Communications of the Association for Computing Machinery 15*, 2 (1972), 1052–1058.
66. Pepper, J. W. The evolution of evolvability in genetic linkage patterns. *BioSystems 69(2-3).* Special issue on Evolvability, C. L. Nehaniv (ed.) (2003), 115–126.
67. Quick, T., Nehaniv, C. L., Dautenhahn, K., and Roberts, G. Evolving embodied genetic regulatory network-driven control systems. In *Proc. European Conference on Artificial Life (ECAL'03), Springer LNAI Vol. 2801* (2003), pp. 266–277.
68. Rasmussen, S., Chen, L., Deamer, D., Krakauer, D., Packard, N., Stadler, P., and Bedau, M. Transitions from nonliving to living matter. *Science 303* (2004), 963–965.
69. Ray, T. S. An approach to the synthesis of life. In *Artificial Life II*, F. Jones, Ed. Addison-Wesley, 1991, pp. 371–408.
70. Rechenberg, I. *Evolutionsstrategie - Optimierung technischer Systeme nach Prinzipien der biologischen Evolution.* Frommann-Holzboog, 1973.
71. Reggia, J. A., Armentrout, S., Chou, H. H., and Peng, Y. Simple systems that exhibit self-directed replication. *Science 259* (1993), 1282–1288.
72. Rendell, L., and Whitehead, H. Culture in whales and dolphins. *Behavioral and Brain Sciences 24*, 2 (2001), 309–382.
73. Ridley, M. *Evolution*, 2nd ed. Blackwell Science, 1996.
74. Sapp, J. *Evolution by Association.* Oxford, 1994.

75. Sayama, H. *Constructing Evolutionary Systems on a Simple Deterministic Cellular Automata Space.* PhD thesis, Department of Information Science, Grad uate School of Science, University of Tokyo, December 1998.
76. Sayama, H. Introduction of structural dissolution into langton's self-reproducing loop. In *Artificial Life VI: Proceedings of the Sixth International Conference on Artificial Life* (1998), C. Adami, R. K. Belew, H. Kitano, and C. E. Taylor, Eds., MIT Press, pp. 114–122. On-line material at: `http://necsi.org/postdocs/sayama/sdsr/`.
77. Sayama, H. A new structurally dissolvable self-reproducing loop evolving in a simple cellular automata space. *Artificial Life 5*, 4 (1999), 343–365.
78. Schilstra, M., and Bolouri, H. Logical modelling of developmental genetic regulatory networks with netbuilder. In *2nd Int. Conf. Systems Biology (ICSB 2001).* Omnipress, 2001.
79. Schilstra, M., and Nehaniv, C. L. The logic of genetic regulation. *submitted.*
80. Schwefel, H.-P. *Numerische Optimierung von Computer-Modellen mittels der Evolutionsstrategie.* Birkhäuser, 1977.
81. Sigmund, K. *Games of Life.* Penguin, 1995.
82. Sipper, M. The artificial self-replication page. `http://www.cs.bgu.ac.il/~sipper/selfrep/`.
83. Sommerville, I. *Software Engineering*, 5th ed. Addison-Wesley, 1996.
84. Szathmáry, E. Chemes, genes, memes: A classification of replicators. In *Mathematical and Computational Biology: Computational Morphogenesis, Hierarchical Complexity, and Digital Evolution.* `volume=26, in Lectures on Mathematics in the Life Sciences` (Providence, Rhode Island, 1999), C. L. Nehaniv, Ed., American Mathematical Society, pp. 1–10.
85. Szostak, J., Bartel, D., and Luisi, P. Synthesizing life. *Nature 409* (2001), 383–390.
86. Tempesti, G. A new self-reproducing cellular automaton capable of construction and computation. In *ECAL'95: Third European Conference on Artificial Life* (1995), vol. Lecture Notes in Computer Science 929, pp. 555–563.
87. Toffoli, T. Integration of phase-difference relations in asynchronous sequential networks. In *Automata, Languages, and Programming (Fifth Colloquium, Udine, July 1978),* Lecture Notes in Computer Science **62** (1978), G. Ausiello and C. Bohm, Eds., Springer Verlag, pp. 457–463.
88. Toffoli, T., and Margolus, N. *Cellular Automata Machines.* MIT Press, 1987.
89. Tyrrell, A. M., Sanchez, E., Floreano, D., Tempesti, G., Mange, D., Moreno, J. M., Rosenberg, J., and Villa, A. E. P. Poetic tissue: An integrated architecture for bio-inspired hardware. In *Evolvable Systems: From Biology to Hardware, 5th International Conference,ICES 2003, Trondheim, Norway, March 17-20, 2003,* Lecture Notes in Computer Science, Vol. 2606 (2003), Springer Verlag, pp. 129–140.
90. Van Belle, T., and Ackley, D. H. Code factoring and the evolution of evolvability. In *GECCO 2002: Proceedings of the Genetic and Evolutionary Computation Conference* (2002), Morgan Kaufmann, pp. 1383–1390.
91. Van Belle, T., and Ackley, D. H. Uniform subtree mutation. In *Proc. EuroGP-2002, the 5th European Conference on Genetic Programming* (2002), pp. 152–161.
92. van Nimwegen, E., Crutchfield, J. P., and Huynen, M. Neutral evolution of mutational robustness. *Proc. Natl. Acad. Sci. U.S.A. 96* (1999), 9716–9720.

93. van Nimwegen, E., Crutchfield, J. P., and Mitchell, M. Statistical dynamics of the royal road genetic algorithm. *Theoretical Computer Science 229,* Special Issue on Evolutionary Computation, A. E. Eiben and G. Rudolph (eds.) (1999).
94. Varela, F. J. *Principles of biological autonomy.* orth Holland, 1979.
95. Varela, F. J., Maturana, H. R., and Uribe, R. Autopoiesis: The organization of living systems. *BioSystems 5*, 4 (1974), 187–196.
96. Varshavsky, V. System time and system timing. In *Algebraic Engineering*, C. L. Nehaniv and M. Ito, Eds. World Scientific Press, 1999, pp. 38–57.
97. Vitányi, P. M. B. Sexually reproducing cellular automata. *Mathematical Biosciences 18* (1973), 23–54.
98. von Neumann, J. Probabilistic logics and the synthesis of reliable organisms from unreliable components. In *Automata Studies (Annals of Mathematics Studies, Number 34)*, C. E. Shannon and J. McCarthy, Eds. Princeton, 1956, pp. 43–98.
99. von Neumann, J. *Theory of Self-Reproducing Automata.* Edited and completed by A. W. Burks. University of Illinois Press, 1966.
100. Wagner, G. P., Ed. *The Character Concept in Evolutionary Biology.* Academic Press, 2001.
101. Wagner, G. P., and Altenberg, L. Complex adaptations and the evolution of evolvability. *Evolution 50*, 3 (1996), 967–976.
102. Watson, J. D., and Crick, F. H. C. Molecular structure of nucleic acids. *Nature 171* (1953), 737–738.
103. Wernick, P., and Lehman, M. M. Software process white box modelling for feast/1. *Journal of Systems and Software 46*, 2-3 (1999), 193–201.
104. West-Eberhard, M. *Developmental Plasticity and Evolution.* Oxford University Press, 2003.
105. Whiten, A., Goodall, J., McGrew, W. C., Nishida, T., Reynolds, V., Sugiyama, Y., Tutin, C. E. G., Wrangham, R. W., and Boesch, C. Culture in chimpanzees. *Nature 399* (1999), 682–685.
106. Wolpert, L. *The Triumph of the Embryo.* Oxford University Press, 1991.
107. Wright, S. The role of mutation, inbreeding, crossbreeding, and selection in evolution. *Proceedings of the Sixth International Congress on Genetics 1* (1932), 356–366.

Appendix A: Are There Degrees of Life? – Converting Resources into Persistence and Progeny

The preceding considerations of evolutionary processes as stochastic algorithms that may be realize in many substrates lead naturally to the question of when the individuals in an evolutionary process should be called *alive.* We consider in this appendix what heritability, self-maintenance, symbiosis and responsive dynamics via genetic regulatory networks can tell us about *possible modes of life*, in whatever medium it might find realization.

A.1 Re-thinking Life

The last two centuries have yielded profound scientific advances in our understanding of the particular nature of life on earth: the nature of the cell, the

Darwinian theory of evolution, its synthesis with Mendelian genetics, and later with biochemistry, the basis of hereditary in the substance of long chains of nucleotides, the (nearly) universal genetic code, the details of protein biosynthesis (see e.g. [73]), increased understanding of the dynamics of development in animals [6] and plants [36], and of evolvability [1,100], as well as ever more detailed understanding of genetic regulatory systems and how they operate in the single-celled and differentiated multicellular organisms (e.g. [18]), among other advances. Recent advances in the construction of 'proto-cells' with various properties of living systems bring us closer to new and very minimal instantiations of life – of various kinds (artificially constructed or parred-down cells) [68,85]. We may possibly find other examples of life elsewhere in the solar system or universe, but also in media other than those of organic biochemistry. Researchers in Artificial Life have sought to capture and reproduce an underlying an "logic of life" in software running on electronic computers (e.g. [69]), or in other media. These developments are leading us to reconsider the notion of living organism: How are we to know whether we are looking at new kinds of life?

A.2 Some Subtle Properties of Life

Here we collect fundamental phenomena found in organic life on earth that have tended not to receive emphasis. We formulate them with a view toward achieving a more universal understanding what life is.

Replicators in Context. The heritable material in replicators does not fully specify the constraints of the environment in which they are capable of replication. Indeed, there is no way for evolution directly to distinguish between genetic, environmental, physical, geometric, energetic constraints, or incidental or universal properties of an evolving population's environment. Catalyzing the ambient dynamics, no matter how, that promote their own replication is enough for the most primitive replicators. Other capabilities come later in the evolution of biological complexity [62].

Pattern Integrity. Life is an organizational pattern that "holds its shape". Although the particular material that makes up an organism in the course of its life may be changing, the individual persists. Just as a wave on the ocean, a dynamical event occurring on a substrate of water molecules, and a slipknot, which be passed down from a string to a rope to a necktie, depend on *configuration* and not particular material for their existence, similarly life relies on persistence of an organizational pattern that is structurally stable enough to perpetuate its existence in a medium (*pattern integrity*). (cf. discussion in [22].)

Enough stability must be present in the structure of the dynamics for this to occur for us to be able to speak about an *individual* of any kind.

In replication, during a transition when a parent is giving rise to its offspring in the dynamics of the medium, the parent is active in the establishment of the offspring's pattern integrity, as is the new individual itself, in changing degrees (*self-production*). Replicators whose dynamics favors conditions that happen to tend to increase their reproductive success through promoting processes of homeostasis and self-repair will increase under the action of evolution. Beyond basic pattern integrity – which is necessarily present already with simple replicators, or for that matter in any persistent individuals – life has also achieved more sophisticated types of pattern integrity in the forms of *self-maintenance*, *homeostasis*, *self-repair* and *regeneration* within an individual.

Robustness, Plasticity and Incessant Responsiveness. Despite perturbation and variability at the level of inheritance (such as mutations and sex), living organisms are often robust and can grow, develop, persist, and thrive successfully (*robustness to genetic variability* [14]). Moreover, despite the lack of any full 'specification' of organisms by their genetic material and despite perturbations to the environment, they are often remarkably able to generate appropriate and adaptive forms and responses to various, changing environmental situations (*phenotypic and developmental plasticity*) [104]. This relies on the fact that organisms continually engage with their local environment, showing incessant adaptive activity, internally and externally (*universal responsiveness* [104]).

Evolvable Dynamical Systems. *Evolvability* (cf. [1]) is defined above as the particular capacity of an evolving population to generate adaptive heritable genotypic and phenotypic variation. Evolvability depends on many details of the genetic system and can be radically different in different instances of evolution. To achieve self-maintenance, plasticity, and continual responsiveness a sufficiently evolvable substrate is necessary. Sufficiently powerful dynamical systems bases (such as the physics, chemistry and genetic system enjoyed by life on earth) are necessary to support evolvability. Moreover, the evolvability of a system can change (e.g. with the advent of a genetic code).

Achieving Heritability and Transitions in Individuality. Things fall apart (in the physical and many artificial worlds). Quick reproduction based on *digital templates* is one of the ways that life uses to help it to triumph over entropy. With a discrete basis of heritability (such as provided by a limited alphabet of nucleotides and codons), copies can be perfect. Engineering shows that rebuilding is often more efficient than repair.

Simple replicators might have no regeneration and repair capabilities (and these might be quite complex to specify in a heritable manner). Without these, replication rather than repair is eventually the only option in the face of the tendency of things to crumble. With increased degrees of self-repair and regeneration, persistence rather than immediate production of offspring becomes an option. Too much persistence however would stop evolution, unless variability is somehow continually generated.

The simplest replicators have offspring like themselves in a similar environment. Fitness of these is likely to be similar to that of the parents (until resources are exhausted, or conditions change): When offspring are produced, they are likely to be successful in the environment of their parent if they are similar to it – that is, if their capabilities of promoting their own persistence and reproductive success in that environment are similar to those of their parents. Heritability of fitness is achieved simply for accurate replicators in a stable environment: if the new one is to be viable, a copy of the original is likely to be a good choice.

Ensuring faithfulness of copies is one way to ensure heritability of fitness. Digital genetic systems help achieve this.

New higher-level, e.g. multicellular, replicators whose components are themselves alive need to solve the problems of heritability of fitness anew, together with those of sex, and self-repair. At higher levels, for life to exist, a transition to individuality must reinvent replication, resulting in heritability of fitness at that level; also reinvented are self-maintenance for the higher-level individual, and sex (receiving or exchanging of heritable material from others) [47]. It is also faced with new problems: suppression of freedom at lower levels and harnessing lower level units into cooperatives that contribute to the fitness of the higher level individual requires a balance of the tendency of replicators at lower levels to pursue their own individual reproductive success at the expense of that of the higher level individual [11,47]. In differentiated multicellularity, constituent cells may pursue their own replication at the higher-level's expense (cancer); in social insects some members of non-reproductive labor castes may 'defect' and become reproductive, at the expense of the colony's integrity.

Higher-level individuals develop from a single (or a small number of) constituents and harness division of labor and the differentiation of their constituents (e.g. cells in a body; insect castes). Epigenetic inheritance (via state and marking) in the population of constituents comprising them makes this possible.

A.3 Degrees and Modes of Being Alive

Biologists asked to define life have not agreed on a universally accepted definition, but instead tend to produce lists of properties. One candidate definition [59], applicable to life-as-it-could-be as well as life-as-we-know-it, is:
A *living organism* is an individual entity that

1. transforms resources into persistence (pattern integrity) and progeny (i.e. to achieve reproduction),
2. results from reproduction (or is able to reproduce),
3. results from an evolutionary process in a population (involving heritable variation, differential reproductive success, finite resources),
4. produces itself in the context of its environment (self-production, growth and possibly development),
5. engages in self-maintenance,
6. uses signaling and interaction (internally and externally), and
7. modifies and is modified by its environment (embodiment and plasticity).

The more of these properties from such lists a system exhibits, the more justified we feel in saying that it is alive. For example, viruses or 'digital organisms' (such as Tierrans [69]) exhibit some but not all the properties of life (see also below). This suggests that there may be different degrees to which it makes sense to call something alive, i.e. different *degrees of life* depending on the degree to which each of requirements in the biologist's list are satisfied [59].

Modes of Life with Different Degrees of Aliveness. In light of the advances in Biology and Artificial Life mentioned above, it will be argued here that the evolution of life from a single ancestor or ancestral population may involve different modes which account for some of the differences in the degree of aliveness. Organic evolution on earth (and in some cases in artificial systems at the first two levels) has apparently produce different *modes of life* (which comprise a evolutionary continuum). These modes show marked qualitative differences in dimensions present among the requirements for life, but are related to each other by evolutionary contingencies:

- Mode 0: **Replicators (no or very limited variation in heritability).** Prions, crystal growth, and cellular automata replicators show many properties of life. They engage in no self-maintenance. Evolution does not occur (with some recent exceptions in cellular automata). At the level of the individual, they do not show genetic, developmental, or phenotypic plasticity.
- Mode 1: **Replicators in Evolving Populations.** Viruses, Tierrans (and to a lesser degree transposons) exist in populations showing heritability, variability, and differential reproductive success – the requirements for evolution. This leads some researchers to assert that they are examples of minimal living systems. They exhibit no homeostatic control, or capabilities of regeneration and self-repair.
- Mode 2: **Self-Maintaining Organisms (showing various degrees of autonomous responsive dynamics and plasticity).** These extend mode 1 capacities and include all the uncontroversial cases of living organisms. [See list of *living organism* dimensional properties above. This class might have identifiable occurring submodes, e.g. some self-maintenance and regenerative capability but little or no phenotypic plasticity. Some examples may be naturally occurring, but others might soon be produced by constructive methods en route to more sophisticated artificial life forms, e.g. proto-cells.]
- Mode 3: **Degenerated Life (arising, e.g., due to Symbiogenesis or Multicellularity).** These are evolving replicators whose ancestors were of another mode but which are in the process of losing (or have lost) most of their individuality. Examples associated with evolutionary transitions: in the RNA world (or other early life scenarios), early replicators → membrane-bound genes/'chromosomes'; in the evolution of eukaryotes, free-living prokaryotic ancestors of organelles → mitochondria, chloroplasts. Mitochondria and chloroplasts are organelles whose ancestors where free-living endosymbionts of ancestors of eukaryotic cells but which are not longer capable of replication without the machinery of the cells and some of the heritable information in the cell's nuclear DNA [42,74].

Transitions Between Modes of Life. The advent of new higher-level replicators such as multicellular entities (possibly with differentiation of constituents into a cooperative division of labor) leads us to ask the question again. Are these replicators (rather than their constituent members) alive? To what degree do they show the properties required of living systems such as being members of an evolving population? self-maintenance, heritability of fitness, and sophisticated responsive dynamics?

If multicellular plants, animals and fungi, or colonies of social insects are self-replicating higher level individuals satisfying the properties of life, how are the properties of life (in the list above) achieved by the higher level entity? Are we dealing with mere replicators, evolution, or evolution of self-maintaining dynamic entities?

With symbiogenesis and increasing dependency on partners, loss of individuality may result (e.g. in the evolution of cellular organelles having endosymbiotic origin, mitochondria and chloroplasts). For most definitions of life, evolution must act on individuals in a population, if individuality is lost, then evolution at the former level of individuality – and hence life at that level – becomes less distinct (mode 3), and eventually it may not be appropriate to speak of life.

The extreme modes are less alive than the middle ones. Transitions between the above modes of life (in the direction of the list) are however natural and may be favored by natural selection (as with the evolution of mitochondria). Even though the last mode involves degeneration of life properties at one level, it is only known to occur in the transitional genesis of higher-level units of selection. Note that it also need not occur (even with new, albeit loose units of selection); e.g. the cells in differentiated multicellular organisms or symbiotic partners in lichens that can also live independently are both mode 2. In the opposite direction, examples which may or may not have arisen as renegade replicators that were originally components of a larger unit of life include viruses and transposons (transitions to mode 1 from higher modes).

New Computation Paradigm for Modular Exponentiation Using a Graph Model[⋆]

Chi Seong Park[1], Mun-Kyu Lee[2], and Dong Kyue Kim[1]

[1] Pusan National University, Busan 609-735, Korea
{cspark, dkkim}@islab.ce.pusan.ac.kr
[2] Inha University, Incheon 402-751, Korea
mklee@inha.ac.kr

Abstract. Modular exponentiation is to compute $x^E \bmod N$ for positive integers x, E, and N. It is an essential operation for various public-key cryptographic algorithms such as RSA, ElGamal and DSA, and it is crucial to develop fast modular exponentiation methods for efficient implementation of the above algorithms. To accelerate modular exponentiation, one can either speed up each multiplication or reduce the number of required multiplications. We focus on the latter.

In this paper, we propose a general model to describe the behavior of modular exponentiation in terms of a graph. First, we show that the problem of finding the minimum number of multiplications for a modular exponentiation is equivalent to finding a shortest path in its corresponding graph. The previously known exponentiation algorithms including the binary method, the M-ary method and the sliding window method can be represented as a specific instance of our model. Next, we present a general method to reduce the number of required multiplications by modifying the pre-computation table which is used for the sliding window method. According to our experimental results, the new method significantly reduces the number of multiplications, especially in the cases that the exponent E has a high Hamming weight.

Keywords: Exponentiation, Modular Exponentiation, Graph Model, Window Method.

1 Introduction

Modular exponentiation is to compute $x^E \bmod N$ for positive integers x, E, and N. It is an essential operation for various public-key cryptographic algorithms such as RSA [1], ElGamal [2], and DSA [3]. To guarantee the security of these applications, very large exponents E should be used. Therefore, it is crucial to develop fast modular exponentiation methods that are practical for large exponents. Because a modular exponentiation is composed of repeated multiplications, the study on the efficiency of modular exponentiation can be divided

⋆ This work was supported by the Regional Research Centers Program(Research Center for Logistics Information Technology), granted by the Korean Ministry of Education & Human Resources Development. Contact Author: dkkim1@pusan.ac.kr.

into two categories: to speed up multiplication itself, and to reduce the number of required multiplications. In this paper, we focus on the latter.

There has been an extensive research to reduce the number of required multiplications for modular exponentiation, e.g., the binary method [4], the M-ary method [4], the addition chain method [5,6], the sliding window method [7], and so forth. It is known that the sliding window method shows the best performance from the practical viewpoint.

In this paper, we propose a general framework to describe the behavior of modular exponentiation and to find an optimized computation sequence. Our work is based on the idea that an exponentiation can be represented by a graph. Specifically, our contributions are as follows:

- We present a new interpretation of modular exponentiation, where each bit in the binary representation of an exponent E is modeled as a vertex in a graph and each multiplication as an edge. Then we show that the problem of finding the minimum number of multiplications for an instance of modular exponentiation is equivalent to finding a shortest path in its graph representation. Actually, the previously known algorithms such as the binary method, the M-ary method, and the sliding window method can be translated into specific paths over the graph constructed as described above.
- We present a general method to reduce the number of required multiplications. Our technique is to properly expand the pre-computation table in such a direction that the overall number of multiplications is reduced. Our experimental results show that the new method reduces the number of required multiplications by 5.38, 8.87 and 15.18, when it is applied to the exponentiation with random exponents $E \approx 2^{512}$, $E \approx 2^{1024}$ and $E \approx 2^{2048}$, respectively. Moreover, the improvements are remarkable, i.e., up to about 33, 54 and 95, respectively, when the new method is applied to the special exponents with high Hamming weights.

The rest of this paper is organized as follows. Section 2 gives previous works to reduce the number of required multiplications. In Section 3, we present a general framework to describe the behavior of modular exponentiation using a graph. In Section 4, we suggest two techniques to expand a pre-computation table which we call a *block-table*, so that the number of multiplications is reduced. Section 5 shows the performance of our method in various practical settings.

2 Related Works

Reducing the number of required multiplications means finding a shorter sequence of multiplications in $x^E \bmod N$. In this section, we review existing algorithms to find such an efficient sequence of multiplications.

2.1 Binary Method

This fundamental method considers the exponent E as a binary number E_B, and then repeats squarings and multiplications according to the individual bits

Algorithm 1 Binary method

```
1. $A \leftarrow x$.
2. for $i$ from $n-2$ to 0 do
3.     $A \leftarrow A^2$.
4.     if $(e_i = 1)$ then $A \leftarrow A \cdot x$.
5. od
6. output $A$.
```

Algorithm 2 M-ary method $(M = 2^d)$

```
1. Compute and store $x^2, x^3, \ldots, x^{2^d-1}$.
2. $A \leftarrow x^{f_{l-1}}$.
3. for $i$ from $l-2$ to 0 do
4.     $A \leftarrow A^{2^d}$.
5.     if $(f_i \neq 0)$ then $A \leftarrow A \cdot x^{f_i}$.
6. od
7. output $A$.
```

in E_B. Algorithm 1 is the binary method, where the exponent is represented as $E_B = (e_{n-1}, \ldots, e_0)$ and $n = \lceil \lg(E+1) \rceil$. We can easily see that the number of required multiplications (and squarings) in the binary method is $k(n-1)$, where $1 \leq k \leq 2$.

2.2 *M*-Ary Method

The M-ary method is a generalization of the binary method. Instead of using radix 2, the M-ary method uses radix M to represent the exponent E, i.e., $E = \sum_{i=0}^{l-1} f_i M^i$, where $l = \lceil \log_M(E+1) \rceil$ and $0 \leq f_i \leq M-1$. Generally, $M = 2^d$ is used for some positive integer d, i.e., the binary representation E_B is partitioned into the blocks of length d. (See Algorithm 2.) Although there is an overhead to construct a pre-computation table for $x^2, x^3, \ldots, x^{2^d-1}$, the number of overall multiplications is reduced because several multiplications for each block is changed into only one multiplication by one of the table elements. Note that the performance of this method depends on the proper choice of the block size d.

2.3 Sliding Window Method

The sliding window method of window size d is a modification of the 2^d-ary method. There are two kinds of sliding window method, i.e., the CLNW (Constant Length Non-zero Window) method and the VLNW (Variable Length Non-zero Window) method. In the CLNW method, we require that every non-zero block should end with a '1' and the length of every non-zero block is fixed as d. By this modification, we can reduce the size of pre-computation table to a half, since only odd powers of x is to be stored. Also, we can expect the number of multiplications should be reduced slightly, since we can partition the exponent so that variable-length zero blocks may exist between adjacent non-zero blocks. On the other hand, the VLNW method requires that both of the starting and ending bits in a non-zero block should be ‘1’, and the length of each non-zero block should be less than or equal to d. We remark that for the VLNW method, another parameter q should be chosen properly. While we will not go into its details, it is known that the optimal parameters of VLNW method are $4 \le d \le 8$, $1 \le q \le 3$ for $128 \le n \le 2048$.

3 General Graph Model for Modular Exponentiation

In this section, we present a new model to describe the behavior of modular exponentiation, which is called the *general graph model.* This model reduces the problem of modular exponentiation into the shortest path problem in a graph, and it can be effectively used to find a short sequence of multiplications. The reason why we use the term ‘general’ is that the existing algorithms including the binary method, the M-ary method and the sliding window method can be represented by our model.

The general graph model employs a block-table of size $2^d/2$ which is equal to that of the sliding window method with a window size d. For $x^E \bmod N$, we define a directed graph $G = (V, E_G)$ as follows. First, the node set V of G is defined as :

$$V = \{v_i \text{ for } 0 \le i \le n-1 : v_i \text{ corresponds to } e_i \text{ in } E_B\} \cup \{\text{source, sink}\},$$

where E_B is the binary representation of exponent E, and $n = \lceil \lg(E+1) \rceil$. Hence the graph has n nodes between the source node and the sink node. Fig. 1 shows the nodes of a graph corresponding to $E_B = 11010110111011111110111110100111_2$, where the left-most node indicates the source node, and the right-most node indicates the sink node. Note that the node next to the source node is v_{n-1} which corresponds to e_{n-1}, and the node next to the sink node is v_0 which corresponds to e_0.

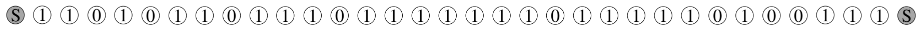


Fig. 1. Nodes of a graph in the general graph model

On the other hand, the edge set E_G of G is defined as

$$E_G = \{(v_i, v_j) : 0 \le i - j \le d \text{ and } v_i, v_j \in V\},$$

where an edge from v_i to v_j corresponds to the computation of $x^{(e_{n-1}, e_{n-2}, \ldots e_i, \ldots, e_j)} \bmod N$ using the value $x^{(e_{n-1}, e_{n-2}, \ldots, e_i)} \bmod N$. We also define the weight of the edge (v_i, v_j), which is denoted by $w(v_i, v_j)$, as the number of required multiplications (including squarings) for this computation.

Now we show how to decide the weight value of an edge. Recall that

$$x^{(e_{n-1},e_{n-2},\dots e_i,\dots,e_j)} \equiv (x^{(e_{n-1},e_{n-2},\dots,e_i)})^{2^{i-j}} \times x^{(e_{i-1},\dots,e_j)} (\mathrm{mod}\ N). \quad (1)$$

Then we have the following three cases.

1. If $e_j = 1$, then $x^{(e_{i-1},\dots,e_j)} \bmod N$ is in the pre-computed block-table. Thus we can get $x^{(e_{n-1},e_{n-2},\dots e_i,\dots,e_j)} \bmod N$ from $x^{(e_{n-1},e_{n-2},\dots,e_i)} \bmod N$ by $(i-j)$ squarings and one multiplication. Therefore $w(v_i, v_j) = i - j + 1$.
2. If $(e_{i-1}, \dots, e_j) = (0, \dots, 0)$, we need only squarings. Thus $w(v_i, v_j) = i - j$.
3. If $e_j = 0$ but not all of the bits $e_{i-1}, \dots, e_j$ are zeros, then the situation is similar to the first case. The only difference is that the multiplication does not occur after squarings, but it occurs between squarings. Therefore $w(v_i, v_j) = i - j + 1$.

Fig. 2 shows a graph corresponding to E_B=$110101101110111111011111010011_2$ and $d = 5$, omitting the weight values.

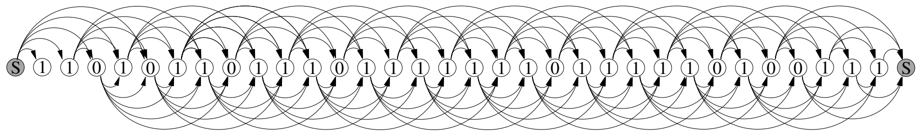
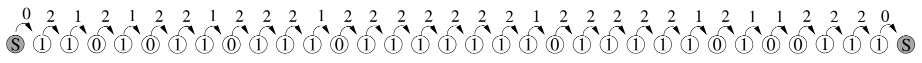
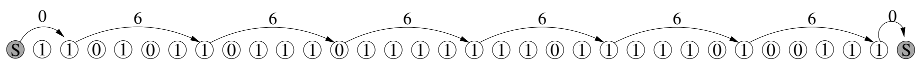


Fig. 2. General graph model

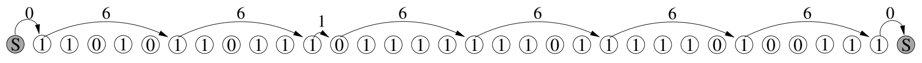
- binary method: weight of the path = 54



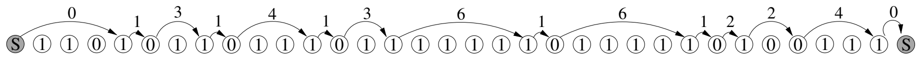
- M-ary method($d = 5$): weight of the path = 36



- CLNW method($d = 5$): weight of the path = 37



- VLNW method($d = 5$, $q = 1$): weight of the path = 35



- shortest path($d = 5$): weight of the path = 34

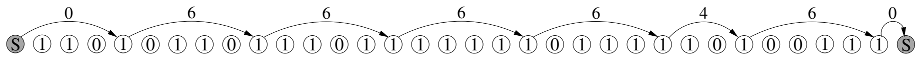


Fig. 3. Paths corresponding to various algorithms

There are many paths from the source to the sink in G, and each path represents a distinct sequence of multiplications. The number of required multiplications for a specific path is (the cost to construct the block-table of size 2^{d-1}) + (the sum of weights in the path). Therefore, G can be viewed as a general model to describe computing sequences for modular exponentiation based on the block partitioning, such as the binary method, the M-ary method, or the sliding window method. Fig. 3 shows the paths corresponding to the existing algorithms for $E_B = 11010110111011111110111110100111_2$ and $d = 5$.

Because we can represent a specific exponentiation algorithm as a path from the source to the sink in our general graph model, it is straightforward to see that we can use one of the well-known shortest path algorithms [8] to find the minimum weight path in G, which corresponds to the optimal computation sequence under the given parameters. The last row of Fig. 3 shows the result of this task.

4 Methods for Constructing the Block-Table

In this section, we describe a general way to reduce the number of required multiplications. Note that since a block-table, i.e., a pre-computation table, determines the weight value of each edge in our model, it is very important to construct a block-table properly. Our basic block-table is the same as that of the sliding window method. In this section, we propose two methods to modify this initial table and get an optimized one.

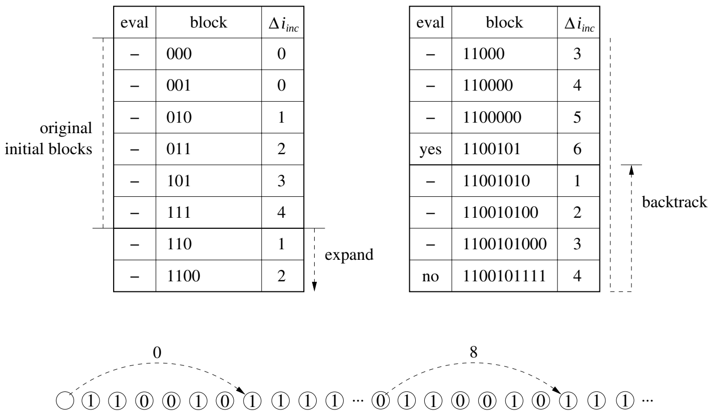
4.1 Conditions for Block Expansion

Before we show the specific methods, we give a general criterion for the optimization of a block-table. First, note that the total number of required multiplications, denoted by $T(E_B, d)$, can be written as follows:

$$T(E_B, d) = Pre(d) + Sq(E_B, d) + Mul(E_B, d),$$

where $Pre(d)$ is the number of required multiplications for block-table construction, and $Sq(E_B, d)$ and $Mul(E_B, d)$ are the numbers of squarings and multiplications along the shortest path, respectively. Thus $Sq(E_B, d) + Mul(E_B, d)$ is the sum of weights along the shortest path.

Initially, the block-table is the same as that of the sliding window method, and $Pre(d) = 2^{d-1}$. But if we can insert a new element into the initial block-table, then $Pre(d)$ will be increased since additional multiplications are needed to compute the new block. On the other hand, the newly inserted table element will also modify the initial graph. Since it is possible to newly link some node pairs by only one multiplication (excluding squarings), some new edges are inserted into the graph. This means that we have more paths than those of the initial graph. Thus we can possibility reduce $Sq(E_B, d)$ and/or $Mul(E_B, d)$ by finding the new shortest path of the modified graph. If we define $\Delta Pre(d)$ as the number of additional multiplications to compute the new table element, and



eval	block	Δi_{inc}
–	000	0
–	001	0
–	010	1
–	011	2
–	101	3
–	111	4
–	110	1
–	1100	2

eval	block	Δi_{inc}
–	11000	3
–	110000	4
–	1100000	5
yes	1100101	6
–	11001010	1
–	110010100	2
–	1100101000	3
no	1100101111	4

Fig. 4. Prefix block expansion

$\Delta Sq(E_B, d)$ and $\Delta Mul(E_B, d)$ as the reduced numbers of squarings and block multiplications, respectively, then the necessary condition for us to expand our block table is obviously

$$\Delta Pre(d) < \Delta Sq(E_B, d) + \Delta Mul(E_B, d). \tag{2}$$

In other words, if the condition (2) is satisfied, then we decide to insert the corresponding new element into the block-table. Note that the table expansion can be done iteratively until we cannot find an adequate element to be inserted.

4.2 Prefix Block Expansion

Our first heuristic to expand the block-table is prefix block expansion, where we define the *prefix block* of E_B as the first partitioned block of E_B in the initial graph.

We expand the prefix blocks as follows. When a computed block is odd, we compare the weights of the initially computed shortest path and the new shortest path determined by adding an edge associated with the expanded block. If the criterion (2) is satisfied, then we continue this expansion for the next odd block; Otherwise, we backtrack to the location where the last odd block was attached, and we stop the expansion of prefix blocks.

Figure 4 shows an example of prefix expansion in case for $E_B = 110101101110\ 1111110111110100111_2$ and $d = 3$. In case of expanding 1100101111_2, we backtrack to the location of 1100101_2 and stop since the reduced weight of shortest path is less than $\Delta Pre(d)$, i.e., 4 in this case. The edges corresponding to the expanded prefix blocks should be added to the graph.

4.3 Addition-Chain Block Expansion

We now describe another heuristic to expand the block-table. We consider a pair of blocks in the current block-table. Then, the new block, which is the sum of two

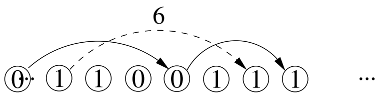


Fig. 5. Addition-chain block expansion

blocks, is called an *addition-chain block*. Note that, addition of two blocks in an exponent represents one multiplication in computing modular exponentiation. Thus, we have $\Delta Pre(d) = 1$ for each addition-chain block. Note that in our graph model, using the parameter d is one of the limitations to get the optimal sequence of multiplications. The addition-chain block expansion method is an effort to overcome such a limitation since the length of an addition-chain block may be larger than d.

Our expansion method is as follows. First, we find candidate elements by adding every pair of blocks in the current block-table. If the newly computed element is odd and the criterion (2) is satisfied, we update the block-table by inserting this candidate. We show an example of addition-chain block expansion in Fig. 5 when $1100_2 + 111_2 = 10011_2$.

4.4 Combination of Two Expansions

We describe how to combine two expansion methods. While there can be various combinations of the two methods, we consider only two representative combination forms as shown in Fig. 6. One is to execute two expansion procedures sequentially as in the left-hand side of Fig. 6. We denote this kind of combination as *ExD*. This combination can minimize expansions of useless blocks by prefix block expansions when we perform addition-chain expansion procedures.

The other combination is to execute the addition-chain block expansion procedure whenever each prefix block is expanded as in the right-hand side of Fig. 6. We denote this kind of combination as *ExM*. This combination can maximize the number of possible candidate blocks for addition-chain block expansion by

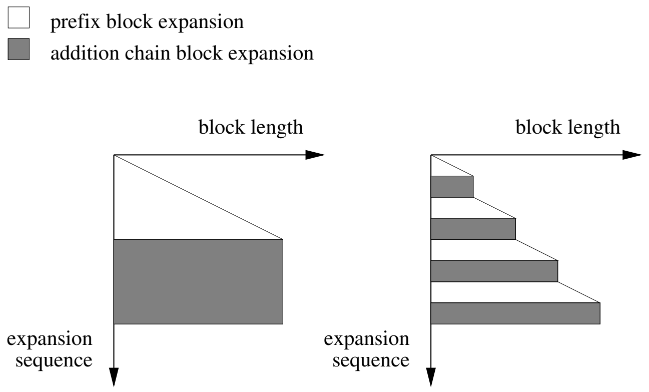


Fig. 6. Two combinations of two expansion processes

newly generated prefix blocks. We try both of the two approaches and show the experimental results in the next section.

5 Experimental Results

In our experiments, we consider two sets of exponents E_B whose lengths n are 512, 1024, and 2048. The first set is composed of 10,000 random exponents E_B generated via the binomial distribution $B(n, 1/2)$. The second set is composed of 9 subsets with 100 elements each, where the subsets are classified by the ratio of non-zero bits in E_B.

We compare the performance of our methods with those of the CLNW method and the VLNW method, which are known as the best practical exponentiation methods. We considered various parameters $d = \{1–7\}$ and $q = \{1–3\}$ for the CLNW and the VLNW methods, and we compared the best results with ours in all of the cases.

Table 1 shows the required numbers of multiplications for the four different methods with 10,000 random exponents generated by the binomial distribution $B(n, 1/2)$. By this table, we can see that the new method reduces the number of multiplications by 5.38 (0.88%), 8.87 (0.73%) and 15.18 (0.64%), when it is applied to the exponentiation with random exponents $E \approx 2^{512}$, $E \approx 2^{1024}$ and $E \approx 2^{2048}$, respectively.

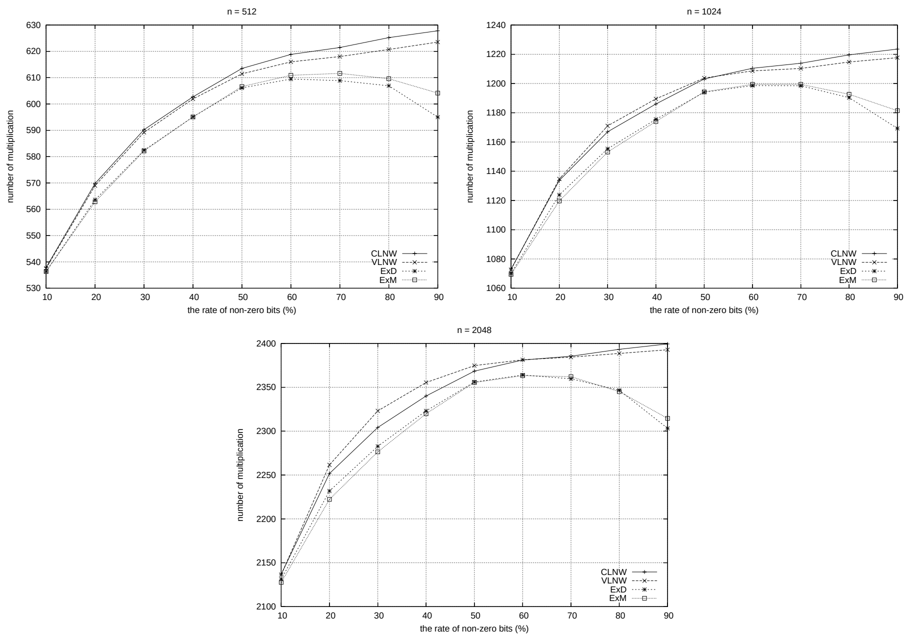


Fig. 7. Experimental results for effects for Hamming weights in exponents

Table 1. Comparison our methods with sliding window techniques

method	sliding window methods			general graph model			difference of
	CLNW	VLNW	min	ExD	ExM	min	min
n = 512	614.07	611.81	611.26	606.46	606.87	605.88	-5.38 (-0.88%)
n = 1024	1203.40	1203.70	1201.97	1193.92	1194.32	1193.10	-8.87(-0.73%)
n = 2048	2369.35	2375.39	2369.26	2355.97	2355.96	2354.09	-15.18(-0.64%)

Figure 7 shows the experimental results for exponents with various Hamming weights. In the figure, the horizontal axis represents the ratio of non-zero bits in E_B, and the vertical axis represents the average number of required multiplications for each method. We can see that if Hamming weight of E_B is low, our expansion techniques do not contribute so much since many of the blocks in E_B will be short. However, our methods reduce the number of multiplications remarkably in the case of high Hamming weights, because our prefix expansion method and addition-chain method can be effectively used to partition E_B into larger blocks. That is, the new method reduces the number of required multiplications up to about 33(5.22%), 54(4.44%) and 95(4.01%) for $E_B \approx 2^{512}$, $E_B \approx 2^{1024}$ and $E_B \approx 2^{2048}$, respectively.

References

1. R. L. Rivest, A. Shamir and L. Adleman: A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM* **21** (2) (1978) 120–126
2. T. ElGamal: A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory* **31** (4) (1985) 469–472
3. National Institute of Standards and Technology, *Digital Signature Standard*, FIPS Publication **186** (1994)
4. D. E. Knuth: *The art of computer programming: Seminumerical algorithms*, Volumn 2, 2nd edition, Addison-Wesley, Reading, MA (1981) 461–485
5. J. Bos, M. Coster: Addition chain heuristics. *In Proc. Crypto '89, Lecture Notes in Computer Science* **435** (1990) 400–407
6. P. Downey, B. Leong and R. Sethi: Computing sequences with addition chains. *SIAM J. Comp.* **10** (3) (1981) 638–646
7. C. K. Koç: Analysis of Sliding Window Techniques for Exponentiation. *Computers and Mathematics with Application* **30** (10) (1995) 17–24
8. Thomas H. Cormen, Charles E. Leiserson and Ronald L. Rivest: *Introduction to algorithms*, The MIT Press (1990) 514–531

Dynamic Facility Location with Stochastic Demands

Martin Romauch and Richard F. Hartl

University of Vienna, Department of Management Science,
Brünner Straße 72, 1210 Vienna, Austria
{martin.romauch, richard.hartl}@univie.ac.at

Abstract. In this paper, a Stochastic Dynamic Facility Location Problem (SDFLP) is formulated. In the first part, an exact solution method based on stochastic dynamic programming is given. It is only usable for small instances. In the second part a Monte Carlo based method for solving larger instances is applied, which is derived from the Sample Average Approximation (SAA) method.

1 Introduction

The Uncapacitated Facility Location Problem (UFLP) has been enhanced into many directions. In [9] and [2] you can find numerous approaches that consider either dynamic or stochastic aspects of location problems. An exact solution method to an UFLP with stochastic demands is discussed in [6]. The problem considered there could be interpreted as a two stage stochastic programm. In [8] you can find dynamic (multi period) aspects as well as the multi commodity aspect. The approach in [10] could be seen as the integration of stochastics into the UFLP. In the work in hand a model will be presented, where the UFLP gets enriched by inventory and randomness in the demand. The UFLP and its generalizations are part of the class of NP-hard problems, where no exact efficient solution methods are known. First of all, the aim of this work is the preparation of tools to develop and investigate heuristics for this problem type. For this reason, an exact method for small instances was developed. This makes possible both, to carve out the range of exact solvability and to compare exact and heuristic solutions. A more detailed description of the problem is now following.

2 Stochastic Dynamic Warehouse Location Problem

Our aim is to find the optimal decisions for production, inventory and transportation, to serve the customers during a certain number of periods, $t \in \{1, ..., T\}$. Assume that the company runs a number for the production sites $i \in I = \{1, 2, ..., n\}$ that have limited storage capacities, $\Delta_i^{(t)}$. These production sites need not be used in all periods. When a production site i is operated at time t, this is denoted by the binary variable $\delta_i^{(t)} = 1$. In this case the fixed costs $o_i^{(t)}$ arise. If a location is active, then the exact production quantity $u_i^{(t)}$ must be fixed. For each period, the production decision is the first stage of the decision process. It has to be done before the demand of the customers is known. Only the current level of inventory $y_i^{(t-1)}$ as well as the demand forecasts are known in advance.

Demand occurs at various customer locations $j \in J = \{1, 2, ..., m\}$. At any given period t the demand d_j^t at customer j will occur with probability p_j^t, whereas customer j will not require any delivery with probability $1 - p_j^t$. Hence, demand can be described by a dichotomous random variable[1] $\mathcal{D}_j^{(\tau)}$ $(\tau \geq t)$. We also assume that the random variables $\mathcal{D}_j^{(t)}$ are stochastically independent.

$$\mathbf{P}(\mathcal{D}_j^{(t)} = d_j^{(t)}) = p_j^{(t)} \quad \mathbf{P}(\mathcal{D}_j^{(t)} = 0) = 1 - p_j^{(t)}$$

In the second stage, when the demand is known, we must decide upon the transportation of appropriate quantities $x_{ij}(t)$ from the production sites i to the customers j. We assume that the time needed for transportation can be neglected (i.e. the transportation lead time is less than one period). Stockouts (shortages) $f_j^{(t)}$ are permitted and are penalized by shortage costs $p_j^{(t)}$ per unit time and per unit of the product. We assume that backordering is not possible and that these potential sales are lost.

The periods are linked by the inventories $y_j^{(t)}$ at the production sites and the usual inventory balance equations (1) apply. Here $\eta_i^{(t)}$ denotes the surplus in period t at site i.

$$y_i^{(t)} + \eta_i^{(t)} = y_i^{(t-1)} + u_i^{(t)} - \sum_{j \in J} x_{ij}^{(t)} \tag{1}$$

In this paper we assume free disposal, therefore the variable $\eta_i^{(t)}$ can be eliminated by turning the equality (1) into the inequality (2).

$$y_i^{(t)} \leq y_i^{(t-1)} + u_i^{(t)} - \sum_{j \in J} x_{ij}^{(t)} \tag{2}$$

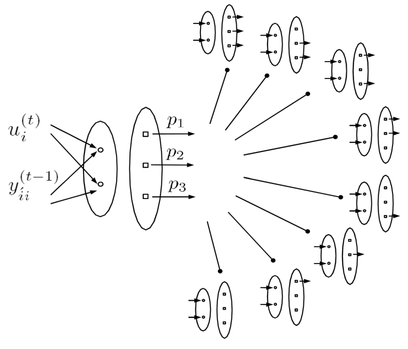
After the completion of the production and transportation decisions and after updating the inventories, the next period can be considered. We note here, that for all periods we have to pay attention to the capacity restrictions (3).

$$0 \leq u_i^t + y_i^{(t-1)} \leq \Delta_i^{(t)} \tag{3}$$

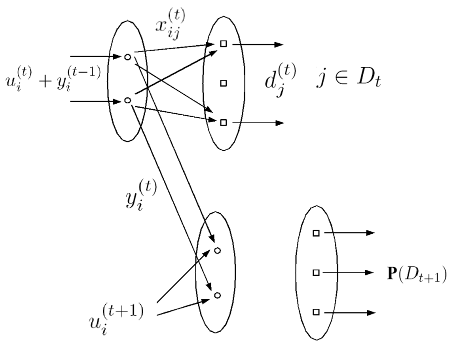
In order to have a convenient notation, we introduce the concept of scenarios. A scenario $D_t \subset J$ is a subset of customers where the demand gets realized. Since the demands of the different customers are independent, the corresponding probability of a scenario to occur is given in formula (4).

$$\mathbf{P}(D_t) = \prod_{j \in D_t} p_j^{(t)} \prod_{j \notin D_t} \left(1 - p_j^{(t)}\right) \tag{4}$$

[1] The embedding of stochastics is similar to the embedding of stochastics into the TSP, see [4].



(a) Scenarios ($|I| = 2, |J| = 3$)



(b) Transition

Fig. 1. Sequencing of Decisions

Solving the SDFLP means finding a strategy that minimizes the expected costs. Because of the sequencing of the decisions and the uncertain demand, the solutions could be understood as scenario dependent strategies, where the decisions are dependent on the forecasts and the level of inventory at hand. Figures 1(a) and 1(b) illustrate the dependency of operative planning[2] and the scenarios (realization of demand).

The left hand side of Figure 1(a) shows the production decisions $u_i^{(t)}$ and all of the possible subsequent scenarios (8 in number). One of the scenarios is magnified in the upper part of Figure 1(b). In each scenario the decisions for transportation, inventory and shortage are necessary.

In order to complete the model formulation, we summarize the decision variables and the corresponding costs in Table 1.

Table 1. Variables and Costs

variable	cost	description
$\delta_i^{(t)} \in \{0,1\}$	$o_i^{(t)}$	operating decision and fixed costs
$x_{ij}^{(t)} \in \mathbf{Z}_+$	$c_{ij}^{(t)}$	transportation decision and unit transportation cost
$y_i^{(t)} \in \mathbf{Z}_+$	$s_i^{(t)}$	inventory level and unit holding cost
$u_i^{(t)} \in \mathbf{Z}_+$	$m_i^{(t)}$	production decision and variable production cost
$f_j^{(t)} \in \mathbf{Z}_+$	$p_j^{(t)}$	shortage (lost sales) and unit shortage cost

The decisions $\delta_i^{(t)}$ and $u_i^{(t)}$ are linked by formula (5)

$$\delta_i^{(t)} = \begin{cases} 1 & \text{if } u_i^{(t)} > 0 \\ 0 & \text{if } u_i^{(t)} = 0 \end{cases} \tag{5}$$

[2] To keep the figure as simple as possible shortages and disposal are not integrated.

while shortages are defined as

$$f_j^{(t)} = \mathcal{D}_j^{(t)} - \sum_{i \in I} x_{ij}^{(t)} .$$

The total cost F is the sum over all periods of fixed operating costs, variable production costs,

$$F = \mathbf{E}\left(\sum_{t=1}^{T} \sum_{i \in I} \left[o_i^{(t)} \delta_i^{(t)} + m_i^{(t)} u_i^{(t)} + s_i^{(t)} y_i^{(t)} \right] \right. \\ \left. + \sum_{t=1}^{T} \sum_{i \in I} \sum_{j \in J} c_{ij}^{(t)} x_{ij}^{(t)} + \sum_{t=1}^{T} \sum_{j \in J} p_j^{(t)} f_j^{(t)} \right)$$

Since all relevant information about the past is contained in the inventory levels, this model is well suited to be solved by dynamic programming. This will be outlined in the next section.

3 Exact Solution Method

3.1 Stochastic Dynamic Programming

The principle of dynamic programming is the recursive estimation of the value function. This value function, henceforward denoted by F, contains the aggregate value of the optimal costs in all remaining periods. It can be derived recursively. It is convenient to first describe the method in general and to apply it to the problem afterwards. Let $z \in \mathbf{R}_+^m$ be the vector of state variables and $u \in \mathbf{R}_+^n$ be the vector of decisions. The set of feasible decisions in state z and period t is denoted by $U_t(z)$. The random influence in period t is represented by the random vector $r^{(t)}$ for which the corresponding distribution is known. It is important to note that the random variables $\{r^{(t)}\}$ have to be stochastically independent. The state transformation is described by

$$z_{t+1} = A(z_t, u_t, r_t)$$

and depends on the current state z_t, the random influence r_t at time t, and the chosen decision u_t. The single period costs in period t and state z when decision u is taken and random variable r is realized is denoted by $g_t(z, u, r)$.

The value function $F_t(z)$ gives the minimal expected remaining costs when starting in state z in period t. We now present a variant of the stochastic Bellman equation (compare Schneeweiß[11] S.151 (10.25) or Bertsekas [1] S.16).

$$\begin{aligned} F_T(z) &= \min_{u \in U_T(z)} \Big\{ \mathbf{E}\big[g_T(z, u, r^{(T)}) \big] \Big\} \\ F_t(z) &= \min_{u \in U_t(z)} \Big\{ \mathbf{E}\big[g_t(z, u, r^{(t)}) + F_{t+1}(A_t(z, u, r^{(t)})) \big] \Big\} \quad t = T-1, \ldots, 1 \end{aligned} \tag{6}$$

Recursively solving the equation (6) we get an optimal strategy that balances the cost for implementing the decision u and the expected resulting remaining costs.

Applying Stochastic Dynamic Programming (SDP) to the SDFLP is almost straight forward. For solving the problem we have to iteratively calculate the functions F_t. We will show later that for the SDFLP it is sufficient to consider integer controls.

3.2 Application to the SDFLP

In order to apply the DP equation (6) to the SDFLP, we first introduce the notation $G\big(D, y_i^{start}, y_i^{end}, t\big)$ for the sum of inventory holding costs, shortage costs, and transportation costs in scenario D in period t when starting with initial inventory levels y_i^{start} and where the final inventories are required to be y_i^{end}. To every given inventory level y_i^{start} and scenario D, the best possible transportation plan has to be calculated. This can be done by solving a linear program:

$$\begin{gathered} G\big(D, y_i^{start}, y_i^{end}, t\big) = \sum_{i \in I} s_i^{(t)} y_i^{end} + \\ \min_{x_{ij}, f_j} \sum_{j \in J} p_j^{(t)} f_j + \sum_{i \in I} \sum_{j \in J} c_{ij}^{(t)} x_{ij} \end{gathered} \tag{7}$$

$$\begin{aligned} s.t. \quad \textstyle\sum_{i \in I} x_{ij} + f_j &= d_j^{(t)} \quad \forall j \in D \\ \textstyle\sum_{j \in J} x_{ij} + y_i^{end} &\le y_i^{start} \; \forall i \in I \\ f_j, x_{ij} &\ge 0. \end{aligned}$$

Now the value function F_T of the final period T can be computed. In G, the starting inventory is now given by $y_i^{start} = u_i^{(T)} + y_i^{(T-1)}$ while the terminal inventory must be zero, $y_i^{end} = 0$:

$$\begin{aligned} F_T(y_i^{(T-1)}) = \min_{\substack{u_i^{(T)} \ge 0 \\ 0 \le u_i^T + y_i^{(T-1)} \le \Delta_i^{(T)}}} &\Big\{ \sum_{i \in I} \delta_i^{(T)} o_i^{(T)} + \sum_{i \in I} u_i^{(T)} m_i^{(T)} + \\ &+ \sum_{D_T \subset J} \mathbf{P}(D_T) G\big(D_T, u_i^{(T)} + y_i^{(T-1)}, 0, T\big) \Big\} \end{aligned} \tag{8}$$

Going back in time, we have to turn to the general case in period $t < T$. Now we have to take into account the remaining costs in periods $t+1, ..., T$ when making the decision in period t. The starting inventory is now given by

$y_i^{start} = u_i^{(t)} + y_i^{(t-1)}$ while the inventory at the end of period t is $y_i^{end} = y_i^{(t)}$. When determining $G(D_t, u_i^{(t)} + y_i^{(t-1)}, y_i^{(t)}, t)$ again a linear program has to be solved. The recursion for the value function becomes:

$$\begin{aligned} F_t(y_i^{(t-1)}) = & \min_{\substack{u_i^{(t)} \geq 0 \\ 0 \leq u_i^{(t)} + y_i^{(t-1)} \leq \Delta_i^{(t)}}} \Big\{ \sum_{i \in I} \delta_i^{(t)} o_i^{(t)} + u_i^{(t)} m_i^{(t)} + \\ & + \sum_{D_t \subset J} \mathbf{P}(D_t) \min_{0 \leq y_i^{(t)} \leq \Delta_i^{(t+1)}} \big\{ G_t\big(D_t, u_i^{(t)} + y_i^{(t-1)}, y_i^{(t)}, t\big) + F_{t+1}(y_i^{(t)}) \big\} \Big\} \end{aligned} \tag{9}$$

In the SDFLP the data and the controls are assumed to be integer. In the problem (7) we therefore have to solve an integer linear program. It turns out to be a min cost flow problem and therefore it is totally unimodular, such that using the Simplex method for solving the relaxed linear program results in integer solutions for the transportation quantities x_{ij} and the shortages f_j.

The computational effort of this exact algorithm is increasing exponentially with the capacity at the locations and the number of customers. The additional effort that emerges from adding additional periods to the problem is linear.

This DP formulation is only applicable for small problem instances and for larger problem instances heuristic approaches are necessary. This is considered in the next section.

4 Heuristic Approach

A heuristic designed to solve stochastic combinatorial optimization problems is the Sample Average Approximation Method (SAA); see Kleywegt et al. [5]. Our model deals with a multi stage problem and that is the reason why this method is not directly applicable. In what follows we first present the classical SAA for solving static stochastic combinatorial optimization problems. Afterwards, we will explain how this method can be modified in order to be applicable to our problem.

Consider the following stochastic combinatorial optimization problem (10) in which W is a random vector with known distribution P, and S is the finite set of feasible solutions.

$$v^\star = \min_{x \in \mathcal{S}} g(x), \quad g(x) := \mathbf{E}_P G(x, W) \tag{10}$$

The main idea of the SAA method is to replace the expected value $\mathbf{E}_P G(x, W) = \int G(x, w) P(dw)$ (which is usually very time consuming) by the average of a sample. The following substitute problem (11) is an estimator of the original problem (10).

$$\min_{x \in \mathcal{S}} \hat{g}_N(x), \quad \hat{g}_N(x) := \frac{1}{N} \sum_{j=1}^{N} G(x, W^j) \tag{11}$$

The SAA method works in three steps

1. Generate a set of independent identically distributed samples: $\{W_i^1, \ldots, W_i^N\}_{i=1}^M$ of the random variable W.
2. Solve the corresponding optimization problems, i.e. optimize:

$$\hat{v}_i = \min_{x \in \mathcal{S}} \hat{g}_i(x), \quad \hat{g}_i(x) = \frac{1}{N} \sum_{j=1}^{N} G(x, W_i^j).$$

3. Estimate the solution quality. This is done by first computing mean and variance of the sample:

$$\hat{v} = \frac{1}{M} \sum_{i=1}^{M} \hat{v}_i, \quad \hat{\sigma}^2 = \frac{1}{M(M-1)} \sum_{i=1}^{M} (\hat{v}_i - \hat{v})^2.$$

Then a solution $\tilde{x}$ is chosen (e.g. we can take the solution with the smallest $\hat{v}_i$) and its objective value is estimated more accurately by generating a larger sample $\{W^1, \ldots, W^{N'}\}$ $(N' >> N)$

$$\tilde{v} = \frac{1}{N'} \sum_{j=1}^{N'} G(\tilde{x}, W^j), \quad \tilde{\sigma}^2 = \frac{1}{N'(N'-1)} \sum_{j=1}^{N'} (G(\tilde{x}, W^j) - \tilde{v})^2.$$

Calculate the value *gap* and σ^2_{gap}:

$$gap = \tilde{v} - \hat{v}, \quad \sigma^2_{gap} = \tilde{\sigma}^2 + \hat{\sigma}^2$$

Since $\mathbf{E}(\hat{v}) \le v^\star \le \mathbf{E}(\tilde{v})$ the values $\tilde{v}$ and $\hat{v}$ can be interpreted as bounds on $v^\star$: let $x^\star \in \mathcal{S}$ denote an optimal solutions of (10) then the first inequality $\mathbf{E}(\hat{v}) \le v^\star$ comes from taking the expected value on the following inequality:

$$\hat{v}_i \le \hat{g}_i(x^\star)$$

which results in

$$\mathbf{E}(\hat{v}_i) \le \mathbf{E}(\hat{g}_i(x^\star)) = v^\star.$$

After completing Step 3, we have to inspect the values of *gap* and σ^2_{gap}. If these values are too large, one must repeat the procedure with increased values of N, M and N'. In [10] this method is applied for a Supply-Chain Management problem that includes location decisions.

Because of the multi-period structure of the SDFLP, the above SAA procedure has to be adapted. In particular, one must pay special attention to the way how the sampling is done. The sampling is done independently in every stage and every state of the SDP sipmly by modifying formulas (8) and (9), where for every period and inventory level we only sum over a small randomly chosen sets of scenarios. To be more specific, the expected value in formula (9) passes over into (12) where $\{D_i\}$ denotes the sample chosen in stage t and state $y_i^{(t)}$.

$$\frac{1}{N} \sum_{i=1}^{N} \min_{0 \le y_i^{(t)} \le \Delta_i^{(t+1)}} \left\{ G_t\big(D_i, u_i^{(t)} + y_i^{(t-1)}, y_i^{(t)}, t\big) + F_{t+1}(y_i^{(t)}) \big) \right\} \qquad (12)$$

5 Results

The implementation was done in C++ and an additional library from the GNU Linear Programming Kit (GLPK) [7] was used. For small examples where the product $\prod_{i \in I} \Delta_i^{(t)}$ is small enough (say ≤ 50) it is possible to find the exact solution using the dynamic programming approach described in Section 3. In Figure 2, we can see the cost distributions for the exact and the heuristc approach in a small example (3 periods, 3 customers, 2 facilities and capacity = 3). Here one can see that the shapes are quite different. The instance considered has very

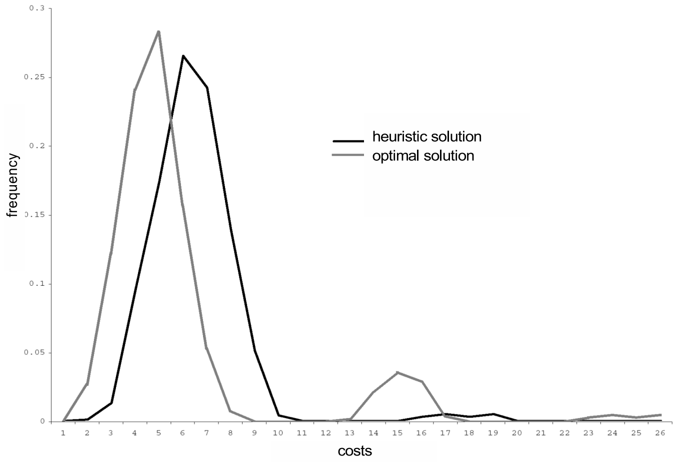


Fig. 2. Comparison of Different Solutions

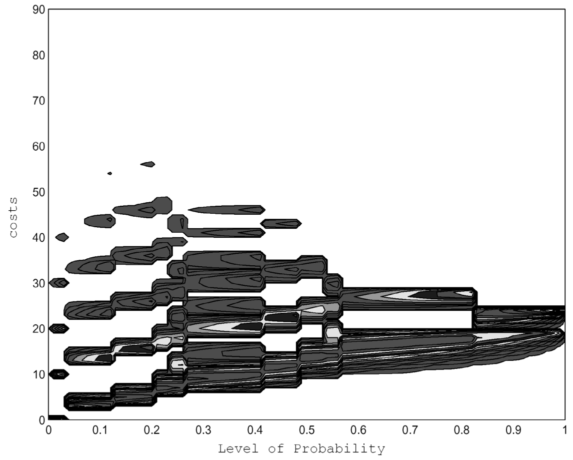


Fig. 3. Distributions of the Optimal Solution to Instances with Different Levels of Probability ($p_j^{(t)} = p \in \{0, 0.01, 0.02, \ldots, 1\}$)

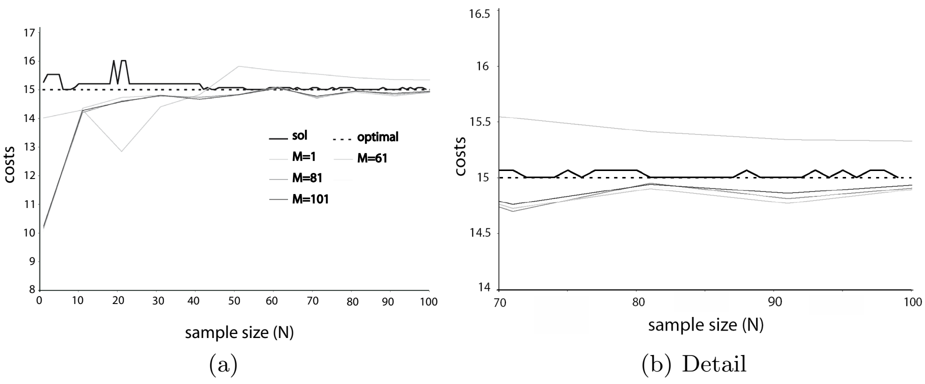


Fig. 4. Choice of sample size N and the number of samples M

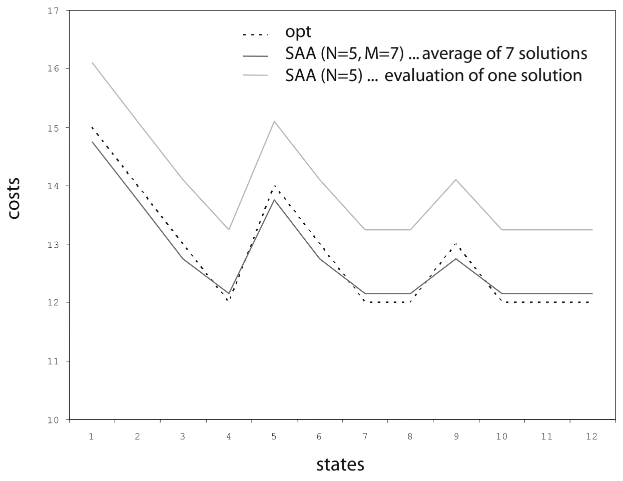


Fig. 5. Expected costs to different levels of inventory y ($1 : [0, 0]$; $2 : [0, 1]$; ...; $4 : [0, 3]$; $5 : [1, 0]$; ...; $12 : [2, 3]$)

uncertain demand (every customer has probability 50%). Hence the two peaks in the optimal solution are not very surprising. It is interesting to observe that the heuristic solution does not show these twin peaks.

This second peak diminishes if the probability is close to 0 or 1. An experiment was made where the probability for the demand varied from 0 to 1 for all customers, i.e.: ($p_j^{(t)} = p \in \{0, 0.01, 0.02, \dots, 1\}$). The result is depicted in Figure 3 where grey areas represent positive probabilities that these cost values occur. Every vertical line (p fixed) corresponds to a distribution function. For instance, at $p = 0.1$ five peaks occur. When the probability p increases, the number of peaks in the distribution function decreases.

The key decision to make the heuristic work well is to choose the right sample size N and the right number of samples M. In Figure 4 the statistical lower bound $\hat{v}$ (calculated in step 3) is depicted for different values of N and M. Choosing a sample size N that is large enough seems to be more important than a large number of samples. In Figure 4(b) the region $N > 70$ of Figure 4) is magnified to see the effect of a choosing the number of samples more clearly. One also can see that the statistical gap stays positive if at least 11 samples of size 71 are chosen. It is also interesting to note that for small values of N and sufficient large M the corresponding bounds are quite good, although the corresponding individual solutions are quite bad. This situation is depicted in Figure 5.

6 Conclusion and Further Research

In this paper a stochastic dynamic facility location problem was proposed and exact and heuristic solution methods were presented. The examples that can be solved to optimality are quite small and therefore of minor practical interest. But the comparison of the SAA results and the exact solution method shows the applicability of the proposed method for larger instances of the SDFLP. To get more insight into this method, it will be necessary to make a transfer of the theoretical results known for the SAA method (see [5] for statistical bounds). For comparison purposes it would be interesting to adopt metaheuristic concepts: e.g. by using the variable-sample approach (for references see [3]). In our further research we also want to consider other exact solution techniques considering the SDFLP as a multistage stochastic program.

References

1. Bertsekas, D.P.: Dynamic Programming and Optimal Control, Athena Scientific, Belmont, MA (1995)
2. Hammer, P.L.: Annals of Operations Research: Recent Developments in the Theory and Applications of Location Models 1/2, Kluwer Academic Publishers (2002)
3. Homem-de-Mello, T.: Variable-Sample Methods for Stochastic Optimimization, ACM Trans. on Modeling and Computer Simulation (2003)
4. Jaillet, P.: Probabilistic Travelling Salesman Problems, Ph.D. thesis, MIT (1985)
5. Kleywegt, A.,Shapiro,A., Hohem-de-Mello, T.: The Sample Average Method for Stochastic Discrete Optimization, SIAM J. Optim. 12 (2001/02)
6. Laporte, G., Louveaux, F., Van Hamme, L.: Exact Solution to a Location Problem with Stochatic Demands, Transportation Science, Vol. 28,(1994)
7. Makhorin, A.:GNU Linear Programming Kit - Reference Manual - Version 4.1, Department for Applied Informatics, Moscow Aviation Institute, Moscow, Russia (2003)
8. Melo, M.T., Nickel, S., Saldanha da Gama, F.: Large-Scale Models for Dynamic Multi-Commodity Capacitated Facility Location, Berichtsreihe des Fraunhofer Inststituts für Techno- und Wirtschaftsmathematik (ITWM), ISSN 1434-997, Kaiserslautern (2003)
9. Hesse Owen, S., Daskin, M.S.: Strategic Facility Location: A Review, Eropean Journal of Operational Research, Vol. 111, (1998)
10. Santoso, T., Ahmed, S., Goetschalckx, M., Shapiro, J.: A Stochastic Programming Approach for Supply Chain Network Design under Uncertainty, The Stochastic Programming E-Print Series (SPEPS), (2003)
11. Schneeweiß, C.: Dynamisches Programmieren, Physica-Verlag, (1974)

The Complexity of Classical and Quantum Branching Programs: A Communication Complexity Approach

Farid Ablayev[⋆]

Dept. of Theoretical Cybernetics, Kazan State University,
420008 Kazan, Russia
ablayev@ksu.ru

Abstract. We present a survey of the communication point of view for a complexity lower bounds proof technique for classical (deterministic, nondeterministic and randomized) and quantum models of branching programs.

Keywords: Branching programs, communication computations, quantum computations.

1 Classical Branching Programs and Communication Complexity

We present a survey of the communication point of view for a complexity lower bounds proof technique for classical (deterministic, nondeterministic and randomized) and quantum models of branching programs. We do not try to present a complete survey on communication complexity methods for branching programs — this is a hard task. Our goal is to give some intuition how communication complexity technique works for the area of branching programs.

A good source of information on classical branching programs is Wegener's book [34], and for an introduction to quantum computation see Nielsen and Chuang [22]. In the paper we try to make citation to original papers on branching programs. Some of these facts are displayed in [34]. Randomized branching programs intensively investigated since 1996 when first result on "exponential separation complexity" for deterministic and randomized read-once branching programs was proved in [3]. Many authors developed lower bound technique for randomized branching programs complexity. We would like to mentions here results due to Martin Sauerhoff whose last decade research results on randomized branching programs draw light to this area. Recently different models of quantum branching programs defined [6,21]. Investigation of quantum branching program is a new perspective area in the theory of branching programs.

[⋆] Supported by the Russia Fund for Basic Research under the grant 03-01-00769.

Branching programs. Recall definition of branching programs (BP)[34].

A *deterministic* branching program (BP) P is a finite directed acyclic graph which accepts some subset of $\{0,1\}^n$. Each node (except for the sink nodes) is labeled with an integer $1 \leq i \leq n$ and has two outgoing arrows labeled 0 and 1. This pair of edges corresponds to querying the i'th bit x_i of the input, and making a transition along one outgoing edge or the other depending on the value of x_i. There is a single source node corresponding to the start state, and a subset *Accept* of the sink nodes corresponding to accepting states. An input σ is *accepted* if and only if it induces a chain of transitions leading from source to a node in *Accept*, and the set of such inputs is the language accepted by the program.

The branching program P computes function g in the obvious way: for each $\sigma \in \{0,1\}^n$ we let $g(\sigma) = 1$ iff there is a directed path starting in the source s and leading to the accepting node *accept* such that all labels $x_i = \sigma_i$ along this path are consistent with $\sigma = \sigma_1\sigma_2 \ldots \sigma_n$.

A *nondeterministic* BP (for short NBP) is branching program with "guessing nodes" (see for example [11]) that is nodes with two outgoing edges being unlabeled. Unlabeled edges allow all inputs to produce. A nondeterministic branching program P computes a function g, in the obvious way; that is, $g(\sigma) = 1$ iff there exists (at least one) computation on σ starting in the source node s and leading to the accepting node *accept*.

A *randomized* BP [3,34] (for short RBP) is a one which has in addition to its standard inputs specially designated inputs called "random inputs". When values of these "random inputs" are chosen from the uniform distribution, the output of the branching program is a random variable.

Say that RBP (a,b)-computes a boolean function f if it outputs 1 with probability at most a for input σ such that $f(\sigma) = 0$ and outputs 1 with probability at least b for inputs σ such that $f(\sigma) = 1$.

As usual for a branching program P (deterministic or random), we define size(P) (complexity of the branching program P) as the number of internal nodes in P. Define, following [11], the size(P) of the nondeterministic branching program P as the number of internal nodes in P minus the number of guessing nodes.

In theory BP-s are useful for investigation the amount of space necessary to compute various functions. That is (see the book [34]),

$$LOGSPACE/poly = BP(poly) \tag{1}$$

where $LOGSPACE/poly$ is the nonuniform analogy of $LOGSPACE$ and $BP(poly)$ is the class of all functions having polynomial DBP size. Another known result due to Barrington (see [8] for more information) is that constant width (width 5) DBP-s are as powerful as $O(\log n)$ depth circuits with constant fan-in gates.

One-way Communication Computations. A *communication protocol* for a function $f : \{0,1\}^{n_1} \times \{0,1\}^{n_2} \to \{0,1\}^m$, is a distributive algorithm of computation among two parties, conventionally referred to as Alice and Bob, in order for Bob to acquire the value of $f(v, \omega)$, where, initially, Alice is given $v \in \{0,1\}^{n_1}$ and Bob is given $\omega \in \{0,1\}^{n_2}$. The *cost* of a communication protocol is defined as the minimal amount of communication (number of bits) necessary to Alice and Bob to compute the value of $f(v, \omega)$ for a worst-case input (v, ω). The *communication complexity* of a function f, is defined as the cost of the best protocol for f. This model of communication computation was introduced by Yao [35]. Since 1979 different models of communication computations were introduced (see books [19,14]).

Randomized communication protocols are that produce results probabilistically [19,14]. Let $p = \frac{1}{2} + \varepsilon$ for $0 \leq \varepsilon \leq 1/2$. Say that the probabilistic protocol ϕ p-computes a function h if for every input $\sigma = (u, w)$ it holds that $h(\sigma) = b$ iff the probability of outputting the bit b in the computation $T_\phi(u, w)$ is no less than p.

In the paper we consider only one-way communication computations. For this model only player A can send messages to B. Player B on obtaining a message and his part of input produces a result. Let a set $U \subseteq \{0,1\}^n$ be such that $U = L \times R$. The randomized communication complexity $C(\phi)$ of the probabilistic protocol ϕ on the inputs from U is $\lceil \log |M(\phi)| \rceil$, where $M(\phi)$ is the set of messages used by ϕ during computations on inputs from U. For $p \in [1/2, 1]$ the randomized communication complexity $PC^U_{p,\pi}(h)$ of a boolean function h is

$$\min\{C(\phi) : \text{ protocol } \phi \ p\text{-computes } h \text{ for the partition } \pi \text{ of inputs from } U\}.$$

1.1 OBDD Models and One-Way Communication

Developments in the field of digital design and verification have led to the restricted forms of branching programs. The most common model used for verifying circuits is a polynomial size *ordered read-once branching program* also called an *ordered binary decision diagram* (for short OBDD), (see [10,33,20]). The importance of OBDD for practice based on the following fact: Boolean manipulations with OBDD (equivalence checking, ...) can be performed deterministically in polynomial time.

Read-once branching program is a BP in which for each path each variable is tested no more than once. A τ-ordered read-once branching program is a read-once branching program which respects an ordering τ of the variables, i.e. if an edge leads from an x_i-node to an x_j-node, the condition $\tau(i) < \tau(j)$ has to be fulfilled. An OBDD, is a τ-ordered read-once branching program which respects some ordering τ of variables.

It has turned out that many important for practice functions cannot be computed by polynomial size OBDD-s and read-once branching programs (see [10], [31], and [24]). One of such important functions is multiplication function $MULT$ which is exponentially hard for deterministic and randomized OBDDs [9,5]. Communication complexity approach is the main method for proving lower bounds for OBDD like models.

Definition 1. *We call branching program a π-weak-ordered branching program if its respects a partition π of variables $\{x_1, x_2, \ldots, x_n\}$ into two parts X_1 and*

X_2 such that if an edge leads from an x_i-node to an x_j-node, where $x_i \in X_t$ and $x_j \in X_m$, then the condition $t \leq m$ has to be fulfilled.

We call branching program P an weak-ordered if it is π-weak-ordered for some partition π of the set of variables of P into two sets.

Using the property 1 below many exponential lower bounds for function presentation by OBDDs were proved. The analog of the property 1 is correct for deterministic and nondeterministic cases.

Property 1. *Let $\varepsilon \in [0, 1/2]$, $p = 1/2 + \varepsilon$. Let randomized π-weak-ordered branching program P $(1-p, p)$-computes function $h : \{0,1\}^n \to \{0,1\}$. Let $U \subseteq \{0,1\}^n$ be such that $U = L \times R$, where L and R are defined in according to partition π of inputs. Then*

$$size(P) \geq 2^{PC^U_{p,\pi}(h)-1}.$$

Proof. The key idea (which now folklore) is to view on program P as a specific communication protocol Φ, which p-computes function h for the partition π of inputs.

Let $\sigma \in U$ be a valuation of x, $\sigma = (u, w)$, $u \in L$, $w \in R$. Players A and B receive respectively u and w in according to partition π of inputs. Let $v_1, \dots, v_d$ be all internal nodes of P that are reachable during paths of computation on the part u of input σ with non zero probabilities $p_1(u), \dots, p_d(u)$.

During the computation on the input u, player A sends node v_i with probability $p_i(u)$ to player B. Player B on obtaining message v_i from A starts its computation (simulation of the branching program P) from the node v_i on the part w of the input σ.

From the definition of the protocol Φ results the statement of Property. ▮

As an example of exponential hard (for OBDDs) function we consider the following Boolean function first considered in [29]. Let n be an integer and let $p[n]$ be the smallest prime greater or equal to n. Then, for every integer s, let $\omega_n(s)$ be defined as follows. Let j be the unique integer satisfying $j = s \bmod p[n]$ and $1 \leq j \leq p[n]$. Then, $\omega_n(s) = j$, if $1 \leq j \leq n$, and $\omega_n(s) = 1$ otherwise. For every n, the boolean function $g_n : \{0,1\}^n \to \{0,1\}$ is defined as $g_n(\sigma) = \sigma_j$, where $j = \omega_n(\sum_{i=1}^n i\sigma_i)$.

It was proved [29] that g_n needs size at least $2^{n-3\sqrt{n}}$ for computation by deterministic read-once branching program.

Theorem 1 ([2]).
Let $\varepsilon \in [0, 1/2]$, $p = 1/2 + \varepsilon$. Then for arbitrary $\delta > 0$ for every n large enough it holds that any randomized ordered read-once branching program that $(1-p, p)$-computes function g_n has the size no less than

$$1/4 \left(\frac{2^{n - \lceil 3\sqrt{n} \rceil}}{n} \right)^{1-(1+\delta)H(p)}.$$

1.2 Regular $(1,+k)$-BP and One-Way Communication with Information Overlap

In this section we consider branching programs with the restriction *only* on the number of variables testing and without limitation on the ordering (weak-ordering) of variables reading. In this case the possibility of applying communication methods for proving lower bounds is not evident as for OBDD model: we have no possibility for natural partition of the set of variables among two players as in the case of OBDD (see for example [25] for more arguments). Simon and Szegedy developed [31] another (different from the communication method) lower bound technique for proving lower bounds for read-once branching programs. Their method is the generalization of technique used by different authors for the read-once BP model.

Below we present communication lower bound proof method that works for more general than read-once BP model.

Let v be a node of BP P. We denote $X(v)$ a set of all variables from X that are tested on paths of P from the source to v.

Definition 2. *A branching program P is called regular if*

(i) for each node v of P on each path from the source to v the same set $X(v) \subseteq X$ of variables is tested and

(ii) on each path from the source to a sink all variables X are tested.

Remark. Notice, that similar BPs model was investigated by Oklonishnikova [23], she used word "uniform" for defining such model. We think that it would be more convenient to use word "regular" than the word "uniform" which already has its own meaning for denoting computational models.

Clearly we have that OBDDs are regular branching programs. Notice that an arbitrary branching program P can be transformed into a regular read-once branching program P' by inserting dummy tests such that $size(P') \leq 2n\, size(P)$ [23].

Definition 3. *Call P a* regular $(1,+k)$-branching program *(for short $(1,+k)$-ReBP) iff (i) P is regular and (ii) along every consistent path at most k variables are tested more than once.*

Notice that the procedure of inserting dummy tests for ordinary $(1,+k)$-BP P (in order to get regular BP P' from P) can violates the $(1,+k)$ property.

Decomposition of $(1,+k)$-ReBP. Consider X-input $(1,+k)$-ReBP P. Call an edge (v,v') of P an x_i-edge if it is labeled by $x_i = 0$ or $x_i = 1$. By a path *path* of P we mean a *consistent* path. That is, *path* is a sequence of nodes $path = (v_0, \dots, v_l)$, where v_0 is a source of P, v_l is one of a sinks of P and an assignment $\gamma \mapsto X$ of constants $\gamma = \gamma_1, \dots, \gamma_n$ to variables in X is such that each x_j-edge (v_i, v_{i+1}) of *path* is labeled by $x_j = \gamma_j$.

We will also use notion *computation* in the paper. By a computation $path_\gamma$ of P we mean path *path* together with its label $\gamma \mapsto X$.

Let $S \subset X$. Define an *S-border* set $B(S)$ of nodes of P as follows:

$$B(S) = \{v : X(v) = S \textit{ and for all offspring nodes } v' \textit{ of } v\ S \subset X(v') \textit{ properly } \}.$$

Denote $P(S)$ a subprogram (subgraph) of P which is determined by all paths *path* with the property: *path path contains a node from $B(S)$*. Note that for nondeterministic P its $P(S)$ subprogram can be deterministic.

The following property is evident.

Property 2. *Each path path of $P(S)$ contains exactly one node from $B(S)$.*

We will view on $P(S)$ as a BP consisting of two parts P^1 and P^2 where P^1 consists of the part of $P(S)$ "before" $B(S)$, including S-border $B(S)$ and P^2 — is a remind part of $P(S)$.

Let $Z \subset S$. Let $P_Z(S)$ be a subprogram (subgraph) of $P(S)$ which determined by *all* paths *path* with the property: *for any computation of $P(S)$ only variables from Z are tested in the second part P^2 of $P(S)$.* From the property that P is $(1,+k)$-ReBP program it follows that $|Z| \leq k$.

Let ρ be a map $\rho : Z \to \{0,1\}$. Denote $P|_\rho(S)$ a subgraph of $P_Z(S)$ which determined by *all* paths *path* with the property: *for path all its z_i-edges, $z_i \in Z$, are marked $z_i = \sigma_i$ in according to the map ρ.*

Definition 4. *We call a family $R = \{S : S \subset X\}$ of subsets of X a P-family if it is true that each computation $path_\gamma$ of P belongs to some subprogram $P(S)$ of P for $S \in R$ and removing arbitrary set S from R violates this property.*

The following lemma is motivated by the exposition in [11].

Lemma 1 (decomposing lemma). *Let P be a nondeterministic $(1,+k)$-ReBP. Let R be a P-family. Then P can be represented in the form*

$$P = \bigcup_{S \in R} \bigcup_{\substack{\rho: Z \to \{0,1\}, \\ |Z| \leq k}} P|_\rho(S).$$

Proof. Each $S \in R$ determines a subprogram $P(S)$ of P and each computation $path_\gamma$ belongs to some subprogram $P(S)$ of P for $S \in R$. So, $P = \bigcup_{S \in R} P(S)$. Each computation $path_\gamma$ of $P(S)$ determines a set $Z \subset X$, $|Z| \leq k$, of all variables tested along $path_\gamma$ both in the first part P^1 and the second part P^2 of $P(S)$. So, $P(S) = \bigcup_{\substack{Z \subset X, \\ |Z| \leq k}} P_Z(S)$. All (suitable) settings $\rho : Z \to \{0,1\}$ determines nonempty subprograms $P|_\rho(S)$ of $P_Z(S)$. ∎

We call the presentation of $(1,+k)$-ReBP P from Lemma 1 an *R-decomposition of P* or just *decomposition of P*.

Lower Bounds for $(1,+k)$-ReBP. Let $MP = P|_\rho(S)$. Let $B(S)$ be an S-border set of $P|_\rho(S)$. Using the property that $size(P) \geq |B(S)|$ we reduce a problem of proving lower bound for $size(P)$ for proving a lower bound for $|B(S)|$.

Denote ψ a boolean function computable by subprogram $P|_\rho(S)$ of P. Note that for deterministic program P that computes function f for all inputs γ of $P|_\rho(S)$ it holds that $f|_\rho(\gamma) = \psi(\gamma)$ but for nondeterministic P this can be not true (for an input γ for which $f|_\rho(\gamma) = 1$ the subprogram $P|_\rho(S)$ can contain only rejecting paths of P).

Denote $S_1 = S\backslash Z$, $S_2 = X\backslash(S_1 \cup Z)$. Subprogram $P|_\rho(S)$ of P is a *weak ordered.*

Denote $NC^{S_1:S_2}(\psi)$ ($DC^{S_1:S_2}(\psi)$) a nondeterministic (deterministic) one-way communication complexity of computing ψ for the case when player A gets only bits in S_1 and B — only bits in S_2.

Denote $CM^{S_1:S_2}(\psi)$ a communication matrix. That is, $CM^{S_1:S_2}(\psi)$ is an $|S_1| \times |S_2|$ boolean matrix, (σ, σ') entry of $CM^{S_1:S_2}(\psi)$ is $\psi(\sigma, \sigma')$. Denote $nrow(CM^{S_1:S_2}(\psi))$ to be a number of different rows of $CM^{S_1:S_2}(\psi)$.

Lemma 2. *Let P be an X-input $(1, +k)$-ReBP. Let $P|_\rho(S)$ be an arbitrary subprogram of decomposition of P and ψ be a function computable by $P|_\rho(S)$. If P is nondeterministic then*

$$|B(S)| \geq 2^{NC^{S_1:S_2}(\psi)}.$$

If P is deterministic then

$$|B(S)| \geq nrow(CM^{S_1:S_2}(\psi)).$$

Proof. Describe the following communication protocol Φ, which computes function ψ. Let $\gamma \in \Sigma$ be a valuation of X. Denote $\gamma = (\sigma^1; \sigma; \sigma^2)$, where $\sigma^1 \mapsto S_1$, $\sigma \mapsto Z$, $\sigma^2 \mapsto S_2$ assignments of γ to S_1, S_2, and Z respectively.

Players A and B receive respectively σ^1 and σ^2. Setting $\sigma \mapsto Z$ is known both to A and B. This model of computation is known as a communication computation with overlap information [19]. Let $v_1, \ldots, v_d$ be all internal nodes of $P|_\rho(S)$ that are reachable during paths of computation on the part σ^1 of γ.

During the computation on the input σ^1 player A nondeterministically selects and sends node v_i to player B. Player B on obtaining message v_i from A starts its computation (simulation of $P|_\rho(S)$) from the node v_i on the part σ^2 of the input γ.

The statements of the lemma result from the definition of the protocol Φ and known fact that $DC^{S_1:S_2}(\psi) = \lceil \log nrow(CM^{S_1:S_2}(\psi)) \rceil$ [35]. ▮

The boolean function $f_{n,k}$ is defined in [30]. Informally it defined as follows. The n variables X are divided into k blocks of length m. For every $j = 1, 2, \ldots, k$ a weighted sum of the bits of block j determines an index $i_j \in \{1, 2 \ldots, n\}$ of input bits. Then, the value of the function is the parity of bits determined by i_j for $j = 1, 2, \ldots, k$. See [30] for the formal definition.

Theorem 2. *Let for $k \geq 1$ the function $f_{n,k}$ is computed by a deterministic $(1, +(k-1))$-ReBP P. Then*

$$size(P) \geq 2^{(n/k - k\log(n/k) - 3\sqrt{n} - k)/2}$$

Proof. The proof uses decomposing lemma and communication technique. ▮

Note that using arguments of the paper [12] we can construct $(1,+k)$-ReBP for presentation $f_{n,k}$ of size $O(n^k)$. Together with the theorem 2 this proves proper hierarchy of the computational power of $(1,+k)$-ReBP-s in respect of parameter k. In particular this proves that $(1,+k)$-ReBP-s are more powerful than read-once BP-s. Note that Savicky and Zak in [30] proved the proper hierarchy for ordinary $(1,+k)$-BP-s with respect to k.

2 Quantum Communication Computations and OBDDs

Quantum communication protocol is a generalization of randomized communication protocols (see for example [16]). In a quantum protocol both players have a private set of qubits. Some of the qubits are initialized to the input before the start of the protocol, the other qubits are in state $|0\rangle$. Alice performs some unitary transformation on her qubits and then sends some of the qubits (channel qubits) to Bob. Bob performs some unitary transformation on channel qubits and his part of qubits. Then some qubits are measured and the result is taken as the output.

In a (bounded error) quantum protocol the correct answer must be given with a probability $1-\delta$ for some $\delta \in (0,1/2)$. The (bounded error) quantum complexity of a function is denoted $Q_\delta(f)$.

For a Boolean function $f : \{0,1\}^{n_1} \times \{0,1\}^{n_2} \to \{0,1\}$ its communication matrix is an $2^n \times 2^n$ matrix with $CM_f[x,y] = f(x,y)$.

For $V \subseteq \{0,1\}^{n_1}$ and $W \subseteq \{0,1\}^{n_2}$ denote $CM^{V,W}$ the $|V|\times|W|$ submatrix of CM_f formed by rows V and columns W of CM_f. Denote $\mathcal{M}_f$ a set of submatrices of CM_f with *pairwise different rows.* For matrix $CM^{V,W} \in \mathcal{M}_f$ call set W a *control set.*

Theorem 3. *For every function $f : \{0,1\}^{n_1} \times \{0,1\}^{n_2} \to \{0,1\}$, every $\delta \in (0,1/2)$, and every $CM^{V,W} \in \mathcal{M}_f$,*

$$Q_\delta(f) \geq \log|V| - |W|H(\delta).$$

where $H(\delta) = -\delta\log\delta - (1-\delta)\log(1-\delta)$.

Proof. The proof uses entropy approach and omitted. Such approach used in several papers for classical randomized and quantum communication computations [1,17,16]. ∎

Klauck's lower bound (Theorem 4) is a corollary of Theorem 3. Recall that for a Boolean function $f : \{0,1\}^{n_1} \times \{0,1\}^{n_2} \to \{0,1\}$, a set $W \subseteq \{0,1\}^{n_2}$ is shattered, if for all $R \subseteq W$ there is an v such that $f(v,\omega) = 1 \Leftrightarrow \omega \in R$ for all $\omega \in W$. The VC-dimension $VC(f)$ of f is the maximal size of a shattered set.

Theorem 4 ([16]). *For every Boolean function $f : \{0,1\}^{n_1} \times \{0,1\}^{n_2} \to \{0,1\}$, every $\delta \in (0,1/2)$,*

$$Q_\delta(f) = VC(f)(1 - H(\delta)).$$

Proof. In according to definition of $VC(f)$ there exists matrix $M^{V,W} \in \mathcal{M}_f$, where W is shattered set with $|W| = VC(f)$ and $|V| = 2^{|W|}$. ∎

2.1 Quantum OBDD

We use quantum OBDD model defined in [6]. A QOBDD P of width d for Boolean function f on n variables that tests its n variables in order $\pi = (\pi(1) \ldots \pi(n))$ is a triple

$$P = \langle T, |\psi_0\rangle, F\rangle$$

where T is a sequence (of the length n) of d-dimensional unitary transformations of d-dimensional Hilbert space $\mathcal{H}^d$:

$$T = (\langle \pi(i), U_i(0), U_i(1)\rangle)_{i=1}^{n},$$

$|\psi_0\rangle$ is the unitary (in respect to the ℓ_2 norm) vector of $\mathcal{H}^d$ (initial *state* of P). $F \subseteq \{1, \ldots, d\}$ is the set of accepting states.

Computation on P for an input $\sigma = \sigma_1 \ldots \sigma_n \in \{0,1\}^n$ is defined as follows:

1. A computation of P starts from the state $|\psi_0\rangle$. On the i-th step, $1 \leq i \leq n$, of computation P transforms state $|\psi\rangle$ to a state $|\psi'\rangle = U_i(\sigma_{\pi(i)})|\psi\rangle$.
2. After the n-th (last) step of quantum transformation P measures its resulting state $|\psi(\sigma)\rangle$. Measurement is presented by a diagonal zero-one projection matrix M where $M_{ii} = 1$ if $i \in F$ and $M_{ii} = 0$ if $i \notin F$. The probability $p_{accept}(\sigma)$ of P accepting input σ is defined by

$$p_{accept}(\sigma) = ||M|\psi(\sigma)\rangle||^2.$$

Acceptance criteria and complexity. Let $\delta \in (0, 1/2)$. We say that an QOBDD P computes f with *bounded error* (with *δ-error*) if for all $\sigma \in f^{-1}(1)$ the probability of P accepting σ is at least $1 - \delta$ and for all $\sigma \in f^{-1}(0)$ the probability of P accepting σ is at most δ. We call such QOBDD δ-error QOBDD and denote it P_δ.

We denote *width*(P) a dimension d of Hilbert space $\mathcal{H}^d$ of program P. For a Boolean function f we define its *quantum width* $Qwidth_\delta(f)$ to be the width of the best quantum OBDD that computes f with δ-error.

Property 3. *Let $\delta \in (0, 1/2)$. Let QOBDD P computes f with δ-error. Let $\pi = (L, R)$ be a partition of inputs of f between Alice and Bob with L and R defined according to an ordering τ of inputs of P. That is, P can read variables from R only after reading variables from L and cannot read variables from L after starting reading variables from R. Then*

$$width(P) \geq 2^{Q_\delta^\pi(f)},$$

here (and below in the paper) Q_δ^π is communication complexity of function f in respect to partition π of inputs.

The proof of Property 3 uses the same approach as the proof of Lemma 1.

In [32] for *indirect storage access* function ISA_n on $n + \log n$ variables proved that any QOBDD δ-error computed ISA_n has size $2^{\Omega(n/logn)}$. Theorem 3 and Property 3 give the following lower bound.

Property 4. $Qwidth_\delta(ISA_n) \geq 2^{(1-H(\delta))n/logn}$.

Below we consider lower bound for complexity for almost all Boolean functions. Denote by $\mathcal{F}(n)$ the set of all boolean functions $f : \{0,1\}^n \to \{0,1\}$. Let E be some property of functions from $\mathcal{F}(n)$. Denote by $\mathcal{F}^E(n)$ the subset of functions from $\mathcal{F}(n)$ without property E. We say that almost all functions have the property E if

$$\mid \mathcal{F}^E(n) \mid / \mid \mathcal{F}(n) \mid \to 0, \quad \text{as} \quad n \to \infty.$$

Theorem 5. *For almost all functions* $f : \{0,1\}^n \to \{0,1\}$, *for* $\delta \in (0, 1/2)$

$$Qwidth_\delta(f) \; \geq \; 2^{(1-(2+\theta)H(\delta))(n-\log 3n)}.$$

Proof Sketch. For an ordering τ of n variables testing for QOBDD let $\pi = (L, R)$ be a corresponding partition of the set of variables such that $|R| = \lceil \log 3n \rceil$, be a partition of n variables of function f.

Consider communication computation for f for the partition π of inputs. Elementary counting proves that 1) $DC(f_n) = n$ for almost all functions f; 2) for an arbitrary $\theta \in (0,1)$ it holds that $n \leq cs(CM) < (2+\theta)n$ for almost all functions f. Now Theorem 3 gives that for almost all functions, for $\delta \in (0, 1/2)$

$$Q_\delta^\pi(f) \; \geq \; (1-(2+\theta)H(\delta))(n - \log 3n).$$

There are $C_n^{|R|}$ different partitions π of the set of n variables determined by different possible orderings τ of variables testings by QOBDDs. Further counting together with the previous inequality and Property 3 proves that for almost all functions f it holds that $Qwidth_\delta(f) \; \geq \; 2^{(1-(2+\theta)H(\delta))(n-\log 3n)}$. ∎

References

1. F.Ablayev, Lower bounds for one-way probabilistic communication complexity and their application to space complexity, *Theoretical Computer Science*, 157, (1996), 139-159.
2. F. Ablayev, Randomization and nondeterminism are incomparable for ordered-read once branching programs, *in Proceedings of the ICALP'97, Lecture Notes in Computer Science, Springer-Verlag*, 1256, (1997), 195-202.
3. F. Ablayev and M. Karpinski, On the power of randomized branching programs, *in Proceedings of the ICALP'96, Lecture Notes in Computer Science, Springer-Verlag*, 1099, (1996), 348-356.
4. F. Ablayev and M. Karpinski, On the Power of Randomized Ordered Branching Programs *Electronic Colloquium on Computational Complexity*, TR98-004, (1998), available at `http://www.eccc.uni-trier.de/eccc/`
5. F. Ablayev and M. Karpinski, A lower bound for integer multiplication on randomized ordered read-once branching programs. *Information and Computation* 186(1), (2003), 78-89.

6. F. Ablayev, A. Gainutdinova, and M. Karpinski, On the computational power of quantum branching programs. *Proc. FCT 2001*, Lecture Notes in Computer Science 2138: 59–70, 2001.
7. M.Agrawal and T. Thierauf, The Satisfiability Problem for Probabilistic Ordered Branching Programs, *Electronic Colloquium on Computational Complexity*, TR97-060, (1997), available at `http://www.eccc.uni-trier.de/eccc/`
8. R. Boppana and M. Sipser, The complexity of finite functions, *in Handbook of Theoretical Computer Science*, Vol A: Algorithms and Complexity, MIT Press and Elsevier, The Netherlands, 1990, ed. J Van Leeuwen, pp. 757-804.
9. R. Bryant, On the complexity of VLSI implementations and graph representations of boolean functions with applications to integer multiplication, *IEEE Trans. Comput.*, 40 (2), (1991), 205-213.
10. R. Bryant, Symbolic boolean manipulation with ordered binary decision diagrams, *ACM Computing Surveys*, 24, No. 3, (1992), 293-318.
11. A. Borodin, A. Razborov, and R. Smolensky, On lower bounds for read-k-times branching programs, *Computational Complexity* , 3, (1993), 1-18.
12. B. Bolling, M. Sauerhoff, D. Sieling, and I. Wegener, On the power of different types of restricted branching programs, *ECCC Reports 1994*, TR94-025. Available at `http://www.eccc.uni-trier.de/eccc/`
13. S. Jukna, Complexity of Boolean Functions, see *Electronic Colloquium on Computational Complexity*, section Lecture Notes, available at `http://www.eccc.uni-trier.de/eccc/`
14. J. Hromkovic, Communication complexity and parallel computations, *Springer Verlag Press*, 1997.
15. M. Karchmer, R. Raz, and A. Wigderson, Super-logarithmic Depth Lower Bounds Via the Direct Sum in Communication Complexity, *Computational Complexity*, 5, (1995), 191-204.
16. H. Klauck. On quantum and probabilistic communication: La Vegas and one-way protocols. *In the Proc. of the 32nd ACM Symp. Theory of Computing*, 2000.
17. Ilan Kremer, Noam Nisan, Dana Ron: On randomized one-round communication complexity. In *Proceedings of the 27th annual ACM symposium on Theory of computing*, 1995, 596-605.
18. M. Krause, C. Meinel, and S. Waack, Separating the eraser Turing machine classes L_e, NL_e, $co-NL_e$, and P_e, *in Proc. of the MFCS'88, Lecture Notes in Computer Science, Springer-Verlag*, 324, 405-413.
19. E. Kushilevitz and N. Nisan, Communication comple xity, *Cambridge University Press*, 1997.
20. C.Meinel and T.Theobald, Ordered Binary Decision Diagrams and Their Significance in Computer-Aided Design of VLSI Circuits - a Survey, *Electronic Colloquium on Computational Complexity*, TR98-039, available at http://www.eccc.uni-trier.de/eccc
21. M. Nakanishi, K. Hamaguchi, and T. Kashiwabara, Ordered quantum branching programs are more powerful than ordered probabilistic branching programs under a bounded-width restriction, *Proc. 6th Intl. Conf. on Computing and Combinatorics (COCOON)* Lecture Notes in Computer Science 1858: 467–476, 2000.
22. M.A. Nielsen and I.L. Chuang, *Quantum Computation and Quantum Information.* Cambridge University Press, 2000.
23. E. Oklonishnikova, On one lower bound for branching programs, *Electronic Colloquium on Computational Complexity*, TR02-020, available at http://www.eccc.uni-trier.de/eccc

24. S. Ponzio, A lower bound for integer multiplication with read-once branching programs, *Proceedings of the 27-th STOC*, (1995), 130-139.
25. S. Ponzio, Restricted Branching Programs and Hardware Verification, PhD Theses, Massachusetts Institute of Technology, 1995. Available at http://www.eccc.uni-trier.de/eccc
26. A. Razborov, Lower bounds for deterministic and nondeterministic branching programs, *in Proceedings of the FCT'91, Lecture Notes in Computer Science, Springer-Verlag*, 529, (1991), 47–60.
27. R.Raz and P. McKenzie, Separation of the monotone NC hierarchy, *in Proc. of the 38th Annual Symposium on Foundation of Computer Science*, (1997), 234-243.
28. M. Sauerhoff, A lower bounds for randomized read-k-times branching programs *Electronic Colloquium on Computational Complexity*, TR97-019, (1997), available at `http://www.eccc.uni-trier.de/eccc/`
29. P. Savicky, S. Zak, A large lower bound for 1-branching programs, *Electronic Colloquium on Computational Complexity*, Revision 01 of TR96-036, (1996), available at `http://www.eccc.uni-trier.de/eccc/`
30. P. Savicky, S. Zak, A hierarchy for $(1,+k)$-branching programs with respect to k, *Electronic Colloquium on Computational Complexity*, TR96-050, (1996), available at `http://www.eccc.uni-trier.de/eccc/`
31. J. Simon and M. Szegedy, A new lower bound theorem for read-only-once branching programs and its applications, *Advances in Computational Complexity Theory*, ed. Jin-Yi Cai, DIMACS Series, 13, AMS (1993), 183-193.
32. M. Sauerhoff and D. Sieling, Quantum branching programs and space-bounded nonuniform quantum complexity. *ph/0403164, March 2004.*
33. I. Wegener, Efficient data structures for boolean functions, *Discrete Mathematics*, 136, (1994), 347-372.
34. I. Wegener, *Branching Programs and Binary Decision Diagrams.* SIAM Monographs on Discrete Mathematics and Applications, 2000.
35. A. Yao. Some Complexity Questions Related to Distributive Computing. In *Proceedings of the 11th Annual ACM Symposium on the Theory of Computing*, (1979), 209-213.

On the Properties of Asymptotic Probability for Random Boolean Expression Values in Binary Bases

Alexey D. Yashunsky

Department of Discrete Mathematics,
Faculty of Mechanics and Mathematics,
Moscow State University,
Leninskie Gory, Moscow, 119992 Russia

Abstract. The present paper deals with the problem of analyzing the value of a random Boolean expression. The expressions are constructed of Boolean operations and constants chosen independently at random with given probabilities. The dependence between the expression value probability and the constants' probabilities is investigated for different sets of operations. The asymptotic behavior of this dependence is given by a probability function, explicitly obtained through analysis of generating functions for expressions. Special attention is given to the case of binary Boolean operations.

The paper demonstrates some probability function properties and their connection with the properties of Boolean operations used in random expressions.

1 Introduction

Let us first state one simple problem illustrating the issue: suppose we consider Boolean expressions constructed of Boolean operations of disjunction $\vee$ and conjunction &, the constants 0 and 1, and the necessary parentheses. Operations are chosen independently and at random, each with a probability of 1/2, the same for the constants. Every expression of the kind has a value of either 0 or 1 which is easily computed.

The question arises then — what is the probability of expressions with value 1 in the set of all expressions with exactly n operations? The solution to this problem is very simple: probability distribution on constants and operations being uniform and the set of operations ($\vee$ and &) being self-dual, every expression has a dual one, which has the same probability. Hence, a random expression with a fixed number of operations (n) takes values of 0 and 1 with equal probabilities.

A natural generalization of the problem is to consider different sets of operations (that we shall name "bases") and probabilities other than 1/2. The present paper treats mostly the case of binary bases, i.e. bases consisting of operations with at most two arguments.

We consider random expressions with operations from an arbitrary fixed Boolean basis B. Constants in the expression are chosen independently at random with probability p of being equal to 1 and probability $1 - p$ of being equal to 0. We are interested in the probability function $P_1(p)$, which is the limiting probability of a random expression value to be equal to 1, when expression length tends to infinity. We intend to investigate the properties of $P_1(p)$ and the way the behavior of $P_1(p)$ is related to the properties of the basis B under consideration.

The motivation for such an approach is provided by the following model problem. Suppose we have a "black box" that takes Boolean values as input and produces Boolean output. The "black box" takes a certain number of random input bits, performs a series of random Boolean operations on them and produces an output bit. The operations are taken from a certain basis B. The probability of each input bit to be equal to 1 is p, the probability of an output bit to be equal to 1 tends to $P_1(p)$.

Let us now suppose that the basis B is known, while the input probability p is not given, though it is fixed. Is it possible to obtain the value of p through the value of $P_1(p)$?

Another model problem is the following: suppose the basis B is a priori unknown, yet it is given that B contains operations, belonging to a certain set (all operations are binary, for instance). What knowledge could one acquire of the basis, while knowing only its $P_1(p)$ function?

Combinatorial analysis in the present paper is based upon the work of Ph. Flajolet and R. Sedgewik [1], notably the theorem that is made known as the Drmota-Lalley-Woods theorem for asymptotics of algebraic generating functions' coefficients.

Certain problems that are in some sense of a similar nature are treated in the works of P. Savický (see in particular [2]) and the works of B. Chauvin et al. (see in particular [3]). The methods applied in the latter work are partly used in this paper as well, yet the problem under consideration is not the same and the results are of an essentially different nature. Both P. Savický and B. Chauvin et al. consider the probabilities of functions realized by random Boolean formulas, devoting their research mainly to the formulas in the $\{\&, \vee\}$ basis with literals x_i and $\bar{x}_i$.

The problem we are considering here, concerns the behavior of the probabilities for random expression values, instead of functions. The expressions are obtained from formulas in an arbitrary basis by individual constant assignments.

2 Basic Definitions and the Main Theorem

Let us now state the problem formally. From now on we prefer to speak of Boolean functions instead of operations. For the exact definitions of notions concerning Boolean functions used in this paper see Glossary (part 4 of this paper). First we define the concept of a basis.

Definition 1. *A set (or multiset) $B = B_1 \cup B_2 \cup \ldots \cup B_r$, where B_k is a set (multiset) of k-ary Boolean functions and $B_r \neq \emptyset$ is called a basis*[1].

The case of a binary basis corresponds to $r = 2$. Now we define the notion of *Boolean expression* inductively.

Definition 2. *Let B be a basis. The constants* 0 *and* 1 *are Boolean expressions over the basis B. If $\Phi_1, \Phi_2, \ldots, \Phi_k$ are Boolean expressions over the basis B and $g \in B_k$, then $g(\Phi_1, \Phi_2, \ldots, \Phi_k)$ is also a Boolean expression over B.*

Definition 3. *The number of function symbols from the basis B in the expression Φ is called the complexity of Φ and is denoted by $|\Phi|$.*

Let us now define a probability distribution on the set of all expressions of a given complexity n over the basis B. Informally, we regard a random expression as the result of substituting constants 0 and 1 for variables in a random Boolean formula of complexity n with non-repeating variables, considering all formulas of complexity n to be of equal probability. Constants are substituted for variables independently at random with fixed probabilities.

For example, we regard the expression $\Psi = \oplus(\oplus(0, \oplus(1, 1)), 1)$ of complexity 3 over the basis $\{\oplus\}$ as the result of substituting constants for variables in the formula $\oplus(\oplus(x_1, \oplus(x_2, x_3)), x_4)$ with non-repeating variables (here and in the following we assume that variables in such formulas are ordered by their index numbers from left to right).

We now proceed to a formal definition of probabilities for expressions that is intended to match the informal description given above. We assign probability p to the constant 1 and $1 - p$ to the constant 0. For an expression Φ we denote by $\pi(\Phi)$ the product of the probabilities of all constants in this expression.

The value of $\pi(\Phi)$ is interpreted in the terms described above. Let us suppose the expression Φ is the result of substituting constants for non-repeating variables in a certain formula. Then the value $\pi(\Phi)$ is the probability to choose this individual assignment of variables from the set of all possible constant assignments for the considered formula. The sum of $\pi(\Phi)$ for all expressions Φ that are obtained by all possible assignments for a given formula is obviously equal to 1. Hence, the sum of $\pi(\Phi)$ for all expressions Φ of complexity n equals the number N_n of all possible formulas of the same complexity n, i.e. $N_n = \sum\limits_{|\Phi|=n} \pi(\Phi)$.

According to our informal description, the probability to choose an individual formula of complexity n with non-repeating variables is $1/N_n$, while the probability to choose an individual constant assignment for this formula to form the expression Φ is $\pi(\Phi)$. Then the probability of Φ should be $\pi(\Phi)/N_n$. Let us now state this in a formal manner.

[1] Functions of zero variables (B_0) are not included in a basis. The problem for a basis with 0-ary functions is easily reduced to a problem with $B_0 = \emptyset$ and a different value of p.

Definition 4. *The probability $P(\Phi)$ of an expression Φ of complexity n is defined by $P(\Phi) = \pi(\Phi)/N_n$. We define $P_{1,n}(p) = \sum_{|\Phi|=n, \Phi=1} P(\Phi)$, i.e. the sum of probabilities of expressions with value* 1 *and complexity n.*

Obviously, $P_{1,n}(p)$ is the probability of an expression value to be equal to 1 for a random expression of complexity n.

Returning to the above example, we have $\pi(\Psi) = p^3(1-p)$, while $N_3 = 5$ and hence $P(\Psi) = p^3(1-p)/5$. Also in this case $P_{1,3}(p) = 4p(1-p)^3 + 4p^3(1-p)$.

We call a basis *uniform* if all of its functions have the same number of variables. In the case of a uniform basis B, the probability distribution defined above on the set of all expressions of a given complexity over the basis B, in fact, induces implicitly a uniform probability distribution on the basis itself, i.e. all functions of B (counting their multiplicities) have equal probabilities of being placed into a random expression. In the case of a non-uniform basis, the probability distribution of Definition 4 also induces a probability distribution on the basis B that is, in general, more intricate and depends upon n.

The case of a uniform basis B that is a multiset demonstrates an interesting possibility of a somewhat more general approach to random Boolean expressions. Due to the fact that one may include a function several times into B, it becomes possible to model the case where functions in B have arbitrary rational probabilities.

Our major interest is the behavior of $P_{1,n}(p)$ in the limit when n tends to infinity.

Definition 5. *For a Boolean basis B and for any fixed p, $0 \le p \le 1$, we define*

$$P_1(p) = \lim_{n \to \infty} P_{1,n}(p) \ ,$$

if the limit does exist and we say $P_1(p)$ is not defined at the point p otherwise.

The existence of the limit for all p, $0 < p < 1$, as well as an explicit formula for $P_1(p)$, is provided by Theorem 1. The formula that represents $P_1(p)$ depends upon two polynomials, corresponding to the basis B. The first of these is the *basis* polynomial $B(S) = |B_1|S + |B_2|S^2 + \ldots + |B_r|S^r$, and the second is the *characteristic* polynomial $A(T, F)$ in the form of:

$$A(T, F) = \sum_{n=1}^{r} \sum_{i=0}^{n} a_{ni} T^i F^{n-i} \ ,$$

where a_{ni} is the number of unit values among the values of functions from B_n on assignments of weight i (i.e., the assignments with exactly i units and $n - i$ zeroes). For a binary basis, $r = 2$.

Considering $P_1(p)$ a function of the variable p we call it the *probability function* corresponding to basis B.

Definition 6. *Two bases are called equivalent if their $P_1(p)$ functions are identical.*

Two bases that share the basis polynomial and the characteristic polynomial, share the function $P_1(p)$ as well and, therefore, are equivalent. Also, there do exist equivalent bases that have non-identical $B(S)$ and $A(T,F)$ polynomials. Obviously, bases are equivalent iff they are indistinguishable in the sense of the second "black box" question.

We now proceed with the main theorem.

Theorem 1. *Let B be a basis, $B(S)$ its basis polynomial and $A(T,F)$ its characteristic polynomial. Then for any fixed p, $0 < p < 1$, there does exist the limit $P_1(p) = \lim_{n\to\infty} P_{1,n}(p)$. Furthermore,*

$$P_1(p) = \frac{A'_F(\tau, \sigma - \tau)}{\omega^{-1} - A'_T(\tau, \sigma - \tau) + A'_F(\tau, \sigma - \tau)} \ ,$$

where ω and σ are real numbers, which form the unique solution of a system of equations:

$$\begin{cases} \sigma = 1 + \omega B(\sigma) \ , \\ 1 = \omega B'(\sigma) \ , \end{cases} \tag{1}$$

under an additional claim that the value of $|\omega|$ is minimal (among all possible solutions of (1)), while A'_T and A'_F are partial derivatives of $A(T,F)$ and $\tau = \tau(p)$ is a certain uniquely defined algebraic function, satisfying the equation:

$$\tau = p + \omega A(\tau, \sigma - \tau) \ .$$

The proof of Theorem 1 makes use of generating functions for probabilities. A system of polynomial equations describing the relations between generating functions leads to an asymptotic expansion of generating functions' coefficients by means of the Drmota-Lalley-Woods theorem from [1].

It should be noted that the function $\tau(p)$ in Theorem 1 in general is not necessarily uniquely defined by its equation; the equation may as well have several solutions. In order to choose the right solution, one has to make use of certain additional relations, following from the proof of Theorem 1. Fortunately, in the case of binary bases, a simple condition $0 \le \tau \le \sigma$ that follows from combinatorial meaning of τ, allows to make the right choice.

Theorem 1 covers only the values of p that satisfy $0 < p < 1$. The boundary values and the continuity of the function $P_1(p)$ are covered by Theorem 2.

Theorem 2. *The function $P_1(p)$ is continuous for $0 < p < 1$.*

If all functions in B preserve zero, then $P_1(0) = 0$. If all functions in B are linear, depending essentially upon all variables, not preserving zero, then $P_1(0)$ is not defined. In all other cases $P_1(p)$ is continuous at $p = 0$.

If all functions in B preserve unity, then $P_1(1) = 1$. If all functions in B are linear, depending essentially upon all variables, not preserving unity, then $P_1(1)$ is not defined. In all other cases $P_1(p)$ is continuous at $p = 1$.

Theorem 2 is partly based on an assertion that is of interest by itself.

Proposition 1. *The function $\tau(p)$ is continuous, monotonously increasing for $0 < p < 1$ and $\frac{d\tau}{dp} = (1 - \omega A'_T(\tau, \sigma - \tau) + \omega A'_F(\tau, \sigma - \tau))^{-1}$.*

Proposition 1 is mainly a consequence of the Implicit Function Theorem applied to the equation $\tau = p + \omega A(\tau, \sigma - \tau)$.

Let us make a simple but useful observation, which is a direct consequence of Theorem 1. It concerns the values of $P_1(p)$ for dual bases: bases B and B^* are called *dual* if each consists of functions dual to those in the other[2].

Theorem 3. *Let B be a basis and $P_1(p)$ its probability function. Let B^* be the basis dual to B and $P_1^*(p)$ the probability function of B^*. Then*

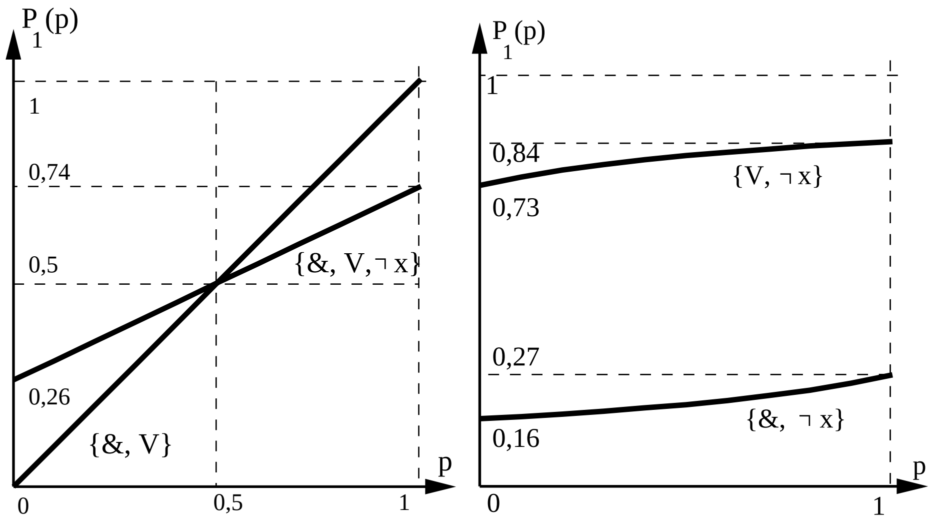
$$P_1(p) = 1 - P_1^*(1-p) \ .$$

The case of self-dual bases follows naturally:

Corollary 1. *Let B be a self-dual basis (i.e., $B = B^*$) and $P_1(p)$ its probability function. Then $P_1(p) = 1 - P_1(1-p)$.*

3 The $P_1(p)$ Function Properties for Binary Bases

The forthcoming analysis will be restricted to binary bases, i.e. the ones with $r = 2$. Some examples obtained by direct application of Theorem 1 are presented below: see Fig. 1 for the graphs of the $P_1(p)$ function of some binary bases, and Table 1 for $P_1(p)$ functions of certain sample binary bases.



$P_1(p)$ for bases $\{\&, \vee, \neg\}$ and $\{\&, \vee\}$ $P_1(p)$ for bases $\{\&, \neg\}$ and $\{\vee, \neg\}$

Fig. 1. The graphs of the $P_1(p)$ function

Note that the graphs on Fig. 1 well illustrate the connection between dual bases and the case of self-dual bases, given by Theorem 3 and Corollary 1.

[2] Taking into account the multiplicity of functions in the case of multiset bases.

Table 1. The $P_1(p)$ function for certain binary bases

Basis	Polynomials	$P_1(p)$ for $0 < p < 1$
All binary and unary functions	$B(S) = 4S + 16S^2$ $A(T,F) = 2T + 2F + 8F^2 + 16TF + 8T^2$	$1/2$
All binary functions	$B(S) = 16S^2$ $A(T,F) = 8F^2 + 16TF + 8T^2$	$1/2$
All binary non-linear functions	$B(S) = 8S^2$ $A(T,F) = 4F^2 + 8TF + 4T^2$	$1/2$
$\{\vee, \downarrow\}$	$B(S) = 2S^2, \quad A(T,F) = F^2 + 2TF + T^2$	$1/2$
$\{\oplus\}$	$B(S) = S^2, \quad A(T,F) = 2TF$	$1/2$
$\{\&, \downarrow, \oplus, \neg\}$	$B(S) = S + 3S^2$ $A(T,F) = F + F^2 + 2TF + T^2$	$\frac{6-\sqrt{3}}{12}$
$\{\&, \vee\}$	$B(S) = 2S^2, \quad A(T,F) = 2TF + 2T^2$	p
$\{\&, \vee, \neg\}$	$B(S) = S + 2S^2$ $A(T,F) = F + 2TF + 2T^2$	$\frac{7+2\sqrt{6}}{25}(p + 3 - \sqrt{6})$
$\{/\}$	$B(S) = S^2$ $A(T,F) = F^2 + 2TF$	$\frac{1}{\sqrt{2+p}}$
$\{\downarrow\}$	$B(S) = S^2$ $A(T,F) = F^2$	$1 - \frac{1}{\sqrt{3-p}}$
$\{\&, \neg\}$	$B(S) = S + S^2$ $A(T,F) = F + T^2$	$\frac{1}{2\sqrt{5+3\sqrt{2}-(3+2\sqrt{2})p}}$
$\{\vee, \neg\}$	$B(S) = S + S^2$ $A(T,F) = F + T^2$	$1 - \frac{1}{2\sqrt{2+\sqrt{2}+(3+2\sqrt{2})p}}$

Let us now write out an explicit formula for $P_1(p)$ in the case of a binary basis. Since the characteristic polynomial has the form:

$$A(T,F) = a_{10}F + a_{11}T + a_{20}F^2 + a_{21}TF + a_{22}T^2 \ ,$$

the function $P_1(p)$ for $0 < p < 1$ is the following:

$$P_1(p) = \frac{a_{10} + 2a_{20}(\sigma - \tau) + a_{21}\tau}{\omega^{-1} + a_{10} - a_{11} + (2a_{20} - a_{21})(\sigma - \tau) + (a_{21} - 2a_{22})\tau} \ , \qquad (2)$$

where ω, σ and the function $\tau = \tau(p)$ are as defined earlier. This explicit form for $P_1(p)$ allows to prove the following statement.

Theorem 4. *For a binary basis B the function $P_1(p)$ is either strictly monotonous or constant for* $0 < p < 1$.

Proof. It follows from (2) that in the case of a binary basis $P_1(p) = \frac{U_0+U_1\tau}{V_0+V_1\tau}$, where U_0, U_1, V_0, V_1 depend upon a_{ni}, ω and σ, but not τ. Let us now differentiate this expression of $P_1(p)$:

$$\frac{d}{dp}P_1(p) = \frac{U_1(V_0 + V_1\tau) - V_1(U_0 + U_1\tau)}{(V_0 + V_1\tau)^2}\frac{d\tau}{dp} = \frac{U_1V_0 - V_1U_0}{(V_0 + V_1\tau)^2}\frac{d\tau}{dp} \ .$$

According to Proposition 1, $\tau(p)$ is a monotonously increasing function, hence its derivative in p is strictly positive. The denominator of the fraction is positive as well. Therefore the derivative's sign depends only upon the sign of $U_1V_0 - V_1U_0$. If $U_1V_0 - V_1U_0 > 0$, then $P_1(p)$ is monotonously increasing; if $U_1V_0 - V_1U_0 < 0$, then $P_1(p)$ is monotonously decreasing. The equality $U_1V_0 - V_1U_0 = 0$ corresponds to a constant $P_1(p)$ function.

Theorem 4, in general, does not hold for bases other than binary: there do exist ternary bases ($r = 3$) for which the function $P_1(p)$ is neither constant nor monotonous.

In the case of binary bases Theorem 4 gives an answer to the first "black box" question, concerning the possibility of obtaining the value of p through the value of $P_1(p)$.

If we restrict the range of p to the interval $0 < p < 1$ then, according to Theorem 4, for a binary basis B, either the value $P_1(p)$ is unique for each p in this interval, or all $0 < p < 1$ share the same value of $P_1(p)$. Thus, depending on the binary basis under consideration, either the value of p is strictly defined by the value of $P_1(p)$, or it is impossible to determine the value of p, given a value of $P_1(p)$.

The two remaining values, $p = 0$ and $p = 1$ need special treatment. If the function $P_1(p)$ is continuous at points $p = 0$ and $p = 1$, then the above statement holds true for the whole interval $0 \leq p \leq 1$. Yet, if $P_1(p)$ has discontinuities at these points, certain corrections have to be made in view of Theorem 2.

We now examine more closely the case of constant $P_1(p)$ functions for binary bases. For the sake of simplicity, we restrict the study to the case $0 < p < 1$.

Theorem 5. *Let B be a binary basis with basis polynomial $B(S)$ and characteristic polynomial $A(T, F)$. Let*

$$d_{ni} = \frac{a_{ni}}{\binom{n}{i}|B_n|} \quad ,$$

where a_{ni} are the characteristic polynomial coefficients and $|B_n|$ are the basis polynomial coefficients[3]*. The function $P_1(p)$ corresponding to the basis B is constant for $0 < p < 1$ iff d_{ni} obey the equality:*

$$\begin{aligned}|B_1| \left[(d_{21} - d_{20})(1 - d_{11}) - (d_{21} - d_{22})d_{10}\right] + \\ +2|B_2|\sigma \left[d_{21}(1 - d_{21}) - d_{20}(1 - d_{22})\right] = 0 \quad .\end{aligned} \tag{3}$$

Theorem 5 follows directly from Theorem 4: (3) is obtained by an easy transformation of the equality $U_1V_0 - V_1U_0 = 0$ from Theorem 4.

Theorem 5 gives an easy criterion for $P_1(p)$ constancy and allows to recognize this property from the basis by simple calculations with the coefficients of the characteristic and the basis polynomials.

It follows directly from Theorem 3 that if a basis B has a constant $P_1(p)$ function, then its dual basis B^* has a constant $P_1^*(p)$ function as well. It is easy to verify that (3) is invariant under the transformation of the basis into its dual.

The following corollaries simplify the $P_1(p)$ constancy conditions for some special cases.

[3] In case $|B_1| = 0$, all $a_{1i} = 0$ as well; we then define $d_{1i} = 0$.

Corollary 2. *Let B be a uniform binary basis. The function $P_1(p)$ is constant for $0 < p < 1$ iff d_{2i} obey the equality:*

$$d_{21}(1 - d_{21}) = d_{20}(1 - d_{22}) \ . \tag{4}$$

Corollary 3. *Let B be a non-uniform binary basis (i.e., $B_1 \neq \emptyset$) and the number σ, corresponding to the basis B, be irrational. Then $P_1(p)$ is constant for $0 < p < 1$ iff d_{ni} obey (4) together with the equality:*

$$d_{10}(d_{22} - d_{21}) = (1 - d_{11})(d_{20} - d_{21}) \ . \tag{5}$$

The conditions of Corollary 3 require the number σ to be irrational. Let us see, when this condition is satisfied. According to Theorem 1, ω and σ are the solution of (1) with the minimal value of $|\omega|$. Solving (1) for a non-uniform binary basis we obtain that the value of σ, corresponding to the minimal value of $|\omega|$, is $\sigma = 1 + \sqrt{1 + \frac{|B_1|}{|B_2|}}$. If the basis is not a multiset, the possible values of $|B_1|$ are $1, 2, 3$ and 4. It is easy to verify that the only case with a rational value of σ is $|B_1| = 3$, $|B_2| = 1$. A straight inspection of all the bases with $|B_1| = 3$ and $|B_2| = 1$ reveals no bases with a constant $P_1(p)$, other than the ones that obey (4) and (5). Still, when B is a multiset, the fulfillment of the conditions (4) and (5) in general is only sufficient for constancy, but not necessary. The following corollaries are other useful and simple sufficient conditions for constancy of $P_1(p)$.

Corollary 4. *Let B be a binary basis with $d_{20} = d_{21} = d_{22}$. Then $P_1(p)$ is constant for $0 < p < 1$.*

Corollary 5. *Let B be a binary basis invariant under negation (i.e., the basis obtained by negating every function in B is identical to B). Then the function $P_1(p)$ is constant for $0 < p < 1$.*

Corollary 4 holds, since the equality of all d_{2i} provides that (4) is satisfied, while (5) becomes the identity $0 = 0$. Corollary 5 follows directly from Corollary 4: if both f and $\bar{f}$ are in B_2 then each d_{2i} is equal to $1/2$.

Corollary 5, in particular, accounts for $P_1(p)$ constancy in the case of the following bases from Table 1: all binary and unary functions, all binary functions, all binary non-linear functions, $\{\vee, \downarrow\}$. Corollary 4 allows a simple verification of constancy for the basis $\{\&, \downarrow, \oplus, \neg\}$.

Using Theorem 5 and its corollaries it is easy to show that equivalent bases do not always share their characteristic and basis polynomials.

Let us consider two bases: $B_\oplus = \{/, \&, \oplus\}$ and $B_\sim = \{/, \&, \sim\}$. Both have the same basis polynomial $B(S) = 3S^2$, while their characteristic polynomials differ: $A_\oplus = F^2 + 4TF + T^2$, $A_\sim = 2F^2 + 2TF + 2T^2$. Yet the ratios d_{2i} for both bases obey the conditions of Corollary 2 and hence for both bases the corresponding function $P_1(p)$ is constant. Moreover, in both cases the constant is $1/2$.

The bases $B_\oplus$ and $B_\sim$ are, in some sense, a better example of equivalent bases than the bases $\{\oplus\}$ and $\{\sim\}$, for instance, though the latter also have

$P_1(p) = 1/2$ for $0 < p < 1$. According to Theorem 2, both $B_\oplus$ and $B_\sim$ possess a function $P_1(p)$ that is continuous at $p = 0$ and $p = 1$, while $\{\oplus\}$ and $\{\sim\}$ have discontinuities at these points.

A consideration of the bases listed in Table 1 and the examples presented above allows us to make several conclusions, concerning the second "black box" question. Neither the basis polynomial, nor the characteristic polynomial are determined by the $P_1(p)$ function. Moreover, in general it is impossible to determine even the arity of functions belonging to the basis, while knowing only the function $P_1(p)$.

4 Boolean Functions Glossary

We present here, for the reader's convenience, a brief summary of basic notions from Boolean functions theory that are used in this paper. An n-ary (or an n-variable) *Boolean function* f is a map $f : \{0,1\}^n \to \{0,1\}$. Truth tables and symbolic notation of binary Boolean functions used in this paper are given in Table 2. The unary function $\neg x$ (or $\bar{x}$) is defined as $\neg 0 = 1$ and $\neg 1 = 0$.

Table 2. Binary Boolean functions

$x\ y$	$x\&y$	$x \vee y$	$x \oplus y$	$x \sim y$	x/y	$x \downarrow y$	$x \to y$
0 0	0	0	0	1	1	1	1
0 1	0	1	1	0	1	0	1
1 0	0	1	1	0	1	0	0
1 1	1	1	0	1	0	0	1

A variable x_i of a Boolean function f is called *dummy* if

$$f(x_1, \ldots, x_{i-1}, 0, x_{i+1}, \ldots, x_n) \equiv f(x_1, \ldots, x_{i-1}, 1, x_{i+1}, \ldots, x_n) \ .$$

If a variable is not dummy, then the function *depends essentially* upon it. A Boolean function f *preserves zero* (*unity*) if $f(0, \ldots, 0) = 0$ (respectively, $f(1, \ldots, 1) = 1$). A Boolean function f is *linear* if it has the form:

$$f(x_1, \ldots, x_n) = \alpha_1 x_1 \oplus \ldots \oplus \alpha_n x_n \oplus \alpha_0 \ ,$$

where $\alpha_0, \alpha_1, \ldots, \alpha_n \in \{0,1\}$.

Boolean functions f and g are *dual* if $f(\bar{x}_1, \ldots, \bar{x}_n) = \bar{g}(x_1, \ldots, x_n)$.

Acknowledgments. The author would like to express his gratitude to his research advisor O.M. Kasim-Zade for posing the problem and helpful advice throughout the work that led to the present paper. The work was supported in part by the Russian Foundation for Basic Research (project no. 05-01-00994), the Support Program for Leading Scientific Schools of the Russian Federation (project no. NS-1807.2003.1), the Program "Universities of Russia" (project no. UR.04.02.528) and the Russian Academy of Sciences Section of Mathematical Sciences Basic Research Program "Algebraic and Combinatorial Methods of Mathematical Cybernetics" (project "Control Systems Optimal Synthesis").

References

1. Ph. Flajolet, R. Sedgewick "Analytic Combinatorics: Functional Equations, Rational and Algebraic Functions". Research Report 4103, INRIA, 2001.
2. P. Savický "Complexity and Probability of some Boolean Formulas". Combinatorics, Probability and Computing, Vol. 7 No. 4, pp. 451-463, 1998.
3. B. Chauvin, P. Flajolet, D. Gardy and B. Gittenberger "And/Or trees revisited". Combinatorics, Probability and Computing, Vol. 13 No. 4-5, pp. 475-497, 2004.

Solving a Dynamic Cell Formation Problem with Machine Cost and Alternative Process Plan by Memetic Algorithms

Reza Tavakkoli-Moghaddam[1], Nima Safaei[2], and Masoud Babakhani[2]

[1] Department of Industrial Engineering, Faculty of Engineering, University of Tehran, P.O. Box: 11365/4563, Tehran, Iran
tavakoli@ut.ac.ir

[2] Department of Industrial Engineering, Iran University of Science and Technology, P.C. 16846/13114, Tehran, Iran
nima.safaei@iust.ac.ir

Abstract. In this paper, we present a new model of a cell formation problem (CFP) for a multi-period planning horizon where the product mix and demand are different in each period, but they are deterministic. As a consequence, the formed cells in the current period may be not optimal for the next period. This evolution results from reformulation of part families, manufacturing cells, and reconfiguration of the CFP as required. Reconfiguration consists of reforming part families, machine groups, and machine relocations. The objective of the model is to determine the optimal number of cells while minimizing the machine amortization/relocation costs as well as the inter-cell movements in each period. In the proposed model, parts have alternative process plans, operation sequence, and produce as batch. The machine capacity is also limited and machine duplication is allowed. The proposed model for real-world instances cannot be solved optimally within a reasonable amount of computational time. Thus, we propose an efficient memetic algorithm (MA) with a simulated annealing-based local search engine for solving the proposed model. This model is solved optimally by the Lingo software then the optimal solution is compared with the MA implementation.

Keywords: Dynamic cell formation, Alternative process plan, Machine relocation, Memetic Algorithm.

1 Introduction

In most industries, the production is dynamic. In other words, the planning horizon can be divided into periods, in which each period has different product mix and demand requirements. In such cases, we face with dynamic production. Note that in the dynamic condition, product mix, and/or demand in each period are different from other periods but it is deterministic (i.e., known as a prior). In a dynamic production condition, the best cell formation (CF) design for one period may not be an efficient design for subsequent periods. By rearranging the manufacturing cells, the CF can continue operating efficiently as the product mix and demand changes. However, it may require some of machines moved from one cell to another cell (i.e., machine relocation) and/or the number of cells changed [1-5]. The CF model belongs to the category of NP-hard problems, in which real-world instances cannot be solved optimality within a reasonable amount of computational time. Thus, the use of metaheuristic is unavoidable.

Some of investigations have been done in content of modeling the dynamic cell formation problem and using its methodology to solve it [6] and [7]. Also, several efforts in similar research areas such as dynamic plant layouts [8] and [9], flexible plant layouts [10] and dynamic layout problems [11], [12] and [13] have been proposed to deal with the dynamic condition. Song and Hitomi [14] have also developed a methodology to design flexible manufacturing cells. The method is to integrate production planning and cellular layout in a long-run planning horizon. Their methodology can determine the production quantity for each part and timing of adjusting the cellular layout in a finite planning horizon period with a dynamic demand. Their objectives are to minimize the sum of inventory-holding cost, group setup cost, material handling cost, and layout adjusting cost. Wilhelm, *et al.* [6] have proposed a multi-period formation of the part family and machine cell formation problem. Their objectives are to minimize the sum of inter-cell material handling cost, investment in additional machines, and cost of reconfiguration over the planning horizon. Harhalaks, *et al.* [15] have used a two-stage design approach to obtain a cellular design with the minimum expected inter-cell material handling cost over a system design horizon which has been broken down into elementary time periods. The most of pervious researches on dynamic cell formation are single objective. The multi-objective cell formation has been considered in single period planning horizon [10] and [16]. Also due to the model complexity, the machine cost, material handling cost, relocation cost along with sequence of operations, and cell size have not been considered in previous researches for a multi-period planning horizon simultaneously.

Evolutionary algorithms have been applied successfully in various domains of search, optimization, and artificial intelligence. They combine the advantages of efficient heuristics incorporating domain knowledge and population-based search approaches. One form of this combination is the use of local search in evolutionary algorithms [17] and [18]. These algorithms, sometimes called ***genetic local search*** algorithms, belong to the class of memetic algorithms (MAs). The generic denomination of MAs is used to encompass a broad class of meta-heuristics. The method is based on a population of agents and proved to be of practical success in a variety of problem domains and in particular for the approximate solution of NP-Hard optimization problems [19]. MAs [19], [20] and [21] are population-based heuristic search approaches for optimization problems similar to genetic algorithms (GAs). MAs are inspired by Dawkins's notion of a meme defined as a unit of information that reproduces itself while people exchange ideas [22]. In contrast to genes, memes are typically adapted by the people who transmit them before they are passed on to the next generation. According to Moscato and Norman [23], "memetic evolution" can be mimicked by combining evolutionary algorithms with local refinement strategies such as local neighborhood search or simulated annealing. However, GAs rely on the concept of ***biological evolution*** and MAs, in contrast, mimic ***cultural evolution***. While in nature, genes are usually not modified during an individual's lifetime, memes are [22]. MAs are based on a population of agents. However, the method is less constrained since it does not use any biological metaphor that will restrict the design of its components. Unlike the traditional evolution methods, MAs are primarily concerned with exploiting all available knowledge about the problem studied [20] and [21]. As a consequence, each agent can make use of previous knowledge of solution results. This knowledge can be optimal solution proprieties, heuristics, truncated exact algorithms, and the like. Different agents may use different algorithms, even having different behaviours that parameterize how they recombine solutions [24]. Also, the agents can use different meta-heuristics for the individual search processes, or independently adapt the search based on their own historical information [25]. A number of investigations have been carried out in the content of applications of MAs; such as minimum sum-of-squares clustering problem [26], unconstrained binary quadratic programming problem [27], multistage capacitated lot-sizing problem [28], arc routing problems [29] and asymmetric travelling salesman problem [30].

2 Problem Formulation

In this section, we present the proposed dynamic cell formation model. The model expanded under the following goals:

1. Establish the part families and machine groups simultaneously.
2. Determine the optimum number of cells in each period.
3. Determine the optimal alternative process plan for each part in each period.
4. Determine the number of needed machines for each type in each period.
5. Add, remove, or relocate machines in cells between each two consecutive periods.

The dynamic cell formation problem is considered under the following assumptions.

2.1 Assumptions

The assumptions for the above-mentioned problem are given bellow:

1. The process time for all part type operations on different machine types are known.
2. Each part must be processed according to a known sequence of operations.
3. The demand for each part type in each period is known.
4. The capabilities and capacity of each machine type is known and constant over time period.
5. Amortization cost in each period to procure one machine of each type is known.
6. Parts are moved between cells in batches. The inter-cell material handling cost for each batch between cells is known and constant (independent of quantity of cells).
7. The maximum number of cells used must be specified as a prior.
8. Bounds and quantity of machines in each cell need to be specified in advance and they remain constant over time.

9. Machine relocation from one cell to another is performed between periods and it requires zero time. The machine relocation cost of each machine type is known and it is independent of where machines are actually being relocated.
10. Each machine type can perform one or more operations (i.e., machine flexibility). Likewise each operation can be done on one machine type with different times (i.e., routing flexibility).
11. Inter-cell handling costs are constant for all moves regardless of the distance traveled.
12. No inventory is considered.
13. Setup times are not considered.
14. Backorders are not allowed. All demand must be satisfied in the given period.
15. No queue in production is allowed.
16. Machine breakdowns are not considered.
17. Batch size is constant for all productions and all periods.
18. Machines are available at the state of the period (i.e., zero installation time).

2.2 Notation

Indexes:

c	Index for manufacturing cells (c=1, …, C)
m	Index for machine types (m=1, …, M)
p	Index for part types (p=1, …, P)
h	Index for time periods (h=1, …, H)
j	Index for operations required by part p (j=1, …, O_p)

Model Inputs:

P	Number of part types.
O_p	Number operations of part p.
M	Number of machine types.
C	Maximum number of cells that can be formed.
D_{ph}	Demand for product p in period h.
B	Batch size for inter-cell material handling.
α_m	Amortization (or depreciation) cost of machine of type m.
γ	Inter-cell material handling cost each batch.
δ_m	Relocation cost of machine type m.
T_m	Capacity of each machine of type m (hours).
LB	Lower bound cell size.
UB	Upper bound cell size.
t_{jpm}	Time required to perform operation j of part type p on machine type m.

$$a_{jpm} \begin{cases} 1 & \text{if operation } j \text{ of part } p \text{ can be done on machine type } m \\ 0 & \text{otherwise} \end{cases}$$

Ψ_{jp}	Set of operating machines for operation j of part type p, where $\Psi_{jp} = \{m \mid a_{jpm} = 1\}$

Decision variables:

N_{mch}	Number of machines of type m used in cell c during period h.
K^{+}_{mch}	Number of machines of type m added in cell c during period h.

K^{-}_{mch} Number of machines of type *m* removed from cell *c* during period *h*.

$$X_{jpmch}\begin{cases}1 & \text{if operation } j \text{ of part type } p \text{ is done on machine type } m \text{ in cell } c \text{ in period } h.\\ 0 & \text{otherwise}\end{cases}$$

$$Z_{jpch}\begin{cases}1 & \text{if operation } j \text{ of part type } p \text{ is done in cell } c \text{ in period } h.\\ 0 & \text{otherwise}\end{cases}$$

$$Y_{ch}\begin{cases}1 & \text{if cell } c \text{ formed in period } h\\ 0 & \text{otherwise}\end{cases}$$

2.3 Mathematical Formulation

By using the above notation, the objective function and constraints are now written in the following equation forms.

$$\min Z=\sum_{h=1}^{H}\sum_{m=1}^{M}\sum_{c=1}^{C}N_{mch}\alpha_m+\frac{1}{2}\sum_{h=1}^{H}\sum_{p=1}^{P}\left\lceil\frac{D_{ph}}{B}\right\rceil\times\sum_{j=1}^{O_p-1}\sum_{c=1}^{C}\gamma\left|Z_{(j+1)pch}-Z_{jpch}\right|+\frac{1}{2}\sum_{h=1}^{H}\sum_{m=1}^{M}\left(k^{+}_{mch}+k^{-}_{mch}\right)\delta_m \tag{1}$$

s.t:

$$\sum_{c=1}^{C}\sum_{m=1}^{M}a_{jpm}x_{jpmch}=1 \qquad \forall j,p,h \tag{2}$$

$$\sum_{p=1}^{P}\sum_{j=1}^{Op}D_{ph}t_{jpm}x_{jpmch}\le T_mN_{mch} \qquad \forall m,c,h \tag{3}$$

$$\sum_{m=1}^{M}N_{mch}+\sum_{m=1}^{M}K^{+}_{mch}-\sum_{m=1}^{M}K^{-}_{mch}\ge LB\times Y_{ch} \qquad \forall c,h \tag{4}$$

$$\sum_{m=1}^{M}N_{mch}+\sum_{m=1}^{M}K^{+}_{mch}-\sum_{m=1}^{M}K^{-}_{mch}\le UB\times Y_{ch} \qquad \forall c,h \tag{5}$$

$$N_{mc(h-1)}+K^{+}_{mch}-K^{-}_{mch}=N_{mch} \qquad \forall m,c,h \tag{6}$$

$$Z_{jpch}=\sum_{m=1}^{M}x_{jpmch} \qquad \forall j,p,c,h \tag{7}$$

$$Y_{ch}\ge Z_{jpch} \qquad \forall j,p,c,h \tag{8}$$

$$x_{jpmch},\ Z_{jpch},\ Y_{ch}=0 \text{ or } 1,\quad N_{mch},K^{+}_{mch},K^{-}_{mch},D_{ph}\ge 0 \text{ and Integer}$$

The objective function given in Equation (1) is a nonlinear integer model. It is to minimizing the total sum of the machine amortization cost, the inter-cell material handling cost, and the machine relocation cost over the planning horizon. The first term represents the cost of all machines required in all periods. The machine amortization (or depreciation) cost is obtained by the product of the number of machine type *m* in cell *c* in period *h* and their respective costs. The second term is the cost of operating machines. It is the sum of the product of the number of hours for each machine type and their respective costs. The third term is the inter-cell material handling costs. The total cost is obtained by summing the products of the number of inter-cell transfer for each part type and the cost of transferring a batch of each part type. The next cost is the machine relocation cost. It is the sum of the product of the number of machines relocated and their respective cost. Equation (2) guarantees that each part operation is assigned to one machine and one cell. Equation (3) ensures that machine capacities are not exceeded and can satisfy the demand. Equations (4) and (5) specify the lower and upper bounds of cells. Equation (6) ensures that the number of machines in the current period is equal to the number of machines in the previous, plus the number of machines being moved in, and minus the number of machines being moved out. In other words, they ensure the conservation of machines over the horizon. In equation (7), if at least one of the operations of part *p* is processed in cell *c* in period *h* then the value of z_{jpch} will be equal to 1, otherwise is set to zero. This constraint is used for calculating inter-cell material handling in the third term of the objective function. Equation (8) determines the number of cells formed in period *h*.

3 Memetic Algorithm Implementation

Memetic algorithms can be thought of as a special kind of genetic search over the subspace of local optima, within which a local optimizer is added to genetic algorithms and applied to every offspring before it is inserted into the population. Crossover and mutation will usually produce solutions that are outside this space of local optima, but a local optimizer can then repair such solutions to produce final offspring that lie within this subspace. In this section, a memetic algorithm with a SA-based local search engine is proposed to solve the considered dynamic cell formation problem. The chromosome representation, adaptive penalty function, genetic operators, local search engine, and selection method of the proposed memetic algorithm are explained.

3.1 Chromosome Representation

The first step in each evolutionary programming is to determine the chromosome representation or solution structure. In this paper, we represent a solution *S* by a three-dimensional structure consisting of two matrices $[X]_{p\times r}$ and $[Y]_{p\times r}$ in each period (e.g., $S^h = [X^h \mid Y^h] \Rightarrow S = [S^1, S^2, \ldots, S^h]$) as shown in Fig. 1. The matrix *X* denotes the allocation of operation-part to machine and the matrix *Y* denote the allocation of operation-part to cell. This idea is taken from Tavakkoli-Moghaddam, *et al.* [31]. $R = \max_p\{O_p\}$ and x^h_{pr} is equal to the machine that operation *r* part *p* must be processed on, where $x^h_{pr} \in \Psi_{rp}$. Also, y^h_{pr} is equal to the cell that operation *r* part *p* allocated to where $1 \le y^h_{pr} \le C$. Because of $O_p \le R\ \forall p$, some of entries in solution representation are equal to zero inherently.

$$\left(\begin{array}{cccc|cccc}
x_{11}^{h} & x_{12}^{h} & \cdots & x_{1R}^{h} & y_{11}^{h} & y_{12}^{h} & \cdots & y_{1R}^{h} \\
x_{21}^{h} & x_{22}^{h} & \cdots & x_{2R}^{h} & y_{21}^{h} & y_{22}^{h} & \cdots & y_{2R}^{h} \\
\vdots & & & & \vdots & & & \\
x_{P1}^{h} & x_{P2}^{h} & \cdots & x_{PR}^{h} & y_{P1}^{h} & y_{P2}^{h} & \cdots & y_{PR}^{h}
\end{array}\right)$$

Fig. 1. Chromosome representation in period *h*

3.2 Genetic Operators

3.2.1 Recombination Operators

We use three kinds of crossover operators: machine level, cell level and conventional. The conventional crossover is accomplished by selecting two parents and swapping corresponding matrixes from parents to form the offspring as shown in Fig. 2. In machine level crossover, one cut point selected randomly on matrix *X* in vertical or horizontal direction then the partial matrixes from parents swapped together as shown in Fig. 3. The cell level crossover is similar to machine level except for matrix *Y*.

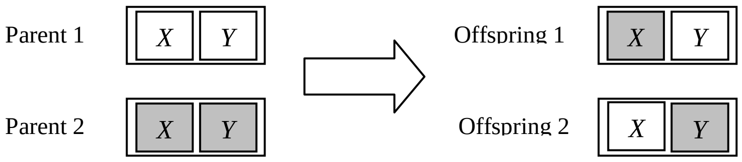


Fig. 2. Conventional crossover implementation

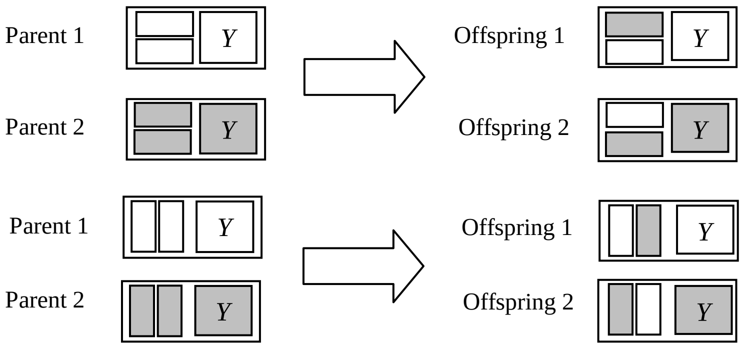


Fig. 3. Machine level crossover implementation

3.2.2 Mutation Operators

For preserving diversity in generations and for escaping from local optimums, we use three mutation operators as follows: single-mutation, multi-mutation, and inversion operators. In single-mutation, an operation from a part (e.g. part-operation [$p\times r$]) is selected at random then its machine (x^{h}_{pr}) or cell (y^{h}_{pr}) is changed as $x^{h}_{pr} \to x^{h\prime}_{pr} \in \Psi_{rp}$

or $y^h_{pr} \rightarrow y^{h\prime}_{pr}$ randomly. In multi-mutation, a part (e.g. part p) is selected at random then the single-mutation is implemented on all its operations in matrix X or Y. In inversion operator, a part (e.g. part p) selected at random then the corresponding row p in matrix Y are inversed as $y^h_{pr} \rightarrow y^h_{p(R-r+1)}$; $1 \leq r < \lceil R/2 \rceil$. To implement each operator, the value of decision variable N_{mch} must be updated with respect to the entries of matrixes X and Y. It can be done as Equation (9) then decision variables K^+_{mch} and K^-_{mch} must be updated by Equations (10) and (11).

$$N_{mch} = \left\lceil \frac{\sum_{x^h_{jp}=m,\, y^h_{jp}=c} t_{jpm}}{T_m} \right\rceil \qquad \forall m,c,h \tag{9}$$

$$K^+_{mch} = \max\left\{N_{mch} - N_{mc(h-1)}, 0\right\} \qquad \forall m,c,h>1 \tag{10}$$

$$K^-_{mch} = \max\left\{N_{mc(h-1)} - N_{mch}, 0\right\} \qquad \forall m,c,h>1 \tag{11}$$

3.3 Adaptive Penalty Function

Because of violating constraints (4) or (5) the genetic operators and local search used to manipulate the chromosomes often yield infeasible offspring. The penalty technique is perhaps the most common one used to handle infeasible solutions in the genetic algorithms for constrained optimization problems. The penalty technique is used to keep a certain amount of infeasible solutions in each generation so as to enforce genetic search towards an optimal solution from both sides of feasible and infeasible regions [32]. Adaptive penalty function is used to handle constraints. The fitness function is composed of two parts: 1) the objective function, i.e., Equation (1), 2) penalties for violating lower and upper bounds cell capacity constraints, i.e., Equations (4) and (5). The adaptive penalty function is constructed with variable penalty coefficients and the penalty for violation of each constraint. The fitness function is defined as follows:

$$\begin{aligned}\min Z = {} & \sum_{h=1}^{H}\sum_{m=1}^{M}\sum_{c=1}^{C} N_{mch}\alpha_m + \frac{1}{2}\sum_{h=1}^{H}\sum_{p=1}^{P}\left\lceil \frac{D_{ph}}{B} \right\rceil \times \sum_{j=1}^{O_p-1}\sum_{c=1}^{C} \gamma \left| Z_{(j+1)pch} - Z_{jpch} \right| + \frac{1}{2}\sum_{h=1}^{H}\sum_{m=1}^{M}\left(k^+_{mch} + k^-_{mch}\right)\delta_m + \\ & + A_1(g) \times \max\left\{ LB - \left(\sum_{m=1}^{M} N_{mch} + \sum_{m=1}^{M} K^+_{mch} - \sum_{m=1}^{M} K^-_{mch} \right),\ 0 \right\} + \\ & + A_2(g) \times \max\left\{ \left(\sum_{m=1}^{M} N_{mch} + \sum_{m=1}^{M} K^+_{mch} - \sum_{m=1}^{M} K^-_{mch} \right) - UB,\ 0 \right\}\end{aligned} \tag{12}$$

Where $A_1(g)$ and $A_2(g)$ denote the penalty coefficients for Lower and upper bounds constraints in the g-th generation, respectively. The quality of solutions heavily depends on the values of these penalty coefficients [33]. If the penalty coefficients are moderate, the algorithm may converge to an infeasible solution. On the other hand, if the penalty coefficients are too large, the method is equivalent to a rejecting strategy. In order to obtain appropriate coefficient values, we use a set-and-test approach to dynamically adjust these coefficients according to the number of infeasible solutions in each generation.

The behavior of genetic search is characterized by a balance between exploitation and exploration in the search space. Here, more infeasible solutions are indicated at the beginning of the memetic algorithm search in favor of a wide exploration of the search space, while less infeasible solutions are recommended at the end in order to exploit the most promising regions in the feasible search space. Hence, the allowable number of infeasible solutions in each generation decreases in steps of the main loop. The dynamic of the resource penalty coefficient is described below:

$$A_1(g) = (1+\alpha)\times A_1(g-1)\ ; \quad \alpha = \left[R_1(g) - R_1(g-1)\right]/R_1(g-1) \tag{13}$$

$$A_2(g) = (1+\beta)\times A_2(g-1)\ ; \quad \beta = \left[R_2(g) - R_2(g-1)\right]/R_2(g-1) \tag{14}$$

Where $R_1(g)$ is the total number of infeasible solutions with respect to lower bound cell capacity constraint in generation g. Also, $R_2(g)$ is the total number of infeasible solutions with respect to the upper bound cell capacity constraint in generation g. Parameters α and β denote the percent change of the number of infeasible solutions in current generation with respect to lower and upper bound cell capacity constraint. If the current number of infeasible solutions is reduce by η percent then decrease the penalty coefficient value by η percent.

3.4 Local Search Engine (LSE)

The local search engine is one of the basal components in memetic algorithms. As said earlier, it implicates to cultural evolution. It is referred to as a local search, since it starts with a feasible solution and, using moves (such as mutation), it searches a neighborhood for another feasible solution with lower cost. The neighborhood of a solution is the set of all solutions that can be reached with a move. If a better solution is found, the current solution is replaced and the neighborhood search is started again. If no further improvement can be made, a local optimal is found. It means that there is no better solution in the neighborhood of the current solution. The balance between genetic search and local search has a significant effect on the search ability of memetic algorithms [34]. One crucial problem of memetic algorithm is how to properly allocate the available computation time wisely between genetic search and local search.

We use a heuristic local search based on a semi-annealing approach in which there is only outside loop (instead of two inside and outside loops) that controlled by three parameters T, α, and δ. Parameters T and α denote the temperature and cooling schedule like to simulated annealing (SA) and δ indicates the relative fitness of recently obtained neighborhood. The computation time allocated to local search depend on initial setting of above parameters. We found the best parameter settings as T=1, δ = 0.1 and α = 0.9. In optimistic status, if in first iteration **Fit(***Tmp_X*) ≤ 0.5×**Fit (***New_X* **)** then δ = 1 and Do-Until loop iterate only one times. In pessimistic status, if it is not found any better neighborhood then ***Do-Until*** loop iterate r times where $r = [ln(\alpha)\ /\ ln(\delta)] \cong 22$ times. The proposed algorithm causes a high computation time is not allocated to local search. A pseudo code of the proposed local search is shown in Fig. 4.

```
 Set $T$=1, $\delta$= 0.1, $\alpha$= 0.9
 Get solution $X$.
 SET $Tmp\_X=\varnothing$, $New\_X=X$
 DO
    Select a kind of mutation at random with specific rate.
    $Tmp\_X$ = MUTATION($X$)
    IF Fit($Tmp\_X$) < Fit ($New\_X$ ) THEN
      $New\_X = Tmp\_X$
      $\delta$ = (Fit ($New\_X$) - Fit ($Tmp\_X$))/ Fit ($Tmp\_X$)
    END IF
    $T= \alpha\times T$
LOOP UNTIL($T \le \delta$)
$X$ = $New\_X$
```

Fig. 4. Pseudo code of the local search engine

3.5 Stoppage Conditions

We use three criteria for stopping the algorithm as follows: 1) Maximum number of the established generation (G). 2) Least variance of the generation (π). 3) Maximum run time for the algorithm (MRT).

Fig. 5 shows the pseudo code of the MA that we have implemented. The **INITIALIZE_POPULATION()** procedure is used for generating a initial population in which individuals are created randomly then improved by **LOCAL_SEARCH_ENGIN()** filter. The population diversity is controlled by a dissimilarity mechanism that calculates the similarity rate for each new individual. The similarity rate is equal to the number of similar genes (i.e., genes with same allele and locus) in new individual and other individuals in current population divided to total number of genes in chromosome structures, i.e., $2\times P\times R$.

```
INITIALIZE_POPULATION()
DO (outside loop)
 Set $i$=0, $g$ =0
 DO (inside loop)
   Select randomly $parent1$ and $parent2$.
   $offspring$ = CROSSOVER($parent1$,$parent2$)
   $offspring$ = LOCAL_SEARCH_ENGIN($offspring$)
   $Similarity\_Rate$= DIVERSITY($offspring$)
   IF $Similarity\_Rate$ <1 THEN  Accept $offspring$ AND Set $i$=$i$+1
 LOOP UNTIL($i = K$)
 Set $g$= $g$+1
LOOP WHILE($g$<$G$)
Print best solution
```

Fig. 5. Pseudo code of the MA

4 Computational Experience

Ten instances have been solved by the Lingo 6.0 software for verifying the proposed model in the linear case. The CPU times is corresponded to an intel® Celeron®

mobile 1.3 GHz processor with 512 MB RAM. The test problems are randomly generated in terms of uniform distribution. For simplicity, a number of parameters are set in advance as shown in Tables 1 and 2. The value of $\Sigma_m a_{jpm}$ denotes the number of operative machines for operation *j* and part *p*. It is obvious that by increasing $\Sigma_m a_{jpm}$ the solution space increases progressively. The parameters initialized in Table 2 are set experimentally. Table 3 includes the comparison of the results obtained from the Lingo solver and MA algorithm in terms of the objective function value (OFV), CPU time, and cell size for small-sized problems. The "Cell Size" column shows the optimum number of cells formed in each period. The "Var" column denotes the number of model variables and the "Cons" column denotes the number of linearization constraints. The columns 'T_m' and '*UB*' are considered for the sensitivity analysis. The capacity is same for all machines in each problem. As shown in Table 3, the model is very sensitive to *UB* and T_m where by decreasing T_m the CPU time increases progressively such as problems 5 and 10. Also by increasing *UB*, the CPU time decreases such as problems 5 and 6 or by decreasing *UB*, it is possible that the number of formed cells increases such as problems 5 and 10. The average gap between optimum and MA runs is equal to 3.67 percent that is very promising and satisfactory. Two instance cells formed in optimum and MA solutions related to problems 1 and 4 are shown in Figures 6 through 9 respectively. As shown in Fig. 6, machine 3 is removed from cell 3 in period 1 and machine 2 is added to cell 3 in period 2. Machines 4 and 6 are duplicated in both periods. The part families for MA

Table 1. Model parameter settings for small-sized problems

Parameter	γ	*B*	*LB*	*C*	$\Sigma_m a_{jpm}\ \forall j,p$	α_m	δ_m
Value	40	50	1	3	2	U(1000, 10000)	$0.5\times\alpha_m$

Table 2. MA parameter settings

	Stoppage conditions			Population size	Tuning		
Parameter	*G*	*MRT*	π	*K*	Crossover	Mutation	Inversion
Value	100	1 hour	5	100	0.8	0.1	0.1

Table 3. Comparison of optimum and MA run for small/medium-sized problems

Test Problem						Optimum Solution			MA run			
No.	Var	Cons	$P(\Sigma_p O_p)$ $\times M \times H$	T_m	*UB*	O.F.V	CPU Time	Cell Size	O.F.V	CPU Time (Sec.)	Cell Size	Gap (%)
1	915	259	7(12)×6×2	3000	3	67234	00:00:56	3-3	67995	30	3-3	1.13
2	915	259	7(12)×6×2	4000	4	57989	00:01:04	2-2	58285	30	2-2	0.51
3	636	302	8(18)×7×2	8000	4	50314	00:01:30	1-2	56901	177	3-3	13
4	735	307	6(13)×6×2	5000	4	36044	00:01:51	2-2	36740	50	2-2	1.9
5	759	406	7(19)×7×2	15000	5	52250	00:07:23	1-1	54000	81	1-1	3.34
6	759	406	7(19)×7×2	15000	3	54380	00:08:54	2-3	57390	58	1-2	5.53
7	1167	314	8(14)×8×2	6000	5	51744	00:13:30	2-2	54430	42	3-3	5.19
8	594	315	5(9)×5×3	7000	4	72602	00:16:24	1-1-2	72602	148	1-1-2	0
9	636	302	8(18)×7×2	6000	4	61010	00:28:16	2-2	61260	371	3-3	4
10	759	406	7(19)×7×2	12000	5	60720	01:30:06	2-2	62000	217	2-2	2.1
									Average gap of O.F.V:			3.67

Period 1			C1	C2	C3		
			P7	P3	P1	P2	P5
C1	3	M6	1				
C2	3	M4		2,3			
C3	1	M3		1	1		
	1	M5				1	1

Period 2			C1		C2	C3	
			P4	P7	P6	P2	P5
C1	3	M6	1	1			
C2	3	M4			3		
C3	1	M2	2		2		
	1	M5			1	1	1

Fig. 6. The formed cells in optimum solution for problem 1 in Table 1

Period 1			C1	C2			C3
			P7	P1	P2	P5	P3
C1	3	M6	1				
C2	2	M2		1			2
	1	M5			1	1	
C3	1	M3					1
	2	M4					3

Period 2			C1		C2	C3	
			P4	P7	P2	P5	P6
C1	3	M6	1	1			
C2	1	M2	2				2
	1	M5			1	1	1
C3	3	M4					3

Fig. 7. The formed cells in MA solution for problem 1 in Table 1

Period 1			C1			C2
			P3	P4	P5	P1
C1	2	M4	1		1	
	1	M5	2	1		
C2	1	M2	3		2	1,3
	1	M3			3	2

Period2			C1			C2	
			P4	P5	P6	P1	P2
C1	2	M4		1			2
	1	M5	1		1		3
C2	2	M2		2	2	1,3	
	1	M3		3		2	1

Fig. 8. The formed cells in optimum solution for problem 4 in Table 1

Period 1			C1		C2	
			P3	P5	P1	P4
C1	2	M4	1	1		
C2	1	M2		2	1,3	
	1	M3		3	2	
	1	M5	2			1

Period2			C1	C2			
			P5	P1	P2	P4	P6
C 1	1	M2	2				
	2	M4	1		2		
C 2	1	M2		1,3			2
	1	M3	3	2	1		
	1	M5			3	1	1

Fig. 9. The formed cells in MA solution for problem 4 in Table 1

Table 4. Parameter settings for large-sized problems

Parameter	γ	B	LB	UB	C	$\Sigma_m a_{jpm}\ \forall j,p$	α_m	δ_m	T_m
Value	40	50	2	$\lceil M/2 \rceil$	5	2	U(1000, 10000)	$0.5\times\alpha_m$	10000

Table 5. MA solutions for large-sized problems

Test Problem		Initial Solution	Final Solution			
No.	$P(\Sigma_p O_p)\times M\times H$	O.F.V	O.F.V	CPU Time (Sec.)	Number of movements	Rate of reduction (%)
1	20(37)×10×2	368661	113665	688	66	69
2	30(61)×15×2	378954	175456	1222	82	53
3	40(79)×20×2	633054	248805	1566	116	57
4	50(102)×25×2	630857	272208	1944	148	56
5	100(203)× 05×2	1556249	633199	4295	254	59

and optimum solutions are same in both periods whereas the group machines are similar in period 1 and same in period 2.

Table 3 includes the results obtained from the MA algorithm for large-sized problems in which they cannot be solved by optimal methods on personal computers. In Table 3, the initial and final solutions are compared with respect to the OFV. The initial solution implicates to the solution obtained in time zero. The number of movements used for achieving to final solution is considered in column "Number of movements". The relative different between initial and final solutions is shown in column "Rate of reduction".

5 Conclusions

In this paper, we have proposed a new multi-objective cell formation (CF) model with assuming dynamic production, alternative process plan, sequence operation, and machine relocation. The main advantages of the proposed model are to form parts family and machine groups simultaneously, determine the optimum number of cells in each period, determine the best processing route for each part in each period, and relocate machines between two consecutive periods as required. Due to the complexity of the proposed model, a memetic algorithm is introduced for solving the proposed CF model by a matrix form representation and several effective specialized genetic operators. The proposed MA finds near-optimal solutions in a reasonable amount of the CPU time. The violation of any assumptions in Section 2.1 can be investigated in the future researches.

References

1. Wemmerlov, U. Hyer, N.: Cellular manufacturing in the US industry: A survey of users. Int. J. of Production research, **27** (1989) 1511-1530.
2. Schaller, J.E., Erenguc, S.S., Vakharia, A.J.: A mathematical approach for integrating the cell design and production planning decision. Int. J. of Production Research, **38** (2000) 3953–3971.
3. Chen, M.: A model for integrated production planning in cellular manufacturing systems. Integrated Manufacturing Systems, **12** (2001) 275–84.
4. Foulds, L.R., Neumann, K.: Techniques for machine group formation in manufacturing cells. Mathematical and Computer Modeling, **38** (2003) 623–635.
5. Shafer, S., Rogers, D.: A goal programming approach to the cell formation problem. J. of Operations Management, **10** (1991): 28-43.
6. Wilhelm, W., Chou, C., Chang, D.: Integrating design and planning considerations in cell formation. Annals of Operations Research, **77** (1998) 97-107.
7. Chen, M.: A mathematical programming model for systems reconfiguration in a dynamic cell formation condition, Annals of Operations Research, **77** (1998) 109-128.
8. Monteruiln, B., Laforge, A.: Dynamic layout design given a scenario tree of probable future. Eur. J. of Operational Research, **63** (1992) 271-286.
9. Rogers, G., Bottaci, L.: Modular production systems: A new manufacturing paradigm. J. of Intelligent Manufacturing, **8** (1997) 147-156.
10. Black, J.T.: The design of the factory with a future. McGraw-Hill, New York (1991).

11. A. Baykasogylu and N. Gindy. A simulated annealing algorithm for dynamic layout problem, Computers and Operation Research, **28** (2001) 1403-1426.
12. Lacksonen, T.A.: Static and dynamic layout problems with varying areas. J. of Operational Research Society, **45** (1994) 59-69.
13. Lacksonen, T.A.: Preprocessing for static and dynamic layout problems. Int. J. of Production Research, **35** (1997) 1095-1106.
14. Song, S., Hitomi, K.: Integrating the production planning and cellular layout for flexible cellular manufacturing. Int. J. of Production Planning and Control. **7** (1996) 585-593.
15. Harhalaks, G., Nagi, R., Proth, J.: An effective heuristic in manufacturing cell formation for group technology applications. Int. J. of Production Research. **28** (1990) 185-198.
16. Caux, C., Bruniaux, R., Pierreval, H.: Manufacturing cell formation with alternative process plans and machine capacity constraints: a new combined approach. Int. J. of Production Economics. **64** (2000) 279–84.
17. Kollen, A., Pesch, E.: Genetic local search in combinatorial optimization. Discrete Applied Mathematics and Combinatorial Operation Research and Computer Science 48 (1994) 273– 284.
18. Merz, P., Freisleben, B.: Fitness landscapes and memetic algorithm design, in: Corne, D., Dorigo, M., Glover, F. (Eds.), New ideas in optimization, McGraw-Hill, London (1999).
19. Moscato, P., Cotta, C.: A gentle introduction to memetic algorithms, in: Glower, F., Kochenberger, G. (Eds.), Handbook of metaheuristics, Kluwer, (1999) 1–56.
20. Moscato, P.: On evolution, search, optimization, genetic algorithms and martial arts: toward memetic algorithms, Caltech Concurrent Computation Program, California Institute of Technology, Pasadena, Technical Report 790 (1989).
21. Moscato, P.: A memetic approach for the traveling salesman problem implementation of a computational ecology for combinatorial optimization on message-passing systems, in: Valero, M., Onate, E., Jane, M., Larriba, J.L., Suarez, B. (Eds.), Parallel computing and transporter Applications, IOS Press, Amsterdam, The Netherlands, (1992) 176–177.
22. Dawkins, R.: The selfish gene. Oxford University Press, Oxford (1976).
23. Moscato, P., Norman, M.G.: A memetic approach for the Traveling Salesman Problem implementation of a computational ecology for combinatorial optimization on message-passing systems. In: Valero, M., Onate, E., Jane, M., Larriba, J.L., Suarez, B. (Eds.), Parallel computing and Transporter Applications. IOS Press, Amsterdam, The Netherlands, (1992). 177–186.
24. Berretta, R.E., Moscato, P.: The number partitioning problem: An open challenge for evolutionary computation? In: Corne, D., Glover, F., Dorigo, M. (Eds.), New ideas in optimization. McGraw-Hill, Maidenhead, UK, (1999) 261–278 (Chapter 17).
25. Holstein, D., Moscato, P.: Memetic algorithms using guided local search: A case study. In: Corne, D., Glover, F., Dorigo, M. (Eds.), New ideas in optimization. McGraw-Hill, Maidenhead, UK, (1999) 235–244 (Chapter 15).
26. Merz, P.: Analysis of gene expression profiles: an application of memetic algorithms to the minimum sum-of-squares clustering problem. BioSystems **72** (2003) 99–109.
27. Merz, P., Katayama, K.: Memetic algorithms for the unconstrained binary quadratic programming problem, BioSystems, **78** (2004) 99–118.
28. Berrettaa, R., Rodrigues, L.F.: A memetic algorithm for a multistage capacitated lot-sizing problem, Int. J. Production Economics, **87** (2004) 67–81.
29. Lacomme, P., Prins, C., Ramdane, W.: cherif, Competitive memetic algorithms for arc routing problems, Annals of Operations Research, **131**(2004) 159–185.
30. Buriol, L., Franca, P.M., and Moscato, P.: A New Memetic Algorithm for the asymmetric traveling salesman problem. J. of Heuristics, **10** (2004) 483–506.

31. Tavakkoli-Moghaddam, R., Aryanezhad, M.B., Safaei, N., Azaron, A.: Solving a dynamic cell formation problem using metaheuristics. Applied Mathematics and Computation, Article in press, 2005.
32. Glover, F., Greenberg, H.: New approach heuristic search: a bilateral linkage with artificial intelligence. Eur. J. Oper. Res., **39** (2) (1989) 119-130.
33. Krasnogor, N.: Studies on the theory and design space of memetic algorithms. Ph.D. dissertation, Univ. of the West of England, Bristol, U.K. (2002).
34. Gen, M., Cheng, R.: Genetic algorithms and engineering design. New York: Wiley (1997).

Eco-Grammar Systems as Models for Parallel Evolutionary Algorithms

Adrian Horia Dediu and María Adela Grando*

Research Group on Mathematical Linguistics, Rovira i Virgili University,
Pl. Imperial Tárraco 1, 43005 Tarragona, Spain
{adrianhoria.dediu, mariaadela.grando}@estudiants.urv.es

Abstract. Evolutionary Algorithms (EAs), biological inspired searching techniques, represent a research domain where theoretical proofs are still missing. Due to the lack of theoretical foundations, an extensive experimental work developed many variations of the basic model. Remarkable tendencies such as variable control parameters or parallel populations try to overcome the stagnation observed at the end of evolutions.

We tried to study from theoretical point of view the possibility of modelling parallel EAs using Eco-grammar systems. We expect that our research opens a new perspective over EAs behavior and our framework can bring theoretical results that will lead to new recommendations for EAs architectures as well as for specific details requested by practical problems.

1 Introduction

The *Eco-grammar systems* (EG systems) were introduced in the formal language theory by [5] in 1997 having biological inspiration and universal computational power. There are only several papers regarding the practical applications of EG systems, most of them showing constructive ways to simulate other computational models. As an example, Petr Sosík in [10] constructed an EG system able to simulate an Artificial Neural Network and viceversa. Also in [6] we found the relation between EG systems and Simple Evolutionary Algorithms.

Evolutionary Algorithms (EAs) were introduced in the 60's as biological inspired searching technics. We can find a large number of references for EAs from introductory materials as [3], [2], to monographic books such as [1], [7], [8]. Despite the large number of existing references in this area, EAs represent a research domain where theoretical proofs are still missing. These computational models suggested by the Darwinian paradigm of evolution have been showed to be powerful and to perform well on a broad class of problems. Yet, when the complexity of the applications increases EAs exhibit some limitations, such as the premature convergence. It is interesting to note that the convergence of an

* This work was possible thanks to the research grant "Programa Nacional para la formación del profesorado universitario", from the Ministery of Education, Culture and Sports of Spain.

EA is defined as the process of multiplication in the population of the same individual. If the *convergence* process finds a local optimum we call it a *premature convergence*. Sometimes it is difficult for EAs to escape from such a local optimum. Remarkable tendencies such as variable control parameters [11] or parallel populations try to overcome the stagnation observed at the end of evolutions. In order to overcome these problems efficient parallel models for EAs have been developed. *Parallel Evolutionary Algorithms* (PEAs) [4] have showed a speed up in computation time, they need less objective function evaluations when compared to sequential versions.

In this paper we continue with the work done in [6] but now we focus on EG systems as models for PEAs. Beside the parallel evolution, we also model the process of genetic information interchange between sub-populations. As EGs and EAs are biologically inspired, we found appropriate associations between the computational models' natural metaphors such as individuals, populations, genetic operators, etc. After presenting the formal definition of a PEA we show the way to construct an EG system that can simulate the behavior of the given formal model. The main result we get is that EG systems can be seen as powerful problem solvers with the possibility of modelling EAs. We believe that our research opens a new perspective over EAs behavior and our studying framework will bring new theoretical results about the EAs architectural implementations recommendations. On the other hand EG systems benefit from this approach being able to be used as efficient combinatorial searchers in a huge variety of NP problems, apart from their traditional used as language generators. In this way we introduce in EG systems, that were conceived as formal frameworks for studying *evolution* in eco-systems, the idea of *parallel-evolution* and the role of genetic exchange inter-populations to the general evolution.

2 Eco-Grammar Systems

Before introducing the formal definition of an EG system, we present some notations and basic concepts. An *alphabet* is a finite and nonempty set of symbols. Any sequence of symbols from an alphabet V is called *word* over V. The set of all words over V is denoted by V^* and the empty word is denoted by λ. Further, $V^+ = V \setminus \{\lambda\}$. The number of occurrences of a symbol $a \in V$ in a word $w \in V^*$ is denoted as $(w)_{\#a}$ and the length of w is denoted as $\mid w \mid$. The cardinality of a set S is denoted as $card(S)$.

A Chomsky grammar is a quadruple $G = (N, T, S, P)$, where N is the symbol alphabet, T is the terminal alphabet, $S \in N$ is the axiom, and P is the (finite) set of rewriting rules. The rules are presented in the form $u \rightarrow v$ and used in derivations as follows: we write $x \Longrightarrow y$ if $x = x_1ux_2$, $y = x_1vx_2$ and $u \rightarrow v$ is a rule in P (one occurrence of u in x is replaced by v and the obtained string is y). Denoting by $\Longrightarrow^*$ the reflexive and transitive closure of $\Longrightarrow$, the language generated by G is defined by:

$$L(G) = \{x \in T^* \mid S \Longrightarrow^* x\}.$$

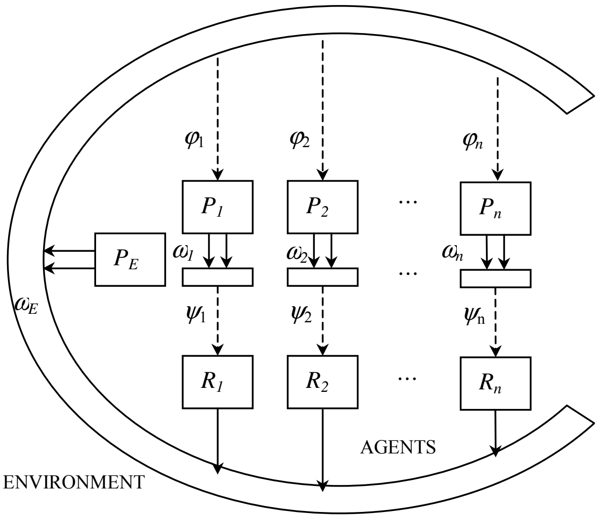


Fig. 1. Description of an eco-grammar system

The families of languages generated by rules of the form $u \to v$ where $u \in N$ and $v \in (N \cup T)^*$ are called context-free grammars and denoted by CF.

Similarly a $0L$ system (an interactionless Lindenmayer system) is a triple $G = (V, \omega, P)$ as above, with context-free rules in P, and with complete P (for each $a \in V$ there is a rule $a \to x \in P$). The derivation is defined in a parallel way: $x \Longrightarrow y$ if $x = a_1 a_2 ... a_n$, $y = z_1 z_2 ... z_n$, for $a_i \in V$ and $a_i \to z_i \in P$, $1 \leq i \leq n$.

For all unexplained notions the reader is referred to [9].

We can find an introductory article about eco-grammar systems (EGs) in [5]. For an easier understanding of the formal definitions we present an intuitive image of the system in figure 1.

We observe that double arrows stand for parallel rewriting $0L-$ rules while simple arrows are used for $CF-$rules.

Definition 1. *(eco-grammar system) An eco-grammar system of degree n is a $(n+1)$-tuple $\Sigma = (E, A_1, \ldots, A_n)$, where $E = (V_E, P_E)$ is the environment that uses V_E as a finite alphabet, and P_E as a finite set of $0L$ rewriting rules over V_E; A_i, $1 \leq i \leq n$, are agents defined by $A_i = (V_i, P_i, R_i, \varphi_i, \psi_i)$ where V_i is a finite alphabet, P_i a finite set of $0L$ rewriting rules over V_i, R_i is a finite set of simple $CF-$rewriting rules over the environment, $R_i \in V_E^+ \times V_E^*$, φ_i, ψ_i are computable functions that select production sets according to the environment state, respectively the agent state: $\varphi_i : V_E^* \to 2^{P_i}$, $\psi_i : V_i^+ \to 2^{R_i}$.*

Until now, the definition provides only the description of the EG system's components. In order to describe the dynamic evolution of EGs, we give the definitions of configuration, derivation and language generated.

Definition 2. *(configuration) A configuration of an eco-grammar system is a tuple $\sigma = (\omega_E, \omega_1, \ldots, \omega_n)$, $\omega_i \in V_i^*, i \in E \cup \{1, \ldots, n\}$ ω_E being a string that represents the environment state and $\omega_1, \ldots, \omega_n$ are strings that represent the agents' state.*

According to the configuration evolution we define the direct derivation of a configuration in an eco-grammar system.

Definition 3. *(direct derivation) Considering an eco-grammar system Σ and two configurations denoted by $\sigma = (\omega_E, \omega_1, \ldots, \omega_n)$ and $\sigma' = (\omega'_E, \omega'_1, \ldots, \omega'_n)$, we say that σ directly derives σ' written as $\sigma \Longrightarrow_\Sigma \sigma'$ iff*

- *$\omega_i \Longrightarrow \omega'_i$ according to the selected set of rules for the i-th agent by the φ_i mapping,*
- *$\omega_E = z_1x_1z_2x_2, \ldots, z_mx_mz_{m+1}$ and $\omega'_E = z'_1y_1z'_2y_2, \ldots, z'_my_mz'_{m+1}$, such that $z_1x_1z_2x_2, \ldots, z_mx_mz_{m+1} \Longrightarrow z_1y_1z_2y_2, \ldots, z_my_mz_{m+1}$ as the result of parallel applying the rewriting rules selected by the ψ_i mappings for all agents and $z_1z_2, \ldots, z_{m+1} \Longrightarrow z'_1z'_2, \ldots, z'_{m+1}$ according to the environment's rules P_E.*

The transitive and reflexive closure of $\Longrightarrow_\Sigma$ is denoted by $\Longrightarrow_\Sigma^+$, $\Longrightarrow_\Sigma^*$ respectively.

The whole "life" of the system is governed by a universal clock, dividing the time in unit intervals: in every time unit the agents act on the environment then the evolution rules rewrite, in parallel, all the remained symbols in the strings describing the environment and the agents. Thus, the action has priority over evolution.

Definition 4. *(the generated language) The language generated by the environment of an eco-grammar system Σ, starting from the initial configuration σ_0 is defined as*

$$L_E(\Sigma, \sigma_0) = \left\{ \begin{array}{c} \omega_E \in V_E^* \mid \sigma_j = (\omega_E, \omega_1, ..., \omega_n), \\ \sigma_0 \Longrightarrow_\Sigma \sigma_1 \Longrightarrow_\Sigma .... \Longrightarrow_\Sigma \sigma_j, j \geq 0 \end{array} \right\}$$

3 Parallel Evolutionary Algorithms

Formally we define a PEA as a $(m + 4)$-tuple:

$$[I, \Phi, StopCondition, \Xi, (\mu_1, \lambda_1, \Omega_1, s_1), ..., (\mu_m, \lambda_m, \Omega_m, s_m)] \tag{1}$$

with m the number of populations that evolve in parallel, $m \geq 0$ and $m \in \mathbf{N}$, where:

- I represents the set of the searching space instances usually called individuals. Sometimes associated with individuals we can keep useful information for genetic operators like their fitness value. Here we use the notation

$\langle \overrightarrow{i}_{q,r}(t), \Phi(\overrightarrow{i}_{q,r}(t))\rangle$, where $\overrightarrow{i}_{q,r}(t)$ denotes the vectorial representation of the chromosome of q-th individual in the r-th population in the t-th generation and $\Phi(\overrightarrow{i}_{q,r}(t))$ corresponds to the fitness value associated to that individual. So, we consider

$$I = \left\{ \begin{array}{l} \langle \overrightarrow{i}_{q,r}(t), \Phi(\overrightarrow{i}_{q,r}(t))\rangle \mid \overrightarrow{i}_{q,r}(t) \in Vector \wedge \Phi(\overrightarrow{i}_{q,r}(t)) \in F \wedge \\ \wedge q \in \mathbf{N} \wedge 1 \leq q \leq \mu_r + \lambda_r \wedge r \in \mathbf{N} \wedge 1 \leq r \leq m \wedge t \in \mathbf{N} \wedge t \geq 0 \end{array} \right\}.$$

- $\Phi : I \rightarrow F$ is the fitness function that determines the adaptation of individuals to the environment, where F is a finite ordered set corresponding to the fitness function values assigned to individuals.
- We consider $StopCondition : F^m \times \mathbf{N} \times F \times \mathbf{N} \rightarrow Boolean$, where $StopCondition(f_1, ..., f_m, gen, impfitness, maxGen)$ is satisfied when the number of generation gen exceeds $maxGen$ generations or when the fitness value of the best individuals from all populations $(f_1, ..., f_m)$ is greater or equal than $impfitness$.
- $\Xi : (I^{\mu_1}) \times ... \times (I^{\mu_m}) \rightarrow (I^{\mu_1}) \times ... \times (I^{\mu_m})$ is an operator that exchanges individuals among the m populations. An structure, usually a graph, is associated to Ξ: each node N_x, $1 \leq x \leq m$, represents a population P_x and each directed vertex V_{xy} connecting nodes N_x and N_y indicates that a certain number of individuals from population P_x are selected with some criteria (best fitness value, random selection, etcetera), removed from P_x and moved to P_y.
- $(\mu_r, \lambda_r, \Omega_r, s_r)$ with $r \in \mathbf{N}$ and $1 \leq r \leq m$ are parameters required for the computation of the evolutive process in the r-th population where:
  - $\mu_r \in \mathbf{N}$, $\mu_r \geq 1$ denotes the number of parents of the r-th population.
  - $\lambda_r \in \mathbf{N}$, $\lambda_r \geq 1$ is the number of offsprings inside the r-th population.
  - $\Omega_r : I^{\mu_r} \rightarrow I^{(\mu_r+\lambda_r)}$ is a set of genetic operators which applied to the individuals of the r-th population, called parents, produce new λ_r individuals called offsprings.
  - $s_r : I^{\mu_r} \times I^{\lambda_r} \rightarrow I^{\mu_r}$ is the selection operator that chooses individuals from parents and offsprings of the r-th population to remain in next generation. There are variants of EAs where s_r selects only from the offsprings of the r-th population $(s_r : I^{\lambda_r} \rightarrow I^{\mu_r})$.

In figure 2 we present a general description of a PEA, where we use the notation $\|procP$ for "parallel execution of procedure $procP$". The variables gen and $isolationtime$ are numerical ones and they indicate respectively the current number of generation and the number of generations that have to be wait until an interchange process can take place. $P_i(gen)$ represents the set of μ_i individuals that are potential parents in i-th population and gen-th generation and $P'_i(gen)$ is the set of λ_i offsprings we get from the application of genetic operators to individuals from $P_i(gen)$. The step 2 of the algorithm, $initialization(n, \mu_i, L)$, is the call to a function that returns the set of μ_i individuals of i-th population initialized with random values. The number of genes of the chromosomes of those individuals is $n \in \mathbf{N}$ and their fitness values are not initialized. Depending on the coding chosen for the problem an ordered list L for genes possible values is given, each gene searching space being a set C_j, $1 \leq j \leq n$. Without loss of

```
1   $gen := 0$;
2   || Initialization process
        $(P_1(0) := initialization(n, \mu_1, L), ...,$
        $..., P_m(0) := initialization(n, \mu_m, L),$
3   ||Evaluate fitness values of all the individuals of $P_1(0), ..., P_m(0)$;
4   do while not $(StopCondition(best(P_1(gen)), ..., best(P_m(gen)),$
                 $gen, impfitness, maxGen))$
5      || Apply genetic operators
           $\Omega_1(P_1(gen)) \to P_1'(gen), ...$
           $..., \Omega_m(P_m(gen)) \to P_m'(gen)$
6      || Evaluate fitness values of all the individuals of $P_1'(gen), ..., P_m'(gen)$;
7      || Select the next generation
           $(P_1(gen+1) := s_1(P_1(gen), P_1'(gen)), ...,$
           $..., P_m(gen+1) := s_m(P_m(gen), P_m'(gen))$;
8      $gen := gen + 1$;
9.1    if $mod(gen, isolationtime) = 0$ then
9.2        Apply exchange of individuals between populations
             $\Xi(P_1(gen), ...., P_m(gen)) \to \{P_1(gen+1), ..., P_m(gen+1)\}$;
9.3        $gen := gen + 1$;
       endif;
   end do;
```

Fig. 2. The structure of a PEA

generality we consider each C_j a finite and discrete set of values of a given type, such that all values of genes in the position j of a chromosome of an individual are of type C_j. The function $best : I^{\mu_i} \to I$ returns the individual with best fitness value from I^{μ_i}. The function $mod : \mathbf{Z} \times \mathbf{Z} \to \mathbf{Z}$ returns the rest of the division between the first parameter called dividend and the second parameter called divisor.

4 Simulating a Parallel Evolutionary Algorithm with an Eco-Grammar System

In the table (Table 1) we show how we associate in this simulation concepts from PEAs and EG systems.

Theorem 1 ($PEA \to EG$)**.** *Given an arbitrary PEA of m populations defined like in (3) and with a description like in figure 2, we can define an EG system able to simulate the PEA behavior.*

Next we show how we can construct an EG system starting from a given PEA. The constructed EG system has $(\mu_1 + \lambda_1) + (\mu_2 + \lambda_2) + ... + (\mu_m + \lambda_m)$ agents, where each agent $A_{x,y}$ represents the x-th individual, $1 \leq x \leq \mu_y + \lambda_y$, of the population P_y, $1 \leq y \leq m$. In EG systems the number of agents (individuals) is fixed, so we have to define for each population P_y a number of $(\mu_y + \lambda_y)$ agents, which is the maximal number of individuals for that population.

We consider that the evaluation of any function and the performance of any communication process takes only one time unit. In the implementations of

Table 1. Associations between entities in Parallel EAs and EG systems

Concept	**Entity of PEA**	**Entity of EG system**
Individual	Vector $\overrightarrow{i}_{x,y}(gen)$	String $\omega_{x,y}$, state of agent $A_{x,y}$
Evolutive Process	Algorithm of figure 2	Production set P_E and a symbol in ω_E
Initial Population	Initialization process	Initialization process embedded in $\varphi_{x,y}$
Fitness function	Function ϕ	Function ϕ, embedded in $\psi_{x,y}$
Termination	Function $StopCondition$	Function $StopCondition$ embedded in $\varphi_{x,y}$
Number of generation	Variable gen	Number of symbols $Generation$ in ω_E
Genetic operators	Set Ω_y of operators	Set Ω_y of operators embedded in $\varphi_{x,y}$
Selection operator	Function s_y	Function s_y embedded in $\varphi_{x,y}$
Interchange operator	Function Ξ	Function Ξ embedded in $\psi_{x,y}$

PEAs not all the functions consume the same time, in particular fitness function evaluation or the interchange process can consume a large amount of time. Yet our modelling hypothesis still represent a general enough framework for EAs while we can consider that a synchronization takes place after each generation.

We define the environment's sate of the EG system as follows:

$$\begin{aligned}\omega_E = ControlSymbol \cdot Generation^{gen} \cdot c_{1,1} \cdot f_{1,1} \cdot Status_{1,1}... \\ ...c_{(\mu_1+\lambda_1),1} \cdot f_{(\mu_1+\lambda_1),1} \cdot Status_{(\mu_1+\lambda_1),1}... \\ ...c_{(\mu_m+\lambda_m),m} \cdot f_{(\mu_m+\lambda_m),m} \cdot Status_{(\mu_m+\lambda_m),m}\end{aligned}$$

where:

- $ControlSymbol \in V_E$ together with the environment productions rules P_E simulate the control sequence described by figure 2. In the formal description of the simulation, the V_E alphabet's symbols like $GeneticProcess$, $Selection$, $NewIteration$, $CheckTermination$, $Termination$, etcetera are used in ω_E to indicate the current step of the algorithm.
- $Generation \in V_E$ represents a special symbol used to code the generation number in the EA. Each time the number of generation has to be increased (steps 8 and 9.3 of the algorithm) a new symbol $Generation$ is introduced in ω_E.
- $c_{x,y} = \langle h_{1_{x,y}}, ..., h_{n_{x,y}} \rangle_{x,y}$ is the string representing the chromosome of the x-th individual in the y-th population. Because the chromosome of all individuals are kept in ω_E, subindexes x and y are needed to address a particular individual.
- $h_{s_{x,y}} \in C_s$ represents the gene value in the s-th position of the chromosome of the x-th individual in the y-th population, it has the type C_s, $1 \leq s \leq n$.

– $f_{x,y}$ is the fitness value $f_{x,y} \in F \cup \{\min\}$ of the x-th individual in the y-th population.
– $Status_{x,y} \in \{O, P, S, R, M, Copy, Stop, Inactive, IncrGen\}$ means respectively that the individual is an offspring, is a potential parent, has been selected for next generation, has been rejected for next generation, is participating in the exchange process, the string representing its chromosome has to be copied in the environment's state, the algorithm's stop condition is satisfied, the x-th individual in the y-th population is in inactive state and the x-th individual in the y-th population is in inactive state and that the EG can increment the number of generation.

In a similar manner we define the agents' sate as follows:

$$\omega_{x,y} = c \cdot f_{x,y} \cdot Status_{x,y}$$

where: $1 \leq x \leq \mu_y + \lambda_y, 1 \leq y \leq m$.

With respect to the mappings $\varphi_{x,y}$ and $\psi_{x,y}$, the first one embeds the process of random initialization of individuals (step 1), the checking of the stop condition (step 4) and the application of functions from Ω_y (step 5), function s_y (step 7) and function Ξ (step 9.2). For mappings $\varphi_{x,y}$ being able to perform step 4, 5, 7 and 9.2 they need to know the fitness values of the individuals during the evolutive process. For the simulation of the step 5 and 9.2 the mappings $\varphi_{x,y}$ also need to have information about the chromosomes of all the individuals of all populations with the status of potential parents. And for step 7 besides the chromosomes of the parents, the offspring's chromosomes are needed. So before performing this operation the corresponding chromosomes are copied in ω_E by the mappings $\psi_{x,y}$. Mappings $\psi_{x,y}$ are also in charge of embedding the fitness function Φ performing the evaluations of steps 3, 6 and 9.2 of the algorithm.

In any computational implementation of PEA following the algorithm shown in Figure 2, populations are suppose to evolve in a parallel and independent way in steps 2, 3, 5, 6 and 7. Each population consumes different times to perform these steps according to their number of individuals, the time consumed by the tasks performed and the number of processors assigned. For example some populations can be in the step 5, while others can be in the step 6 or 7; performing the step 8, all the populations have to synchronize. In this definition of EG system we assign to each individual one agent or processing unit and one time-unit to the execution of any action, therefore all the populations execute the same step of the algorithm in a parallel and synchronized manner. At a given time unit all populations are performing the same evolutive step so it is enough to keep in ω_E only one control symbol to deal with parallelism and synchronization. We exemplify this simulating step 5 of the algorithm with the definition of EG system we give. We show how genetic operators Ω_y from all populations P_y are applied in a simultaneous and parallel way in one derivation step of the system. We start the simulation from a configuration equivalent to the following one:

$$\sigma_s = (GeneticProcess \cdot Generation^{gen} \cdot c_{1,1} \cdot f_{1,1}, Status_{1,1} ...$$
$$... c_{(\mu_m+\lambda_m),m} \cdot f_{(\mu_m+\lambda_m),m}, Status_{(\mu_m+\lambda_m),m},$$

$$\begin{array}{l}
\omega_{1,1},...,\omega_{(\mu_1+\lambda_1),1},...,\omega_{1,m},...,\omega_{(\mu_m+\lambda_m),m}) \text{ where} \\
(\text{for all } i,j : 1 \leq i \leq \mu_j + \lambda_j \wedge 1 \leq j \leq m \wedge c_{i,j} = \langle h_1, h_2, ..., h_n\rangle_{i,j}) \wedge \\
\wedge \left(\begin{array}{l} \text{for all } \omega_{x,y} \in Individuals_{\ 0} : \\ \left(\begin{array}{l} \omega_{x,y} = \langle m_{1_{x,y}}, m_{2_{x,y}}, ..., m_{n_{x,y}}\rangle P \wedge \\ \wedge c_{x,y} = \langle m_{1_{x,y}}, ..., m_{n_{x,y}}\rangle \wedge \\ \wedge f_{x,y}, Status_{x,y} = \Phi(\langle m_{1_{x,y}}, ..., m_{n_{x,y}}\rangle), P \end{array} \right) \vee \\ \vee(\omega_{x,y} = Inactive \wedge f_{x,y}, Status_{x,y} = f_{x,y}, I) \end{array} \right) \wedge \\
\wedge \left(\begin{array}{l} \text{for all } y : 1 \leq y \leq m \wedge (\omega_{1,y}....\omega_{(\mu_y+\lambda_y),y})\#P = \mu_y \wedge \\ \wedge(\omega_{1,y}....\omega_{(\mu_y+\lambda_y),y})\#Inactive = \lambda_y \end{array} \right) \wedge \\
\wedge StopCondition(f_{s_1,1}, ..., f_{s_m,m}, gen, impfitness, maxGen) \equiv false \wedge \\
\wedge \left(\begin{array}{l} \text{for all } r : 1 \leq r \leq m \wedge f_{s_r,r} \text{ is a substring of } \omega_E \wedge \\ \wedge \left(\begin{array}{l} \text{for all } k : 1 \leq k \leq \mu_r + \lambda_r \wedge \\ \wedge f_{s_r,r} \geq f_{k,r} \wedge f_{k,r}, P \text{ is a substring of } \omega_E \end{array} \right) \end{array} \right)
\end{array}$$

The presence of the symbol *GeneticProcess* in the environment's state of σ_s indicates that performance of step 5 has to take place and the number *gen* of occurrences of the symbol *Generation* shows that the number of generation is *gen*. The number of agents in each population P_y whose state contain the symbol P (have the status of parents) is μ_y while the number of agents containing symbol $Inactive$ (are in inactive state) is λ_y. The environment ω_E contains all the information needed for the application of the genetic operators: the strings corresponding to the chromosomes of the potential parents of every population and their fitness values, what we denoted $P_y(gen)$, $1 \leq y \leq m$. And the stopping condition is not satisfied.

From a configuration equivalent to σ_s we get a new configuration in one evolution step of the EG system. Rules $GeneticProcess \to OffSpringEvaluation \in P_E$ and the following definition of mappings $\varphi_{x,y}$, for $1 \leq x \leq \mu_y + \lambda_y$ and $1 \leq y \leq m$ are applied:

$$\begin{array}{l}
\varphi_{x,y}(GeneticProcess \cdot Generation^{gen} \cdot \alpha) = \\
\left\{ \begin{array}{l} Inactive \to \langle y_1, y_2, ..., y_n\rangle \text{ such that } \langle y_1, y_2, ..., y_n\rangle \\ \text{is the chromosome assigned to individual x} \\ \text{resulting from the application of genetic operators from } \Omega_y \\ \text{to those individuals from population } P_y \text{ in } \omega_E \\ \text{with the status of parents} \end{array} \right\}
\end{array}$$

Step 5 of the algorithm is simulated: genetic operators from Ω_y are applied to each population P_y. λ_y individuals of each population that were in an inactive state ($\omega_{x,y} = Inactive$) rewrite it for a string corresponding to a chromosome resulting from the application of the genetic operators followed by the symbol O that indicates status of offspring ($\omega'_{x,y} = \langle m_{1_{x,y}}, m_{2_{x,y}}, ..., m_{n_{x,y}}\rangle O$).

The environmental rule replaces the symbol $Genetic\mathrm{Pr}ocess$ from ω_E for $OffspringEvaluation$ indicating that in next derivation step of the EG system step 6 of the algorithm has to be simulated: fitness values of the offsprings have to be actualized in ω_E. So from σ_s we get a configuration equivalent to this one:

$$\sigma_t = (OffspringEvaluation \cdot Generation^{gen} \cdot c_{1,1} \cdot f_{1,1}, Status_{1,1} \ldots$$
$$c_{(\mu_m+\lambda_m),m} \cdot f_{(\mu_m+\lambda_m),m}, Status_{(\mu_m+\lambda_m),m},$$
$$\omega'_{1,1},\ldots,\omega'_{(\mu_1+\lambda_1),1},\ldots,\omega'_{1,m},\ldots,\omega'_{(\mu_m+\lambda_m),m}) \text{ where}$$
$$\wedge \left(\begin{array}{l} \text{for all } \omega'_{x,y} : \\ \left(\begin{array}{l} \omega_{x,y} = \langle m_{1_{x,y}}, \ldots, m_{n_{x,y}} \rangle P \rightarrow \\ \rightarrow \left(\begin{array}{l} \omega'_{x,y} = \omega_{x,y} \wedge c_{x,y} = \langle m_{1_{x,y}}, \ldots, m_{n_{x,y}} \rangle \wedge \\ \wedge f_{x,y}, Status_{x,y} = \Phi(\langle m_{1_{x,y}}, \ldots, m_{n_{x,y}} \rangle)_{x,y}, P \end{array} \right) \end{array} \right) \wedge \\ \wedge(\omega_{x,y} = Inactive \longrightarrow \omega'_{x,y} = \langle m_{1_{x,y}}, \ldots, m_{n_{x,y}} \rangle O) \wedge \\ \wedge \left(\begin{array}{l} \text{for all } y : 1 \leq y \leq m \wedge (\omega'_{1,y} \ldots \omega'_{(\mu_y+\lambda_y),y}) \# P = \mu_y \wedge \\ \wedge (\omega'_{1,y} \ldots \omega'_{(\mu_y+\lambda_y),y}) \# O = \lambda_y) \end{array} \right) \end{array} \right)$$

It is important to mention that we always keep in the environment state the evolution of the individuals fitness values of every populations. The main reason for this is that the application of functions embedded in mappings $\varphi_{x,y}$ require this information. Also when the stop condition is satisfied the evolution of the system is stopped introducing a symbol $Termination$ in the environment state, from which no evolution can take place. In this case ω_E has to be examine to find the individual with best fitness value, which will be considered the solution of the problem.

5 Conclusions and Future Work

In this paper we present a method to simulate the behavior of a given Parallel Evolutionary Algorithm using an EG system. We think that from now on, EG systems may be used not just as language generators that give exact answers, but also as efficient probabilistic solvers that can be used to obtain good solutions of very complex problems in an efficient way. As the the spectrum of applications that use EAs is quite large, we mention only several of them like game playing, face recognition, financial time-series prediction, etc., then all these applications might be better studied using EG systems.

For future work we will focus on studying theoretical aspects regarding the alphabet used in the coding of individuals in EAs, the control parameters and appropriate operators for EAs, all of them analyzed from the EG system's point of view.

References

1. Th. Bäck, *Evolutionary Algorithms in Theory and Practice - Evolution Strategies, Evolutionary Programming, Genetic Algorithms*, Oxford University Press (1996).
2. D. Beasley, D.R. Bull, R.R. Martin, *An Overview of Genetic Algorithms, Part 1*, Fundamentals, University Computing (1993), 15(2) pp. 58-69.
3. D. Beasley, D.R. Bull, R.R. Martin, *An Overview of Genetic Algorithms, Part 2*, Research topics, University Computing (1993), 15(4) pp. 170-181.
4. E. Cantu-Paz, A survey of Parallel Geneti Algorithms, IlliGAL Report No. 97003 (1997).

5. E. Csuhaj-Varjú., J. Kelemen, A. Kelemenová, Gh. Păun, *Eco-grammar systems: A grammatical framework for studying lifelike interactions*, Artificial Life 3 (1997) pp. 1-28.
6. A. H. Dediu, M. A. Grando, *Simulating Evolutionary Algorithms with Eco-grammar Systems* IWINAC 2005, LNCS 3562 ,J. Mira and J.R. Alvarez (eds.) Springer-Verlag Berlin Heidelberg (2005) pp. 112-121.
7. D.E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley (1989).
8. Z. Michalewicz, *Genetic Algorithms + data Structures = Evolution Programs*, Springer - Verlag, New York (1996).
9. G. Rozenberg, A. Salomaa (eds.), *Handbook of Formal Languages*, Springer-Verlag, Berlin (1997).
10. Petr Sosík, *On eco-grammar systems and artificial neural networks*, Computers and Artificial Intelligence 15 (1996) pp. 247-264
11. M Srinivas, L. M. Patnoik, *Adaptive Probabilities of Crossover and Mutation in Genetic Algorithms*, IEEE Transactions on Systems, Man and Cybernetics, Vol. 24, No. 4 (1994) pp. 656-668.

Author Index