BIOINSPIRED OPTIMIZATION METHODS AND THEIR APPLICATIONS

BIOINSPIRED OPTIMIZATION METHODS AND THEIR APPLICATIONS

Proceedings of the International Conference on Bioinspired Optimization Methods and their Applications, BIOMA 2004

11–12 October 2004, Ljubljana, Slovenia

Edited by
BOGDAN FILIPIČ
JURIJ ŠILC

Jožef Stefan Institute, Ljubljana

Jožef Stefan Institute
Ljubljana, Slovenia

Editors: Bogdan Filipič, Jurij Šilc

Cover design by Studio Design Demšar, Škofja Loka
Logo design by Gregor Papa
Published and printed by Jožef Stefan Institute, Ljubljana, Slovenia

Typesetting in kapproc-based LaTeXstyle with kind permission from Kluwer Academic Publishers

CIP - Kataložni zapis o publikaciji
Narodna in univerzitetna knjižnica, Ljubljana

004.8:519.8(063)(082)

INTERNATIONAL Conference on Bioinspired Optimization Methods and their Applications (2004 ; Ljubljana)
Bioinspired optimization methods and their applications : proceedings of the International Conference on Bioinspired Optimization Methods and their Applications - BIOMA 2004, 11-12 October 2004, Ljubljana, Slovenia / edited by Bogdan Filipič, Jurij Šilc. - Ljubljana : Jožef Stefan Institute, 2004

ISBN 961-6303-61-9
1. Gl. stv. nasl. 2. Filipič, Bogdan
215508736

BIOMA 2004 is part of the Information Society multiconference (IS 2004).
The IS 2004 multiconference is partially financed by the Ministry of Education, Science and Sport, and Jožef Stefan Institute.

ISSN-1581-9973

Contents

Preface

Solving complex computational problems requires powerful hardware and clever algorithms. Thanks to unimagined technological progress, computing power has been increasing exponentially over generations of computers, and all recent trends indicate the growth rate will continue in the future. In parallel to this process, computer scientists are developing novel algorithms to solve the tasks whose complexity is often beyond human comprehension. Engineering design, dynamic systems control, econometric modeling and bioinformatics are only few fields where traditional well-understood algorithms are insufficient. In search for better computational techniques, biological systems have proved to be a rich source of inspiration. This comes at no surprise: efficiency and perfection of natural phenomena, such as the evolution of species, collective behavior of organisms, their adaptivity and information processing capabilities, are still far ahead of those exhibited by man-made systems.

Mimicking the principles of biological systems and employing them as problem-solving heuristics is a particularly well-established practice in the design of optimization algorithms. In some cases, computer models were originally implemented to formally study the phenomena occurring in nature, and later adopted as general-purpose optimization techniques by computer specialists. Genetic algorithms are the most widely known technique of such origin. On the other hand, some approaches, such as ant colony optimization, arose from straightforward attempts of applying concepts from nature in solving optimization problems of a certain class. After years of theoretical and empirical research, bioinspired optimization has reach the stage where the algorithms, either in the form of commercial software packages or tailored applications, are being regularly used in solving complex optimization problems in science, engineering and business. Interestingly, many techniques produce excellent results in practice despite the lack of the theory explaining and predicting their behavior.

This volume contains some of the latest theoretical and practical contributions to the field of bioinspired optimization. The papers were presented at the International Conference on Bioinspired Optimization Methods and their Ap-

plications (BIOMA 2004), held in Ljubljana, Slovenia, on 11 and 12 October 2004.

The aim of organizing (the first) BIOMA was to bring together a group of theoreticians and practitioners to present their recent achievements in a single stream of talks, and exchange the ideas in informal discussions. We felt that, in addition to numerous well-established large conference series, an event of this format could also contribute to the progress of the field and its promotion. The participating authors were mainly attracted through our research links. After the review process, 14 papers were accepted for publication, contributed by 26 (co)authors coming from 7 countries. Thomas Bäck, a distinguished expert in theory and applications of evolutionary algorithms, presented the work of himself and coauthors from NuTech Solutions on evolution strategies as a bioinspired optimization tool for engineering. The remaining contributions were divided into two categories, one dealing with theoretic and algorithmic issues, and the other presenting practical applications. The latter addressed a variety of real-world problems, such as production process optimization and scheduling, engineering design and testing, flight maneuvers control, and balancing of healthy meals. In addition to oral presentations, the participants demonstrated several software packages and applications in a demo session.

BIOMA 2004 was sponsored by the Ministry of Science, Education and Sport of the Republic of Slovenia. It was organized as part of the 7th International Multiconference Information Society (IS 2004) taking place at the Jožef Stefan Institute, Ljubjana, from 11 to 15 October 2004. BIOMA was held at the newly founded Jožef Stefan International Postgraduate School that opens for the first generation of students in the academic year 2004/2005 and will also include bioinspired optimization in its curriculum.

We are grateful to the conference sponsors, members of the program and organizing committees, the invited speaker, and regular paper and software presenters for taking part in making the conference successful. Shaping the conference and putting this volume together was enjoyable, and we hope you will find the event stimulating and the book informative.

Ljubljana, 24 September 2004

BOGDAN FILIPIČ AND JURIJ ŠILC

Program Committee

BOGDAN FILIPIČ, *Co-Chair*, Jožef Stefan Institute, Ljubljana, Slovenia
JURIJ ŠILC, *Co-Chair*, Jožef Stefan Institute, Ljubljana, Slovenia
THOMAS BÄCK, NuTech Solutions, Dortmund, Germany
CARLOS A. COELLO COELLO, CINVESTAV-IPN, Mexico
MARCO DORIGO, Université Libre de Bruxelles, Belgium
ROLF DRECHSLER, University of Bremen, Germany
DONALD E. GRIERSON, University of Waterloo, Canada
BARBARA KOROUŠIĆ SELJAK, Jožef Stefan Institute, Ljubljana, Slovenia
THIEMO KRINK, University of Aarhus, Denmark
MARJAN MERNIK, University of Maribor, Slovenia
ZBIGNIEW MICHALEWICZ, University of North Carolina at Charlotte, USA
GREGOR PAPA, Jožef Stefan Institute, Ljubljana, Slovenia
LALIT M. PATNAIK, Indian Institute of Science, Bangalore, India
BORUT ROBIČ, University of Ljubljana, Slovenia
JIM TØRRESEN, University of Oslo, Norway
ROMAN WYRZYKOWSKI, Częstochowa University of Technology, Poland
ALBERT Y. ZOMAYA, The University of Sydney, Australia

Organizing Committee

GREGOR PAPA, *Chair*, Jožef Stefan Institute, Ljubljana, Slovenia
PETER KOROŠEC, Jožef Stefan Institute, Ljubljana, Slovenia
TEA ROBIČ, Jožef Stefan Institute, Ljubljana, Slovenia

Contributing Authors

Inès Alaya recieved the diploma and master degree in computer science applied to management from the High Institute of Management, Institut Supérieur de Gestion de Tunis, Tunisia, in 2004. Her current research interests include ant algorithms, combinatorial optimization problems.

Thomas Bäck is an expert on theory and applications of evolutionary algorithms. He received his Ph.D. in Computer Science from Dortmund University, Germany, in 1994, and then worked for the Informatik Centrum Dortmund (ICD) as department leader of the Center for Applied Systems Analysis, and later for Divis Digital Solutions GmbH as President and Chief Executive Officer. Since early 2000, he is Managing Director of NuTech Solutions GmbH and CTO of NuTech Solutions, Inc. His role at NuTech Solutions includes the scientific supervision of projects with Fortune 1000 customers such as BMW Group, Beiersdorf, Corning, Inc., Ford of Europe, Fujitec, Hawker Batteries, Honda, Siegwerk Druckfarben, Siemens, Unilever, and others. Since 1996, Thomas Bäck has been Associate Professor of Computer Science at the Leiden Institute of Advanced Computer Science (LIACS), Leiden University, The Netherlands, and Full Professor for Natural Computing since 2002. Thomas Bäck has more than 120 publications on natural computing technologies, as well as a book on evolutionary algorithms, entitled Evolutionary Algorithms: Theory and Practice. He is editorial board member and associate editor of a number of journals on evolutionary and natural computation, and has served as program chair for all major conferences in evolutionary computation. His expertise lies in adaptive technologies for optimization and data-driven modeling, predictive analytics, and bioinformatics. He received the best dissertation award from the Gesellschaft für Informatik (GI) in 1995 and is an elected fellow of the International Society for Genetic and Evolutionary Computation for his contributions to the field.

Barrett R. Bryant is a professor and the associate chair in the Department of Computer and Information Sciences at the University of Alabama at Birming-

ham (UAB). He joined UAB in 1983 after completing his Ph.D. in computer science at Northwestern University. He completed his M.S. at Northwestern in 1980 and his B.S. at the University of Arkansas at Little Rock, both in computer science. He has held various visiting positions at universities and research laboratories since joining UAB. Barrett's primary research focus is in theory and implementation of programming languages, especially formal specification languages, and object-oriented and component-based software technology. He has authored or co-authored over 80 technical papers in these areas. He has also chaired conferences, program committees, and special sessions of many international conferences in these areas. Barrett is a member of ACM, the IEEE Computer Society, and the Alabama Academy of Science. He is a Distinguished Lecturer for the ACM and is currently the Chair of the ACM Special Interest Group on Applied Computing (SIGAPP).

Salvatore D'Angelo is professor at the Aeronautical and Space Engineering Department, Politecnico di Torino. He completed his M.S. degree at Politecnico di Torino in 1969. His current research interests are in the field of flight mechanics, aerodynamics, evolutionary computation and artificial neural networks. He has published several technical papers in journals and conference proceedings.

Marco Dutto is a graduated student at the Aeronautical and Space Engineering Department, Politecnico di Torino. He completed his M.S. degree at Politecnico di Torino in 2004 with a thesis on controller design by evolutionary algorithms.

Michael Emmerich works as a computer scientist at the University of Dortmund. He received his diploma in Applied Informatics (application field: chemical engineering) from the University of Dortmund in 1998. As a researcher he worked at the Center of Applied Systems Analysis / Informatik Centrum Dortmund, where, until 2002, he took part in projects on optimization in the process industries. After short term projects at the RWTH Aachen, in 2003 he joined the chair of systems analysis at the University of Dortmund, where he now works in the Collaboratory Research Center on Computational Intelligence. His current research interests are the design of problem-specific optimization algorithms, with a focus on engineering applications, and the theory of stochastic process models.

Bogdan Filipič received his Ph.D. in Computer Science from the University of Ljubljana, Slovenia, in 1993. He is currently a research associate at the Department of Intelligent Systems of the Jožef Stefan Institute, Ljubljana, and an

assistant professor of Computer and Information Science at the University of Ljubljana. His research interests include evolutionary computation, intelligent data analysis and knowledge-based systems. He also participates in industrial projects dealing with optimization of production processes, such as continuous casting of steel and automobile production. He has published in a number of scientific journals, including IEEE Transactions on Systems, Man and Cybernetics, Engineering Applications of Artificial Intelligence, Computers in Industry, Review of Scientific Instruments, International Journal of Human-Computer Studies, and Applied Soft Computing. Bogdan Filipič serves as a member of editorial boards of several journals and a programme committee member for the Genetic and Evolutionary Computation Conference (GECCO) and Congress on Evolutionary Computation (CEC). He is a member of the IEEE Computer Society, IEEE Computational Intelligence Society, and the Executive Board of the Slovenian Artificial Intelligence Society (SLAIS).

Tomasz Garbolino is an assistant professor in the Faculty of Automatic Control, Electronics and Computer Science at the Silesian University of Technology at Gliwice, Poland. He received his M.S. and Ph.D. degrees (with honors) in Electronics from the Technical University of Gliwice in 1993 and 2002, respectively. His research interests encompass built-in self-test structures for digital circuits and SOCs, with particular focus on test pattern generators and test pattern decompression techniques, as well as design for testability issues. He is a co-author of several papers that have been published in proceedings of international conferences and journals.

Khaled Ghédira is a professor at the Computer Science Department, Institut Supérieur de Gestion de Tunis, Tunisia. He received the diploma of engineer in Hydrolytic at ENSEEIHT (Ecole Nationale Supérieure d'Electrotechnique, d'Electronique, d'Informatique et d'Hydraulique de Toulouse), France, in 1983, and the diploma of engineer in Computer Science and Applied Mathematics at ENSIMAG (Ecole Nationale Supérieure d'Informatique et de Mathématiques Appliquées de Grenoble), France, in 1986. He recieved PhD degree at ENSAE (Ecole Nationale Supérieure de l'Aéronautique et de l'Espace), Toulouse, France, in 1993, and HDR at ENSI (Ecole Nationale des Sciences de l'Informatique), Tunisia, in 2001. He is the president of the Tunisian Association of Artificial Intelligence ATIA, the director of the unity of research SOIE and the director of ENSI. His current research interests include multi-agent systems, constraint satisfaction problems, mono and multi-criteria optimization, combinatorial optimization problems, logistic and transport.

Andrzej Hławiczka is a professor in the Faculty of Automatic Control, Electronics and Computer Science at the Silesian University of Technology at Gliwice, Poland. He received the M.S. degree in Electrical Engineering from the Technical University of Gliwice in 1965. His Ph.D. degree, in Computer Engineering, and D.Sc., in Electronics, were received from the Silesian University of Technology at Gliwice, in 1973 and 1998, respectively. From 1965 to 1979 he was mainly with the Institute of Mathematical Machines and Institute of Control Systems, Katowice, Poland, where he published research related to hazards detection, logic simulation, and fault test generation. Professor Hławiczka's research interests include digital circuits and SoCs built-in self-testing, fault-tolerant computing, and design for testability. He takes part in Program Committees projects of several European conferences in his field. He is the editor, co-editor, and co-author of several scientific books published in Polish, Russian and English. He has published over 120 scientific papers in international journals and conferences, and has supervised several Ph.D. dissertations. He is a Senior Member of the IEEE and a chairman of the Computer Society Chapter, IEEE Poland Section.

Jürgen Jakumeit is the head of the Process Simulation Development Group at ACCESS e.V. and Privatdozent at the University of Cologne. He received a diploma (1990) and a Ph.D. degree (1994) in Physics from the University of Cologne. Since October 2001 he has the vina legendi for experimental physics from the University of Cologne. Until 1998 he studied and worked at the II. Physical Institute of the University of Cologne in the field of semiconductor physics with a strong emphasis on computer simulation. In 1998 he moved to the German National Research Center for Information Technology (GMD) in Sankt Augustin. The simulation and optimization of industrial production processes became now the focus of his work. Since May 2002 he leads the development of the casting simulation package CASTS at Access e.V. at the RWTH Aachen. Here, his main interests are the simulation of complex macroscopic phenomena during casting processes. During his time at the GMD and at Access a main field of active research was the numerical optimization of industrial production processes.

Peter Korošec is a research assistant at the Jožef Stefan Institute in Ljubljana, Slovenia. He received his M.S. degree in Computer Science from the University of Ljubljana, Slovenia, in 2004. His research interests include metaheuristics algorithms, parallel processing, and combinatorial optimization.

Barbara Koroušić Seljak was born in Ljubljana, Slovenia. She received her M.S. and Ph.D. degrees in Computer Science and Informatics from the Univer-

sity of Ljubljana in 1992 and 1997, respectively. Currently, she is a research assistant in the Computer Systems Department at the Jožef Stefan Institute, Ljubljana. Her research focuses on design of real-time and embedded systems. She is also interested in modern heuristic methods applicable to practical optimization problems.

Peter Krause received his diploma degree in Electrical Engineering in 1996 from the University of Dortmund, Germany. From 1996 to 2001 he worked at the Chair of Electrical Control Engineering at the Electrical Engineering Department at the University of Dortmund. His research was part of the Collaborative Research Centre 531 "Computational Intelligence" at the University of Dortmund. His research interests were centred on data mining, fuzzy logic and generation and optimization of fuzzy systems. In 2001 he received his PhD from the Electrical Engineering Department at the University of Dortmund. In addition to being the author of more than 20 research publications, he also possesses expert knowledge of object oriented development, GUI development, SQL and C/C++. At the beginning of 2002 he joined NuTech Solutions Inc. as Senior Scientist and Project Manager.

Shih-Hsi Liu is currently a Ph.D. student at the Department of Computer and Information Sciences in the University of Alabama at Birmingham under Dr. Barrett R. Bryant. He received his B.S. at National Chiao-Tung University, Taiwan, in 2000, and M.S. degree University of Houston, USA, in 2002. His current research interests are in the fields of distributed real-time and embedded systems, programming languages, and evolutionary computation.

Marjan Mernik received his M.S. and Ph.D. degrees in Computer Science from the University of Maribor in 1994 and 1998 respectively. He is currently an associate professor at the University of Maribor, Faculty of Electrical Engineering and Computer Science. He was a visiting professor in the Department of Computer and Information Sciences at the University of Alabama at Birmingham in 2004. His research interests include principles, paradigms, design and implementation of programming languages, compilers, formal methods for programming language description, and evolutionary computation. He is a member of the IEEE, ACM and EAPLS.

Edmondo Minisci is a research fellow at the Aeronautical and Space Engineering Department, Politecnico di Torino. He completed his M.S. degree at Politecnico di Torino in 1998 and he received Ph.D. degree from Politecnico di Torino in 2004. His current research interests are in the field of computa-

tional intelligence, flight mechanics and aerodynamics. He has published some technical papers in journals and conference proceedings.

Gregor Papa is a researcher at the Computer Systems Department of the Jožef Stefan Institute, Ljubljana, Slovenia. He received his M.S. and Ph.D. degrees in Electrical Engineering from the University of Ljubljana, Slovenia, in 2000 and 2002, respectively. His current research interests are in the field of optimization techniques and metaheuristic algorithms. He has published several papers in journals and conference proceedings.

Silvia Poles is product manager in Applied Mathematics at ESTECO srl, AREA Science Park - Trieste. She completed her M.S. degree in Mathematics at University of Padua in 1996 and she is completing a biennial master in Modelling and Simulation of Complex Realities at the International Center for Theoretical Physics in Trieste. Her current research interests are in the field of multi-objective optimization and response surfaces methods. She has published several technical reports for ESTECO.

Mike Preuss is a research associate at the Computer Science Department, University of Dortmund, Germany, where he also received his diploma degree in 1998. His current research interests focus on the field of evolutionary algorithms and include structured populations, niching and diversity maintenance mechanisms, and real-world applications.

Enrico Rigoni is an applied mathematics researcher at ESTECO srl, AREA Science Park - Trieste. He completed his M.S. degree at University of Trieste in 2002. His current research interests are in the field of multi-objective optimization and design of experiments. He has published several technical reports for ESTECO.

Tea Robič is a research assistant at the Department of Intelligent Systems, Jožef Stefan Institute, Ljubljana, Slovenia. She received her B.S. degree in Applied Mathematics at the University of Ljubljana, Ljubljana, Slovenia, in 2002 and is currently completing a master in Computer Science at the same university. Her research interests include evolutionary algorithms for single and multi-objective optimization and machine learning methods for text processing.

Lutz Schönemann is a scientific assistant at the Systems Analysis Group at the University of Dortmund, Germany. He is a co-worker of the collaborative research center (SFB 531) "Design and Management of Complex Technical

Processes and Systems by Means of Computational Intelligence Methods". His main research interests are in the field of evolutionary algorithms.

Christine Solnon received the diploma and Ph.D. degrees in Computer Science from the University of Nice-Sophia Antipolis, France, in 1989 and 1993, respectively. Since 1994, she has been an associate professor with the Computer Science Department of the University Claude Bernard of Lyon, France. Her current research interests include ant algorithms, constraint satisfaction problems, and more generally combinatorial (optimization) problems.

Jurij Šilc received his Ph.D. in Electrical Engineering from the University of Ljubljana in 1992. In 1980 he joined the Jožef Stefan Institute, where he is now a researcher. At the Institute, he served as the head of the Computer Architecture Laboratory from 1986 to 1994. He is presently the deputy head of the Computer Systems Department. He has published over 150 research papers on subjects including magnetic bubble memories, dataflow computing, algorithm mapping, parallel processing, high-level synthesis, combinatorial optimization, and processor architecture. The latter includes a recent book entitled Processor Architecture: From Dataflow to Superscalar and Beyond. He has been involved in a number of conferences and professional activities concerned with programming languages, parallel processing, computer architectures, and bio-inspired optimization algorithms.

Lars Willmes is a senior consultant at Nutech Solutions GmbH, Dortmund, Germany. He received his computer science diploma form Dortmund University in 2002. His current research interests are in optimization in general and evolutionary optimization in particular. He has published several scientific papers in international conference proceedings.

I

INVITED CONTRIBUTION

EVOLUTION STRATEGIES: BIOINSPIRED OPTIMIZATION FOR ENGINEERING

Thomas Bäck
NuTech Solutions GmbH
Dortmund, Germany
and
Leiden Institute of Advanced Computer Science (LIACS)
Leiden University, The Netherlands
baeck@nutechsolutions.de

Lars Willmes
NuTech Solutions GmbH
Dortmund, Germany
willmes@nutechsolutions.de

Peter Krause
NuTech Solutions GmbH
Dortmund, Germany
krause@nutechsolutions.de

Abstract The basic variants of evolution strategies, a special instance of evolutionary algorithms, are discussed in this paper. Gleaned from the model of organic evolution, evolution strategies are characterized by the additional self-adaptive process that fine-tunes their strategy parameters during optimization. This property is a fundamental ingredient for the application to challenging engineering applications involving resource-intensive simulation runs. For one instance of such applications, the single-criterion and multi-criterion airfoil design problem, the results of an evolution strategy are presented and discussed in this paper.

Keywords: Evolutionary algorithm, Multi-criterion optimization, Airfoil design

1. Introduction

Evolution strategies [1, 10, 13] are one of the main paradigms in the field of *evolutionary computation*, focusing on algorithms for adaptation and opti-

mization which are gleaned from the model of organic evolution. Evolution strategies are nowadays a widely accepted method for optimization in the field of engineering.

In the following sections we will give an overview of the working principles of evolution strategies and then demonstrate its capabilities with an example of the field from airfoil design. Finally, some conclusions are discussed.

2. The Algorithm

2.1 Working Principle

In general, evolutionary algorithms mimic the process of natural evolution, the driving process for the emergence of complex and well adapted organic structures, by applying variation and selection operators to a set of candidate solutions for a given optimization problem. The following structure of a general evolutionary algorithm reflects all essential components of an evolution strategy as well (see e.g. [3]):

Algorithm 1:

$t := 0$;
initialize $P(t)$;
evaluate $P(t)$;
while not *terminate* **do**
  $P'(t) :=$ *variation*$(P(t))$;
  evaluate$(P'(t))$;
  $P(t+1) :=$ *select*$(P'(t) \cup Q)$;
  $t := t + 1$;
end while

In case of a (μ,λ)-evolution strategy, the following statements regarding the components of Algorithm 1 can be made:

- $P(t)$ denotes a population (multiset) of μ individuals (candidate solutions to the given problem) at generation (iteration) t of the algorithm.
- The initialization at $t = 0$ can be done randomly, or with known starting points obtained by any method.
- The evaluation of a population involves calculation of its members quality according to the given objective function (quality criterion).
- The variation operators include the exchange of partial information between solutions (recombination) and its subsequent modification by adding normally distributed variations (mutation) of adaptable step sizes. These step sizes are themselves optimized during the search according to a process called *self-adaptation*.

- By means of recombination and mutation, an offspring population $P'(t)$ of $\lambda \gg \mu$ candidate solutions is generated.
- The selection operator chooses the μ best solutions from $P'(t)$ (i.e., $Q = \emptyset$) as starting points for the next iteration of the loop. Alternatively, a (μ+λ)-evolution strategy would select the μ best solutions from the union of $P'(t)$ and $P(t)$ (i.e., $Q = P(t)$).
- The algorithm terminates if no more improvements are achieved over a number of subsequent iterations or if a given amount of time is exceeded.
- The algorithm returns the best candidate solution ever found during its execution.

In the following, these basic components of an evolution strategy are explained in some more detail. For extensive information about evolution strategies, refer to [1, 10, 13].

Using a more formal notation following the outline given in [14, 12], one iteration of the strategy, that is a step from a population $P^{(T)}$ towards the next reproduction cycle with $P^{(T+1)}$, can be modeled as follows:

$$P^{(T+1)} := opt_{ES}(P^{(T)}) \tag{1}$$

where $opt_{ES} : I^\mu \to I^\mu$ is defined by

$$opt_{ES} := \mathbf{sel} \circ (\mathbf{mut} \circ \mathbf{rec})^\lambda \ , \tag{2}$$

operating on an input population $P^{(T)}$ according to

$$opt_{ES}(P^{(T)}) = \mathbf{sel}(P^{(T)} \sqcup \left(\sqcup_{i=1}^{\lambda} \{\mathbf{mut}(\mathbf{rec}(P^{(T)}))\} \right) \tag{3}$$

(here, $\sqcup$ denotes the union operation on multisets). Equation (3) clarifies that the population at generation $T + 1$ is obtained from P^T by first applying a λ-fold repetition of recombination and mutation, which results in an intermediate population P' of size λ, and then applying the selection operator to the union of $P^{(T)}$ and P'. Recall that the recombination operator generates only one individual per application, which can then be mutated directly.

In the following, both the formal as well as the informal way of describing the algorithmic components will be used as it seems appropriate.

2.2 The Structure of Individuals

For a given optimization problem

$$f : M \subseteq \mathbb{R}^n \to \mathbb{R} \quad , \quad f(\vec{x}) \to \min$$

an individual of the evolution strategy contains the candidate solution $\vec{x} \in \mathbb{R}^n$ as one part of its representation. Furthermore, there exist a variable amount (depending on the type of strategy used) of additional information, so-called *strategy parameters*, in the representation of individuals. These strategy parameters essentially encode the n-dimensional normal distribution which is to be used for the variation of the solution.

More formally, an individual $\vec{a} = (\vec{x}, \vec{\sigma}, \vec{\alpha})$ consists of up to three components $\vec{x} \in \mathbb{R}^n$ (the solution), $\vec{\sigma} \in \mathbb{R}^{n_\sigma}$ (a set of standard deviations of the normal distribution), and $\alpha \in [-\pi, \pi]^{n_\alpha}$ (a set of rotation angles representing the covariances of the n-dimensional normal distribution), where $n_\sigma \in \{1, \ldots, n\}$ and $n_\alpha \in \{0, (2n - n_\sigma) \cdot (n_\sigma - 1)/2\}$. The exact meaning of these components is described in more detail in Sec. 2.3.

2.3 Mutation

The mutation in evolution strategies works by adding a normally distributed random vector $\vec{z} \sim N(\vec{0}, \mathbf{C})$ with expectation vector $\vec{0}$ and covariance matrix $\mathbf{C}^{-1}$, where the covariance matrix is described by the mutated strategy parameters of the individual. Depending on the amount of strategy parameters incorporated into the representation of an individual, the following main variants of mutation and self-adaptation can be distinguished:

- $n_\sigma = 1$, $n_\alpha = 0$: The standard deviation for all object variables is identical (σ), and all object variables are mutated by adding normally distributed random numbers with

$$\sigma' = \sigma \cdot \exp(\tau_0 \cdot N(0,1)) \quad (4)$$
$$x_i' = x_i + \sigma' \cdot N_i(0,1) \quad , \quad (5)$$

  where $\tau_0 \propto (\sqrt{n})^{-1}$. Here, $N(0,1)$ denotes a value sampled from a normally distributed random variable with expectation zero and variance one. The notation $N_i(0,1)$ indicates the random variable to be sampled anew for each setting of the index i.

- $n_\sigma = n$, $n_\alpha = 0$: All object variables have their own, individual standard deviation σ_i, which determines the corresponding modification according to

$$\sigma_i' = \sigma_i \cdot \exp(\tau' \cdot N(0,1) + \tau \cdot N_i(0,1)) \quad (6)$$
$$x_i' = x_i + \sigma_i' \cdot N(0,1) \quad , \quad (7)$$

  where $\tau' \propto (\sqrt{2n})^{-1}$ and $\tau \propto (\sqrt{2\sqrt{n}})^{-1}$.

- $n_\sigma = n$, $n_\alpha = n \cdot (n-1)/2$: The vectors $\vec{\sigma}$ and $\vec{\alpha}$ represent the complete covariance matrix of the n-dimensional normal distribution, where the

covariances are given by rotation angles α_j describing the coordinate rotations necessary to transform an uncorrelated mutation vector into a correlated one. The details of this mechanism can be found in [1] (pp. 68–71) or [11]. The mutation is performed according to

$$\sigma_i' = \sigma_i \cdot \exp(\tau' \cdot N(0,1) + \tau \cdot N_i(0,1)) \quad (8)$$

$$\alpha_j' = \alpha_j + \beta \cdot N_j(0,1) \quad (9)$$

$$\vec{x}' = \vec{x} + N(\vec{0}, \mathbf{C}(\vec{\sigma}', \vec{\alpha}')) \quad (10)$$

where $N(\vec{0}, \mathbf{C}(\vec{\sigma}', \vec{\alpha}'))$ denotes the correlated mutation vector and $\beta \approx 0.0873$.

The amount of information included into the individuals by means of the self-adaptation principle increases from the simple case of one standard deviation up to the order of n^2 additional parameters in case of *correlated mutations*, which reflects an enormous degree of freedom for the *internal models* of the individuals. This growing degree of freedom often enhances the global search capabilities of the algorithm at the cost of the expense in computation time, and it also reflects a shift from the precise *adaptation* of a few strategy parameters (as in case of $n_\sigma = 1$) to the exploitation of a large *diversity* of strategy parameters.

One of the main design parameters to be fixed for the practical application of the evolution strategy concerns the choice of n_σ and n_α, i.e., the amount of self-adaptable strategy parameters required for the problem.

2.4 Recombination

In evolution strategies recombination is incorporated into the main loop of the algorithm as the first variation operator and generates a new intermediate population of λ individuals by λ-fold application to the parent population, creating one individual per application from ϱ ($1 \leq \varrho \leq \mu$) individuals. Normally, $\varrho = 2$ or $\varrho = \mu$ (so-called global recombination) are chosen. The recombination types for object variables and strategy parameters in evolution strategies often differ from each other, and typical examples are *discrete recombination* (random choices of single variables from parents, comparable to uniform crossover in genetic algorithms) and *intermediary recombination* (arithmetic averaging). A typical setting of the recombination consists in using discrete recombination for object variables and global intermediary recombination for strategy parameters. For further details on these operators, see [1].

When $\mu > 1$ is chosen, the recombination operator needs also be specified for a (μ,λ)-evolution strategy .

2.5 Selection

Essentially, the evolution strategy offers two different variants for selecting candidate solutions for the next iteration of the main loop of the algorithm: (μ,λ)-selection and (μ+λ)-selection.

The notation (μ, λ) indicates that μ parents create $\lambda > \mu$ offspring by means of recombination and mutation, and the best μ offspring individuals are deterministically selected to replace the parents (in this case, $Q = \emptyset$ in Algorithm 1). Notice that this mechanism allows that the best member of the population at generation $t + 1$ might perform *worse* than the best individual at generation t, i.e., the method is not *elitist*, thus allowing the strategy to accept temporary deteriorations that might help to leave the region of attraction of a local optimum and reach a better optimum. Moreover, in combination with the self-adaptation of strategy parameters, (μ,λ)-selection has demonstrated clear advantages over its competitor, the (μ+λ) method.

In contrast, the (μ+λ)-strategy selects the μ survivors from the union of parents and offspring, such that a monotonic course of evolution is guaranteed ($Q = P(t)$ in Algorithm 1).

For reasons related to the self-adaptation of strategy parameters, the (μ,λ)-evolution strategy is typically preferred.

2.6 Termination Criterion

There are several options for the choice of the termination criterion, including the measurement of some absolute or relative measure of the population diversity (see e.g. [1], pp. 80–81), a predefined number of iterations of the main loop of the algorithm, or a predefined amount of CPU time or real time for execution of the algorithm.

3. Airfoil Design

3.1 Introduction

Airfoil design provides a wealth of multi-criteria optimization problems. The layout of a wing heavily influences its efficiency regarding e.g. fuel consumption, etc. Efficiency of an airfoil design can not be measured independently of the anticipated use of the wing, since even the most efficient design must still be able to produce enough lift at low speeds to allow a plane to take off. Different flight conditions like starting and landing or cruising at high altitudes, induce different conditions for optimality. This naturally leads to the formulation of a multi-criteria optimization problem where each flight condition (flight point) states its own objective function. The airfoil design problem considered in this study and the resulting objective function are introduced in Sec. 3.2 and 3.3. Section 3.5 introduces basic features of the *Strength Pareto Evolutionary*

Algorithm 2 (SPEA2) and the *Non-dominated Sorting Genetic Algorithm II* (NSGA-II) which for this study represent the state of the art in evolutionary multi-criteria optimization.

3.2 The Design

One of the main characteristics of a wing design is its pressure profile, i.e. the distribution of pressure over the chord. The inverse design problem under study is to find the wing profile that produces a given pressure profile at given flow conditions. The test problem at hand was proposed for the AEROSHAPE (`http://aeroshape.cira.it`) project and consists of two target wing designs, namely the standard NACA0012 wing at typical starting flow conditions and the standard NACA4412 wing at typical cruise flow conditions. The flow conditions for the two wings are given in Table 1; the target wing designs and their respective pressure profiles are shown in Fig. 1.

Table 1. Flow conditions for high lift with the NACA4412 wing and for low drag with the NACA0012 wing.

	High Lift	*Low Drag*
Target wing	NACA0012	NACA4412
Mach number	0.2	0.77
Reynolds number	$5.1 \cdot 10^6$	10^7
Angle of Attack	10.8°	1.0°
c_w	$2.252 \cdot 10^{-2}$	$1.682 \cdot 10^{-2}$
c_a	1.252	0.5794

The optimization goal is the identification of a set of wing designs whose pressure distributions provide a certain performance for the lift off situation at the expense of cruise condition efficiency. This set is supposed to contain designs very similar to the NACA0012 design on the one hand and the NACA4412 design on the other hand as extreme solutions. Ultimately, an engineer would select one design from this collection of wing profiles that fits best to a given aeroplane concept where neither the standard NACA0012 nor the standard NACA4412 would be an optimal choice.

The rational behind using the NACA0012 and NACA4412 designs is to construct a test case that contains all major difficulties of fluid dynamics and its simulation, but at the same time produces verifiable and comprehensive results. In a real world application, the target pressure distribution may be given independently from a standard airfoil.

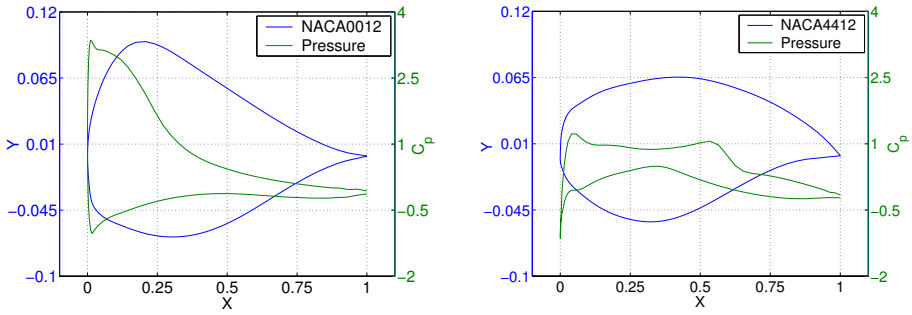


Figure 1. NACA0012 with pressure profile (left) and NACA4412 with pressure profile (right).

3.3 Objective Function

The pressure distribution $p(s)$ at position s of the chord is computed by solving the two-dimensional Navier-Stokes Equations. Together with the pressure p_∞, the density ρ_∞ and the speed $\vec{v}_\infty$ of the surrounding stream the value of

$$c_p(s) = \frac{p(s) - p_\infty}{\frac{\rho_\infty}{2} \cdot \vec{v}_\infty^2} \tag{11}$$

is calculated, so that the two objective functions to be minimized are the cumulated squared differences between the actual pressure profile of the current configuration and the target pressure profiles:

$$f_1(\vec{x}) = \int_0^1 \left[c_p^{\vec{x}}(s) - c_p^{ld}(s) \right]^2 ds \tag{12}$$

$$f_2(\vec{x}) = \int_0^1 \left[c_p^{\vec{x}}(s) - c_p^{hl}(s) \right]^2 ds \tag{13}$$

where s is normalized by the chord length ($s \in [0, 1]$), $c_p^{\vec{x}}$ is the pressure profile of the current design, c_p^{ld} is the low-drag pressure profile of the NACA4412 wing and c_p^{hl} is the high-lift pressure profile of the NACA0012 wing.

Since a single evaluation of the objective function is costly the total number of evaluations was restricted to 1000. Figure 2 give some indication of the quality of the problem: For both objective functions the function values are plotted against the first two design variable dimensions. It is quite clear from Fig. 2 that any optimization algorithm may easily get trapped in local optima.

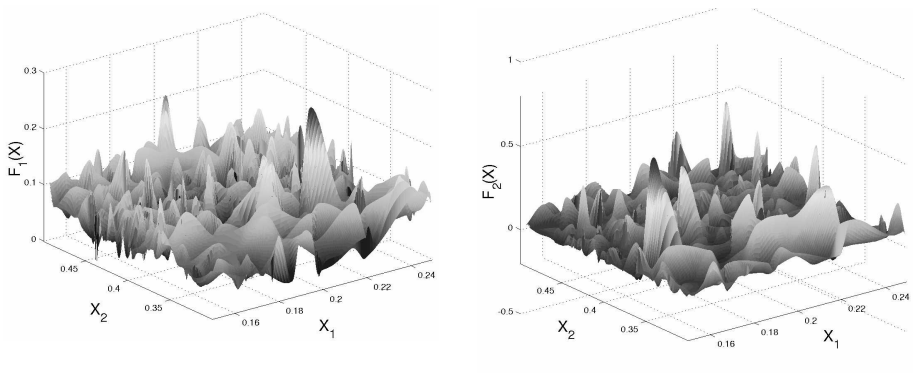


Figure 2. Objective function landscape of f_1 (left) and f_2 (right), projected on the first 2 dimensions.

3.4 Single-Criteria Optimization

The first goal is to re-design one of the extreme solutions introduced in Sec. 3.2. Here, we have chosen the NACA0012 wing for typical starting flow conditions. Thus, we need only the objective function (12).

Starting with a standard implementation of an evolution strategy [2] different parameter settings have been studied. Different numbers of parents and offspring have been tested with standard evolution strategies using comma and plus selection schemes. Varying the proportion of parents to offspring directly influences the selection pressure of the algorithm and the goal of search. Decreasing the proportion leads to a smoother selection pressure and thus to more exploring the search space. On the other hand a larger proportion leads to a larger selection pressure and more exploiting the search space.

Furthermore 1 and n step sizes for the mutation have been tried to have one global step size or one step size in each direction of the search. Different kinds of recombination have been performed, especially intermediate recombination on the strategy parameters in combination with intermediate or discrete recombination on the object variables.

First results showed that the (5+20)-ES seems most promising for further investigations. Recombination was applied to the object variables in a discrete way and to the set of strategy parameters in an intermediate way.

Using derandomized mutation step size control [8] the results could be further improved. This best ES variant outperformed the best values known so far in 90% of all tests.

The airfoil shape and pressure distribution of the best result achieved with the derandomized step size control mutation is presented in Fig. 3. In the left

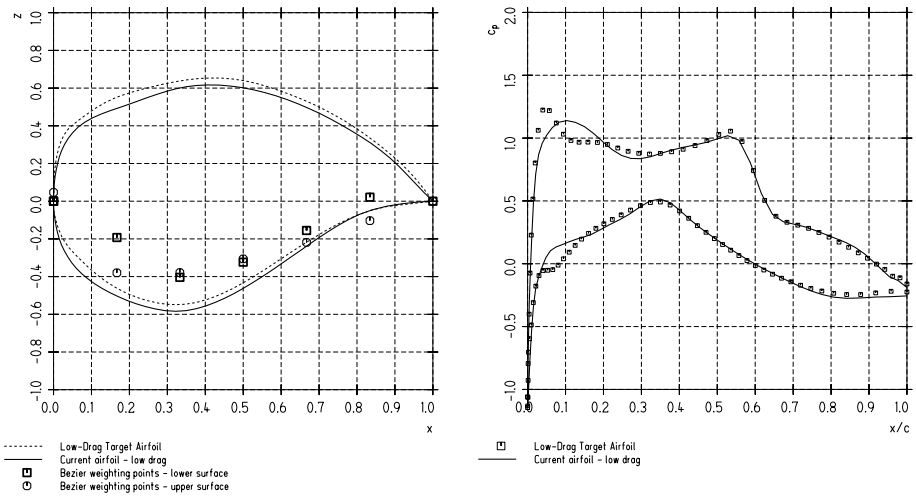


Figure 3. Airfoil shape (left) and pressure distribution (right) found with the ES variant.

part of the figure the airfoil shape is shown and in the right part the pressure can be seen.

A major difference in the airfoil shape between the results so far and the ones presented here is hard to recognize. The experts in the field recognize the advantage of the ES variant in the lower surface. This advantage is much clearer visible in the pressure distribution. Here the pressure generated with the ES variant is much more in coincidence with the given target than the ones so far. This result is in deed 14 % better than the best known so far.

Another major difference is the ability to start the optimization process with randomly chosen individuals, i.e. pressure distributions and airfoil shapes. This fact will become very important when leaving the re-design testcase and looking for really new airfoil designs. Here starting with already predefined airfoils will narrow search to a specific search space area and maybe exclude the best solutions.

3.5 Multi-Criteria Optimization

The main problem in evolutionary multi-criteria optimization lies in the selection operator that chooses the parent individuals for the next reproduction cycle. The single-criteria selection operators of $(\mu + \lambda)$ and (μ, λ) Evolution Strategies rely on the total order of scalar fitness values. Since in general, there is no such total order given for vector valued objective functions, it must be derived by additional selection criteria. SPEA2 and NSGA-II both prefer non-dominated individuals to dominated ones and they both try to establish evenly

spread parent populations by preferring parents in sparsely populated regions of the objective function space.

The SPEA2 operator accumulates information on dominance relationships by summing so called strength values that are computed for an individual by counting the number of individuals it dominates. To enable comparison of individuals with identical strength count, a density value $0 \leq \rho \leq 1$ based on a k-nearest neighbor method is added that is low for sparsely populated regions of the objective function space. Additionally, a special "exclude worst" method based on smallest pairwise distances is applied, if a Pareto front must be reduced to a given size. The details of the algorithm can be found in [6].

The NSGA-II operator uses the non-dominated sorting method to split a population in disjunct Pareto fronts, such that in each front individuals do not dominate each other. NSGA-II then adds new parent individuals front wise, starting with the global Pareto front. If the addition of the next Pareto front would yield more than the prescribed number of new parents, each individual of the current front is assigned a density value based on the city block distance of its closest neighbors. As with SPEA2, individuals from sparsely populated regions of the objective function space are preferred. Details on the methods used in NSGA-II can be found in [4, 5].

Since the computational cost of the CFD-Simulation does not allow numerous experiments even for the two-dimensional models, we show representative results of single runs from a number of experiments. We do not try to average several runs in any way, because the small number of experiments that could be conducted does not allow meaningful statistics. Additionally, from visual inspection, the runs made did not significantly differ from each other.

The pareto-fronts displayed in Fig. 4, 5 and 6 contain a reference Pareto front (denoted "Reference") that is the common Pareto front of all optimization experiments that were conducted for this study and results computed with a multi-criteria genetic algorithm (MOGA) as described in [9]. The reference set was included to show the quality of a single run optimization compared to an aggregated pareto-set that needs much more fitness function evaluations and to supply a benchmark line the individual algorithms have to approach.

Figure 4 demonstrates that SPEA2 and NSGA-II are closer to the reference set than the MOGA. SPEA2 is slightly better in reconstructing the NACA4412 profile, while NSGA-II is slightly more successful for the NACA0012 profile. In the compromise region there is hardly a difference between SPEA2 and NSGA-II.

Figure 5 displays a rather surprising result: The SPEA2 selection operator in combination with the pooling mutation [15] converges close to the reference pareto-set in the compromise region, but fails to place good solutions at the tails of the reference set. The NSGA-II selection operator, contrarily, has a better spread of solutions when combined with pooling mutation, while failing

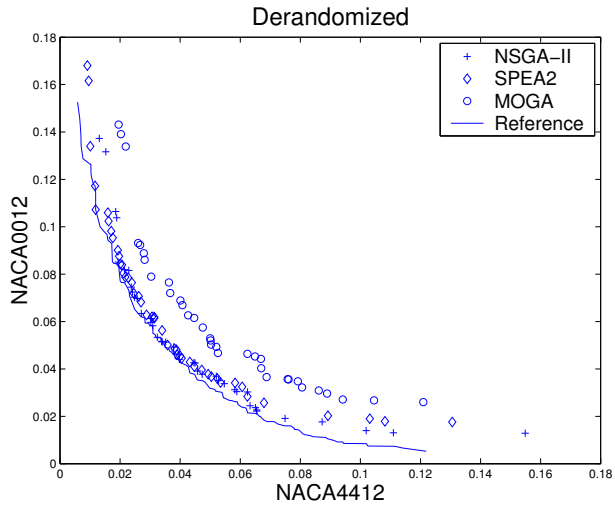


Figure 4. Pareto front with derandomized mutation and NSGA-II- and SPEA2-Selection.

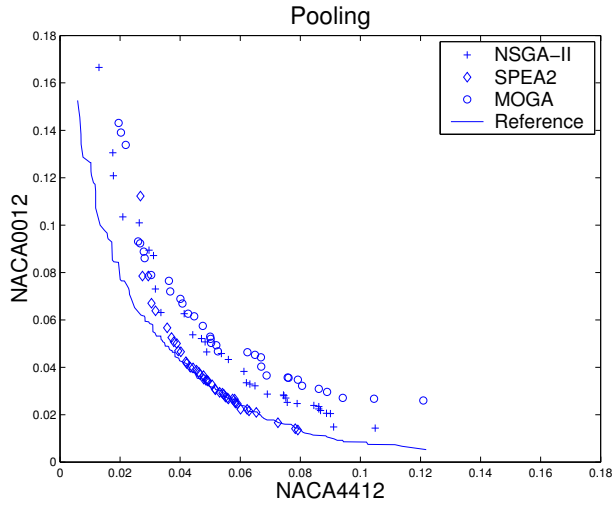


Figure 5. Pareto front with pooling mutation and NSGA-II- and SPEA2-Selection.

to closely approach the reference pareto-set. But still, both Evolution Strategies outperform the MOGA.

Figure 6 finally shows that both selection methods produce nicely spread solutions when combined with Schwefel's mutation [7] which are not as close to the reference set as with the derandomized mutation. Once again, the pareto-set produced by the MOGA is worse than the Evolution Strategies' solutions.

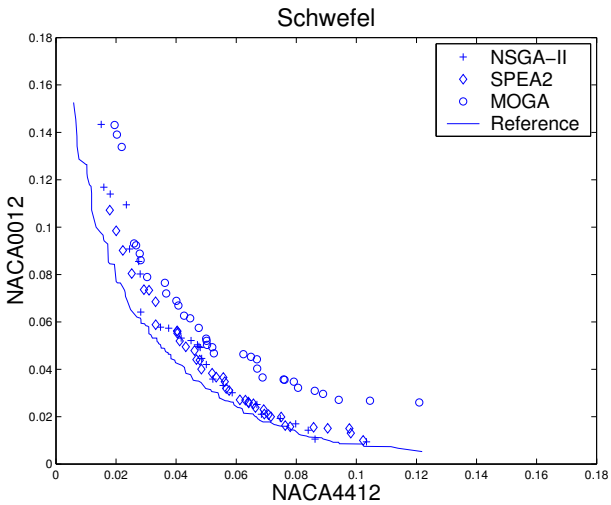


Figure 6. Pareto front with Schwefel's mutation and NSGA-II- and SPEA2-Selection.

It is difficult to make serious judgements on the basis of the few data available. Comparing the results available from the experiments, the derandomiszd mutation operator seems to work very well, but neither Schwefel's mutation nor the pooling mutation are clearly inferior. Clarification of the question which mutation operator to choose will involve further tests with extended computational effort that was beyond the scope of this study.

The same statement holds for the choice of selection method. The results with the pooling mutation as shown in Fig. 5, where the NSGA-II selection provides better spread of solutions than the SPEA2 selection and the SPEA2 selection converges closer to the reference set than the NSGA-II selection, should be considered with some suspicion, as there seems to be no obvious reason for this behavior that could be attributed to the selection method. In fact the effect was less obvious in the other experiments with pooling mutation. For the airfoil design problem, as stated in this study, SPEA2 and NSGA-II perform comparably well.

It can clearly be seen from the results, though, that using NSGA-II or SPEA2 in combination with an Evolution Strategy yields better performance than using the MOGA as described in [9]. This result is supported by all experiments that were made for the airfoil design problem.

4. Conclusions

As demonstrated by numerical experiments on scientific test functions (see e.g., [13]), evolution strategies yield very good optima in case of nonlinear,

high-dimensional global optimization problems. The self-adaptation of strategy parameters (i.e., variances and covariances of the normally distributed mutation operator) is an important and mandatory component of the algorithm to achieve this quality as a global optimization method. Self-adaptation is clearly the most distinguishing feature of evolution strategies, and has been increasingly studied and accepted by the evolutionary computation community over the past decade.

In many practical applications to engineering optimization problems, evolutionary algorithms have been widely neglected for a quite long period of time, mostly because the number of function evaluations - i.e., simulator calls - required seemed unacceptable to practitioners working with these simulators. Given the fact that simulator runtimes can range from several minutes to several hours, and the short project cycles in industry, this seems completely understandable.

Evolution strategies with self-adaptation, however, can be used with very small population sizes (e.g., a (1, 7)-strategy) and few generation cycles, such that they can achieve the desired balance between minimization of resource utilization (i.e., simulator calls) and finding an optimum as good as possible. Indeed, the application to airfoil design problems as discussed in this paper, with 10^3 simulator calls, by far is not representative of the extreme end of that spectrum: At NuTech Solutions, we are currently using evolution strategy variants with as few as around 10^2 simulator calls for high-dimensional problems (around 100 to 150 design parameters), and the results beat any of the optimizers that have been used previously by clients of NuTech Solutions.

These variants of evolution strategies, being under constant further development (both from an academic and a commercial point of view), exploit methods such self-adaptation and meta-modelling and are clearly defining the state-of-the-art in global optimization under tight resource constraints.

References

[1] T. Bäck. *Evolutionary Algorithms in Theory and Practice*. Oxford University Press, New York, 1996.

[2] T. Bäck, D.B. Fogel, and Z. Michalewicz (eds.). *Handbook of Evolutionary Computations*. Oxford University Press, New York, and Institute of Physics Publishing, Bristol, 1997.

[3] T. Bäck, U. Hammel, and H.-P. Schwefel. Evolutionary computation: History and current state. *IEEE Transactions on Evolutionary Computation*, 1:3–17, 1997.

[4] K. Deb. *Multi-Objective Optimization using Evolutionary Algorithms*. Wiley-Interscience Series in Systems and Optimization. John Wiley & Sons, Chichester, New York, 2001.

[5] K. Deb and T. Goel. Controlled elitist non-dominated sorting genetic algorithms for better convergence. In *Proceedings of the First International Conference on Evolutionary Multi-Criterion Optimization (EMO-2001)*, Zurich, Switzerland, 2001, pp. 67–81.

[6] M. Laumanns, E. Zitzler, and L. Thiele. Spea2: Improving the strength pareto evolutionary algorithm. TIK-Report 103, Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH), Zürich, May 2001.

[7] B. Naujoks, L. Willmes, W. Haase, T. Bäck, and M. Schütz. Multi-point airfoil optimization using evolution strategies. In *Proceedings of the European Congress on Computational Methods in Applied Sciences and Engineering (ECCOMAS 2000)*, Barcelona, Spain, 2000.

[8] A. Ostermeier, A. Gawelczyk, and N. Hansen. Step-size adaptation based on non-local use of selection information. In Y. Davidor, H.-P. Schwefel, and R. Männer (eds.): *Parallel Problem Solving from Nature — PPSN III International Conference on Evolutionary Computation*, *Lecture Notes in Computer Science*, 866:189–198, 1994.

[9] C. Poloni. Multi objective aerodynamic optimisation by means of robust and efficient genetic algorithm. In Kozo Fujii and George S. Dulikravich (eds.). *Recent development of aerodynamic design methodologies: inverse design and optimization*, *Notes on numerical fluid mechanics*, 68:1–24, Vieweg, Braunschweig/Wiesbaden, 1999.

[10] I. Rechenberg. *Evolutionsstrategie '94*, *Werkstatt Bionik und Evolutionstechnik*, vol. 1, Frommann–Holzboog, Stuttgart, 1994.

[11] G. Rudolph. On correlated mutations in evolution strategies. In R. Männer and B. Manderick (eds.): *Parallel Problem Solving from Nature 2*, Elsevier, Amsterdam, 1992, pp. 105–114.

[12] H.-P. Schwefel and T. Bäck. Artificial evolution: how and why? In D. Quagliarella, J. Périaux, C. Poloni, and G. Winter (eds.): *Genetic Algorithms in Engineering and Computer Science*, Wiley, Chichester, 1997, ch. 1, pp. 1–19.

[13] H.-P. Schwefel. *Evolution and Optimum Seeking*. Sixth-Generation Computer Technology Series. Wiley, New York, 1995.

[14] H.-P. Schwefel and G. Rudolph. Contemporary evolution strategies. In F. Morán, A. Moreno, J. J. Merelo, and P. Chacón (eds.): *Advances in Artificial Life. Third International Conference on Artificial Life*, *Lecture Notes in Artificial Intelligence*, 929:893–907, 1995.

[15] M. Schütz and J. Sprave. Application of parallel mixed-integer evolution strategies with mutation rate pooling. In L.J. Fogel, P.J. Angeline, and T. Bäck (eds.): *Proceedings of the 5th Annual Conference on Evolutionary Programming (EP-96)*, San Diego, USA, 1996, pp. 345–354.

II

THEORY AND ALGORITHMS

MAXIMAL LIFE SPAN IN EVOLUTIONARY ALGORITHMS

Lutz Schönemann
Systems Analysis Group
University of Dortmund, Germany
lutz.schoenemann@cs.uni-dortmund.de

Abstract In contemporary evolutionary algorithms a main aspect is the capability of adaptation. Especially evolution strategies exhibit of an ability that is called self-adaptation. It is well-known that the selection operator has a great influence on the obtainable results. Two options are used for the selection operator that chooses the individuals to survive for the next generation. In the comma strategy every individual can survive only the current generation. In the plus strategy an individual could survive for infinity. In natural systems such a mechanism is unknown. As a starting point, in this study we investigate the influence of different settings for the maximal life span of the individuals on two test functions.

Keywords: Evolutionary algorithms, Maximal life span

1. Introduction

Evolutionary algorithms (EA) rely on the concept of a population of individuals representing potential solutions to a given optimization problem. They mimic nature's behavior by using the evolutionary operators mutation, recombination and selection. An overview of self-adaptive evolution strategies (ES) can be found in [2, 3, 5]. In contrast to the standard strategies Schwefel and Rudolph have given a more generalized variant of evolution strategies [6]. The so called $(\mu, \kappa, \lambda, \rho)$-ES consists of two additional parameters ρ and κ. The parameter ρ specifies the number of parents that are combined to form a new offspring. The value of κ defines the maximal life span of a single individual. By setting $\kappa = 1$ we get the well-known comma strategy. Setting $\kappa = \infty$ results in the plus strategy. Using a different value than one or infinity accords to the natural paradigm in which the maximal life span is limited. The underlying idea is that fitter individuals might have a longer period to bequest their genetic material. But a possible super-individual can not dominate the population for

the remaining evolutionary process. Thus, self-adaptation is not hampered for too long.

Although a limited life span was introduced into ES in 1995, a complete study of its influence on the obtainable results are not available. We take a first step in this direction. We use an ES presented in [3] and enhanced by a maximal life span κ as suggested in [6]. Keeping this in mind we speak of the (μ, κ, λ)-ES in the rest of this paper. The number of object variables is n (problem dimension) and the number of mutation step sizes is $n_\sigma = 1$ (one strategy parameter). For the experiments we set the number of individuals chosen for recombination to $\rho = 2$. The selection scheme chooses the μ best individuals of the set of parents and offspring if the individual's age is less than the maximal life span. Subsequently the age of the surviving individuals is incremented.

The remainder of this paper is organized as follows. In Sect. 2 a motivation for the usage of different κ is given. Section 3 and 4 present some results on two well-known test functions. Last, Sect. 5 offers concluding remarks.

2. Some Observations on Super-Individuals

Whereas for the plus strategy $\lambda < \mu$ is a valid value, for $\kappa < \infty$ we have to choose $\lambda \geq \mu$. This is necessary because it may occur in one generation that the maximal life span of all parental individuals is expired. In our study in all experiments the population sizes are set to $\mu = 15$ and $\lambda = 100$.

A short consideration of the theoretical background should show the potential problem of the (μ, ∞, λ)-ES. We assume that the current population consists of an individual I_S with a considerably high fitness but inappropriate setting of the strategy parameters. This super-individual dominates the other individuals in a way that the probability is very small that the other individuals produce (with recombination and mutation) an offspring with a higher fitness. In addition, we assume that the offspring generated by I_S have a higher fitness than other offspring. We now show that there exists a high probability that in the next generation the whole population consists only of offspring from I_S. This circumstance may lead to a longer period of stagnation.

To estimate this probability we have to distinguish two cases. In case **(a)** the only genetic operator used is the mutation operator. If we use an equal probability for every parent to be the parent of the next offspring, every parent would produce on mean λ/μ offspring. Using a $(15, \infty, 100)$-ES nearly seven offspring are generated by I_S. Following our assumption regarding the high fitness of the offspring of I_S, all elements are selected for the next parental generation. Without stating the detailed calculation, the probability that every individual of the following generation is a direct offspring of I_S or its children is close to 1. This means that in a $(15, \infty, 100)$-ES after two generations almost every individual is an offspring of the super-individual I_S.

In case **(b)** we use the recombination operator. Here we distinguish between two scenarios. In the first scenario we allow *cloning* which means that both recombination partners are identical. To receive an impression of the belonging probability that the whole next generation consists only of offspring from I_S we calculate the mean number of offspring of a single individual. This number is identical for every individual, so it is for I_S. For every offspring the probability P_1 that I_S is at least one parent is

$$P_1 = 1 - \left(\frac{\mu - 1}{\mu}\right)^{\rho} \quad . \tag{1}$$

With $\rho = 2$ and a total number of λ offspring every individual is involved in generating on average

$$N_1 = \lambda \cdot \left(1 - \left(\frac{\mu - 1}{\mu}\right)^{2}\right) \tag{2}$$

offspring. For the case of a $(15, \infty, 100)$-ES this results in ≈ 12.9 offspring.

For the second scenario of case **(b)** we assume that the recombination operator uses two different parents. The probability that I_S is chosen as one parent for a single offspring is

$$P_2 = 1 - \left(\frac{\mu - 1}{\mu} \cdot \frac{\mu - 2}{\mu - 1}\right) = \frac{2}{\mu} \tag{3}$$

and the mean number of offspring in which I_S is involved is

$$N_2 = \lambda \cdot \left(\frac{2}{\mu}\right) \quad . \tag{4}$$

With $\mu = 15, \lambda = 100$ this equals ≈ 13.3, where the mean number of generated offspring is a little bit higher than in the case of possible cloning. The difference between N_2 and N_1 follows the simple equation

$$N_2 - N_1 = \lambda \cdot \left(\frac{2}{\mu}\right) - \lambda \cdot \left(1 - \left(\frac{\mu - 1}{\mu}\right)^{2}\right) = \frac{\lambda}{\mu^2} \quad . \tag{5}$$

To sum up, using the $(15, \infty, 100)$-ES, in the next generation a huge part of the population may consist of offspring of a super-individual I_S.

In contrast to the calculations above we now assume that due to bad settings of the strategy parameters the offspring generated by I_S have such a minor fitness that they are not selected for the next parental population. In this case the number of generated offspring by I_S is still the same. What changes is the number offspring from which the next generation is selected. In case **(b)** this

number is reduced to $\lambda - N_1$ or $\lambda - N_2$ respectively, depending on the chosen recombination operator. It has already been shown that even slightly different population sizes may lead to significant different results [4]. An assumption is, that a limited life span may reduce this attribute. Hence, it would be helpful to investigate the influence of a limited life span. This is done by the experiments in the following two sections. For all experiments the number of reproduction cycles is set to $g = 1000$. To reduce statistical fluctuations, for each strategy we perform 31 independent single runs. In every run we collect the function value of the best individual of the last reproduction cycle.

3. Life Span on the Sphere

For the first test series we conduct experiments on the well-known sphere model

$$f_{\text{sphere}}(x) = \sum_{i=1}^{n} x_i^2 \ . \tag{6}$$

At first, we use a $(15, \infty, 100)$-ES with $n_\sigma = 1$ mutation step size on the 30-dimensional sphere model. The initial population is spread around $x_i^{(0)} = 10^6$ and the mutation step size is set to $\sigma^{(0)} = 10^5$. We are interested in the distribution of the life span of every individual. We call an offspring an individual if it is chosen as a parent for at least one reproduction cycle. Therefore, we compute the frequency distribution of the life span of all individuals. The result is depicted in Fig. 1. As most of the individuals (about 74%) died after one reproduction cycle and no individual lived longer than eight reproduction cycles it does not make any sense to fit a statistical distribution. Instead it should be adequate to say that the median life span of all individuals is $\tilde{\kappa} = 1$ and the mean is only slightly higher ($\overline{\kappa} \approx 1.3$).

In the following paragraphs we investigate the effects of a limitation of the life span to different values. To do this we use a $(15, \kappa, 100)$-ES with varying κ. As above, we initialize the starting population far away from the optimum and provide for equal test conditions.

An interesting question is how many of the $g \cdot \lambda$ offspring are selected to become parents. In a comma strategy this fraction is maximal because in every reproduction cycle we select μ individuals from the λ offspring. Thus, the fraction is μ/λ or exactly 15%. Because in the $(15, \infty, 100)$-ES most of the individuals died after one reproduction cycle this fraction is near to the maximal fraction (about 11%). Hence, for this function the influence of a limited life span should have only a marginal effect on the results.

The distribution of the function values of a $(15, 1, 100)$-ES are depicted in Fig. 2. On the sphere it is appropriate to compute the logarithm of these function values and to plot them in a histogram. The histogram is not sufficient to reflect the corresponding density. As a consequence we add a simple continuous

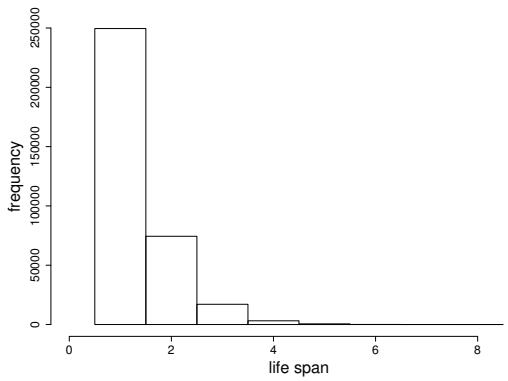


Figure 1. Distribution of the individuals' life span of a $(15, \infty, 100)$-ES on the sphere. The distribution is drawn from 31 independent single runs over 1000 reproduction cycles each.

estimation of the probability density function. At this point we do not want to present a detailed introduction to statistical theory. Instead, we refer to the relevant literature (e.g., [7, 8]). But it should be remarked that we used a Gaussian kernel function. This kernel function serves the purpose for an easy comparison even in the case when the underlying data are skewed and not normally distributed.

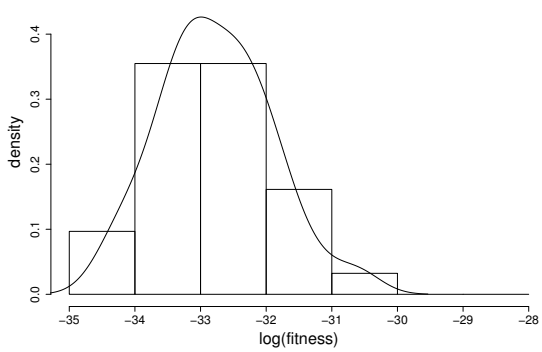


Figure 2. Distribution of the final function values of a $(15, 1, 100)$-ES on the 30-dimensional sphere. 31 independent runs were performed. Additionally to the histogram, a continuous estimation of the probability density distribution is plotted.

Plotting the density functions of different strategies in one figure leads to a simple technique for comparing these strategies. For the experiments in this study we concentrate on the values $\kappa = 1, 2, 3, 5, 10, \infty$. The results for these settings are shown in Fig. 3. Without performing a statistical test we could see

that the density estimators show significant differences. In these experiments a $(15, 1, 100)$-ES leads to the best results of the strategies tested. Furthermore, with increasing κ the strategies lead to more worse results. The worst results are reached with the plus strategy.

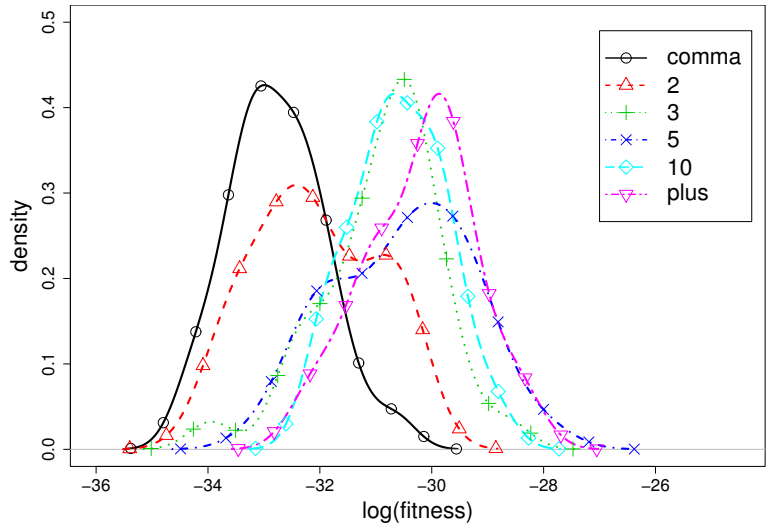


Figure 3. Estimated probability density distribution of the final function values of a $(15, \kappa, 100)$-ES after 1000 reproduction cycles each on the 30-dimensional sphere using 31 independent runs.

Restricting the life span should result in different frequencies of the life time of the individuals. But a closer look at the belonging distribution shows that they are similar to the plus strategy. For $\kappa = 3$ the frequency for age 3 is nearly the same as the sum of the frequencies for ages $3, 4, 5, \ldots$ of the plus strategy. Moreover, the number of offspring which survived for at least one reproduction cycle was always nearly identical.

4. Life Span on the Ackley Function

The Ackley function

$$f_{\text{Ackley}}(x) = -a \cdot \exp\left(-b \cdot \sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n} \cos(cx_i)\right) + a + e \ , \qquad (7)$$
$$a = 20, b = 0.2, c = 2\pi \ ,$$

is an often used function with a moderate multi-modal structure. As Bäck pointed out, the Ackley function should cause only moderate complications to an optimization algorithm [1].

In our experiments we spread the initial population near a local optimum ($x_i = 25$). Additionally, we choose a non-optimal initialization of the mutation step size ($\sigma = 5$). So, the ES must adapt the step sizes and then leave the attractor of the local optimum. As it is often the case for multi-modal landscapes, we assume that a plus strategy stagnates for some time in the attractor of the local optimum until it could escape, whereas we hope that other variants with $1 \leq \kappa < \infty$ are able to perform better.

Figure 4 shows the estimated probability density distribution of the final function values if $n = 30$. The differences between the strategies are only marginal and all strategies were able to obtain an acceptable accuracy. The

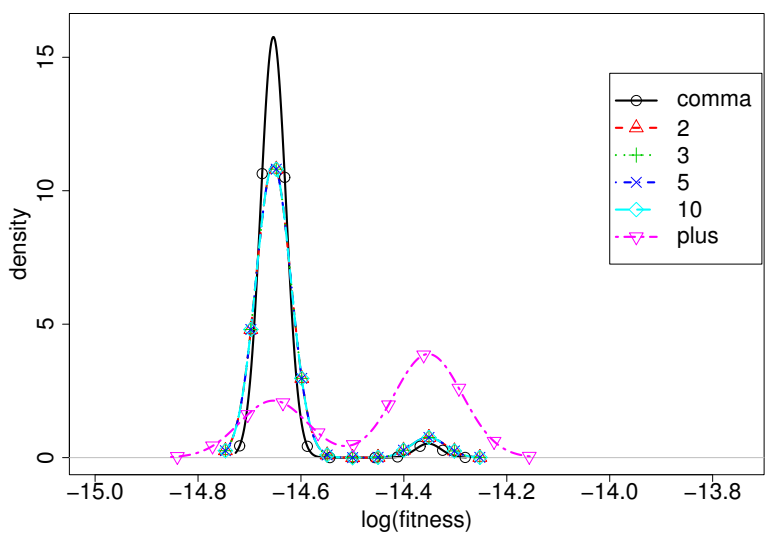


Figure 4. Estimated probability density distribution of the final function values of a $(15, \kappa, 100)$-ES after 1000 reproduction cycles each on the 30-dimensional Ackley function using 31 independent runs.

comparison of the single runs for example $\kappa = 1$ and $\kappa = 3$ shows that the differences of the convergence rate are only small, too. Thus, although the fraction of offspring which became parents in a $(15, \infty, 100)$-ES is only 5%, for this function there were no significant differences observed between the strategies with $\kappa \leq 10$.

To get some insight, we take a closer look at the distribution of life span. In Fig. 5 (left) we see for a plus strategy that some individuals reached a life span of $\kappa = 30$. This indicates that at this moment the adaptation mechanism worked improper. But with an increasing number of reproduction cycles less individuals survived for a longer time. And after 35 reproduction cycles the life span of all individuals are less than four.

In contrast to this, the right figure shows the distribution of life span for a $(15, 10, 100)$-ES. Due to the limited life span the adaptation seems to work better. This is indicated by the fact that the life span of most individuals is less than three already after 25 reproduction cycles. At this time the mutation step size σ has adapted to the problem and in every reproduction cycle some children are generated that are transferred to the next parental population.

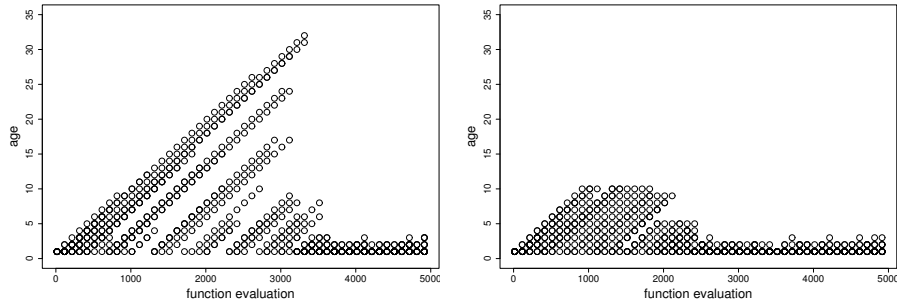


Figure 5. Distribution of the age of the individuals of the population during the first 50 reproduction cycles of a single run of a $(15, \infty, 100)$-ES (left) and a $(15, 10, 100)$-ES (right) on the Ackley function ($n = 30$). Every circle marks at least one individual of the appropriate age.

For both functions used the application of a limited life span led not to better results than a usual comma strategy. Therefore, we try a last experiment on the Ackley function with $n = 100$. The results of the strategies with $1 < \kappa < \infty$ are lying between the results of the other two variants (Fig. 6). With an increasing κ the results grows more and more worse. In this figure the results of the plus strategy are missing because they are not comparable because many of the single runs got stuck in an early local optimum with a very bad function value. For the same problem, Fig. 7 shows the distribution of life span of the plus strategy. Here, only 3.9% of the offspring survived for at least one reproduction cycle.

5. Conclusion

In real applications it could be observed that single individuals could dominate a whole population inhibiting adaptation. For evolution strategies we developed the underlying theory and showed that after one or two generations the whole population could be filled with the offspring of one such a super-individual.

We took a first step to study if the limitation of life span to $1 < \kappa < \infty$ leads to better results. For every function tested here, a comma strategy performed best. For the 100-dimensional Ackley function the conducted experiments indicated an advantage of $1 < \kappa < \infty$ over $\kappa = \infty$. But in the other cases the plus strategy yielded similar results. A deeper look showed, that on the functions tested here, the adaptation mechanism worked appropriate, because most of

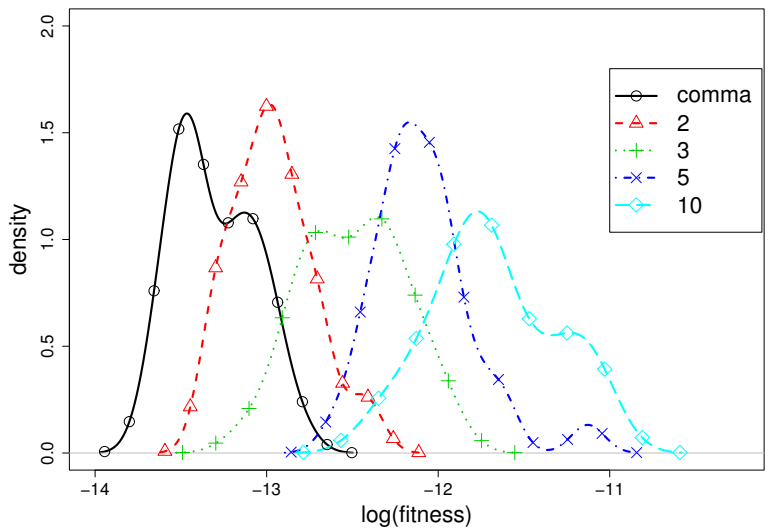


Figure 6. Estimated probability density distribution of the final function values of a $(15, \kappa, 100)$-ES after 1000 reproduction cycles each on the 100-dimensional Ackley function using 31 independent runs.

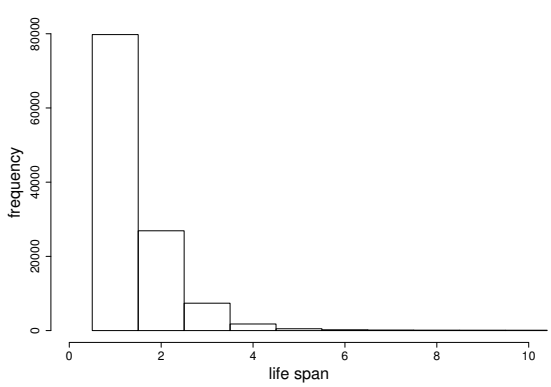


Figure 7. Distribution of the individuals' life span of a $(15, \infty, 100)$-ES on the 100-dimensional Ackley function. The distribution is drawn from 31 independent single runs over 1000 reproduction cycles each.

the individuals were replaced at the latest after two reproduction cycles. A limitation of life span promises to yield better results if many individuals reach a high life span. Further studies must show if this is the case on harder problems.

Acknowledgment

This work was supported by the Deutsche Forschungsgemeinschaft (DFG) as part of the *Collaborative Research Center "Computational Intelligence" (SFB 531)*.

References

[1] T. Bäck. *Evolutionary Algorithms in Theory and Practice*. Oxford University Press, New York, 1996.

[2] T. Bäck and H.-P. Schwefel. Evolution strategies I: Variants and their computational implementation. In G. Winter, J. Périaux, M. Galán, P. Cuesta (eds.): *Genetic Algorithms in Engineering and Computer Science, Proceedings of the First Short Course EUROGEN'95*, Wiley, New York, 1995, pp. 111–126.

[3] H.-G. Beyer and H.-P. Schwefel. Evolution strategies: A comprehensive introduction. *Natural Computing*, 1:3–52, 2002.

[4] L. Schönemann. The impact of population sizes and diversity on the adaptability of evolution strategies in dynamic environments. In *Proceedings of the 2004 Congress on Evolutionary Computation (CEC 2004)*, Piscataway, USA, 2004, pp. 1270–1277.

[5] H.-P. Schwefel. *Evolution and Optimum Seeking*. Wiley, New York, 1995.

[6] H.-P. Schwefel and G. Rudolph. Contemporary evolution strategies. In F. Morán, A. Moreno, J.J. Merelo, P. Chacón (eds.): *Advances in Artificial Life, Proceedings of the Third European Conference on Artificial Life*, Springer, Berlin, 1995, pp. 893–907.

[7] D.W. Scott. *Multivariate Density Estimation: Theory, Practice, and Visualization*. Wiley, New York, 1992.

[8] B.W. Silverman. *Density Estimation for Statistics and Data Analysis*. Chapman and Hall, London, 1986.

ON THE EXTINCTION OF SUB-POPULATIONS ON MULTIMODAL LANDSCAPES

Lutz Schönemann
System Analysis Group
University of Dortmund, Germany
lutz.schoenemann@cs.uni-dortmund.de

Michael Emmerich
System Analysis Group
University of Dortmund, Germany
michael.emmerich@cs.uni-dortmund.de

Mike Preuss
System Analysis Group
University of Dortmund, Germany
mike.preuss@cs.uni-dortmund.de

Abstract Population based evolutionary algorithms (EA) are frequently used to optimize on multimodal functions. A common assumption is that during search several sub-populations might coexist in different attraction regions of the search space. Practical experience and takeover–time considerations suggest that this is not true in general. We therefore analyze the stability of sub-populations within a simplified EA on a two-attractor model, focussing on two extreme cases: (1) Function values of both local minima are exactly the same and (2) function values on the first attractor are always better than on the second. Realistic scenarios for bimodal optimization are assumed to be located in between these two extremes, such that upper and lower bounds for extinction times can be estimated, e. g. by Markov chain analysis and empirical studies. The obtained results provide new insights into the effect of $(\mu \dot{+} \lambda)$ selection on the stability of sub-populations and the effect of genetic drift. Moreover, the effect of idealized niching on the same scenarios is investigated, leading to an immense increase of the EA's ability to perform concurrent search.

Keywords: Evolutionary algorithms, Multimodal fitness landscape, Niching techniques, Random genetic drift, Gambler's ruin

1. Introduction

EA are preferable tools for optimization on multimodal functions. It has often been assumed that the strength of EA stems from the ability to search concurrently in different high performance regions of the search space. Contrary to this, experimental results on multimodal function optimization suggest that EA using the panmictic $(\mu^{+}\lambda)$ selection tend to rapidly concentrate on a single attractor, even if all optima have the same size and function values. It seems impossible to maintain individuals in different regions at the same time, without employing niching techniques.

In this paper we trace back the effect of extinction on neutral landscapes to random genetic drift dynamics, which can be observed in a simplified scenario. Though, dealing with an theoretical issue, the paper provides valuable insights for the practitioner on how to design niching techniques, when searching on multimodal landscapes.

The paper is structured as follows: First, we demonstrate the effect on a simple test-case (2-sphere model) (Sect. 2). Second, we ascribe extinction to random genetic drift dynamics that can be reproduced and analyzed with a simple Markov model which is set up and analyzed in Sect. 3. Based on the theoretical observation we motivate a design principle for niching techniques and demonstrate its benefit on the test case (Sect. 4).

2. Population Dynamics on a Bimodal Test-Case

As an example, the extinction of sub-populations has been observed for the minimization of a simple two-sphere problem $f(\mathbf{x}) = \min((x_1 - 1)^2 + x_2^2, (x_1 + 1)^2 + x_2^2)$ (cf. Fig. 1).

Algorithm 1 describes the $(\mu^{+}\lambda)$-EA [2] that will be studied in this paper. Let $\mathbb{I}$ define the individual space. Each individual $a \in \mathbb{I}$ consists of information on its position in the search space and its objective function value. Furthermore, let $P_t \in \mathbb{I}^{\mu}, Q_t \in \mathbb{I}^{\lambda}$ and $M_t \in \mathbb{I}^{\nu}$ denote multisets of individuals (or *populations*) with $\nu = \mu + \lambda$ for the $(\mu + \lambda)$ selection and $\nu = \lambda$ for the (μ, λ) selection. P_t will be termed the *parent populations*, while Q_t will be termed the *offspring population* for $t = 0, \ldots, t_{\max}$.

The EA starts with initializing the population of parents P_t in the individual space $\mathbb{I}$. Then the following procedure is repeated while the generation counter does not exceed a user defined maximum $t_{\max}$: Generate a multiset of λ offspring by means of variation operators (usually recombination and mutation), then select the best μ individuals out of M_t. Here $M_t = Q_t$ in case of (μ, λ) selection and $M_t = Q_t \cup P_t$ in case of $(\mu + \lambda)$ selection. Finally increase the generation counter and jump to the beginning of the loop.

We make the convention that in case of equal objective function values for M_t, sel(M_t) draws randomly k individuals out of the M_t individuals, without

Algorithm 1: Basic EA

$t \leftarrow 0$
$P_t \leftarrow \text{init}()$ /* Initialize population $P_t \in \mathbb{I}^\mu$*/
while $t < t_{\max}$ **do**
  $Q_t \leftarrow \text{gen}(P_t)$ /* Generate $Q_t \in \mathbb{I}^\lambda$ by variation operators */
  $M_t \leftarrow \begin{cases} Q_t & \text{for } (\mu, \lambda) \text{ selection} \\ Q_t \cup P_t & \text{for } (\mu + \lambda) \text{ selection} \end{cases}$
  $P_{t+1} \leftarrow \text{sel}(M_t)$ /* Select μ best individuals from M_t for P_{t+1} */
  $t \leftarrow t + 1$
end while

choosing one of the individuals twice and without preferring offspring individuals in case of $(\mu + \lambda)$ selection.

Figure 2 shows the average extinction time and probabilities for the $(\mu + \lambda)$-EA. It can be observed that the takeover probability of one sub-population grows proportionally with its ratio in the initial population.

3. Markov Model for the Extinction Dynamics

Imagine an objective function (for minimization) with two large local optimal regions with equal or slightly different optimal function values. In between these plateaus there is a large barrier with extremely high function values (Fig. 3), such that it is very improbable that an individual from one area crosses the barrier by a single mutation. This is similar to the case that the optimization has reached the bottom of two equal or similar local optima of a bimodal function with flat bottoms.

In order to simulate the dynamics of the $(\mu \dot{+} \lambda)$-EA on such a system, let us define the following rules of the game:

For a population P_t at a time t let $\text{black}(P_t)$ define the number of individuals on the first attractor (we will call them *black individuals*). Accordingly, $\mu - \text{black}(P_t)$ individuals are located on the second attractor (we call them *white individuals*). Furthermore, let us assume that all individuals on an attractor have the same function value. Individuals cannot move from one attractor to another attractor or leave their attractor by means of mutation. Hence, the reduced mutation operator simply results in a copy of the individual.

Assuming an initial population with a specified number of black individuals, we are now interested in the dynamics of the simplified EA for the case that (1) the function value for both plateaus is equal and (2) the function value for the plateau that contains the black individuals is better than the function value on the plateau that contains the white individuals. Markov chain analysis can be a powerful tool for understanding simple models of evolution [3, 5]. Next, we provide the reader with the derived Markov chain model.

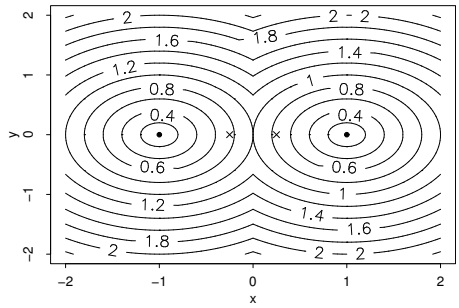


Figure 1. Two-sphere model: Crosses mark starting points for the EA.

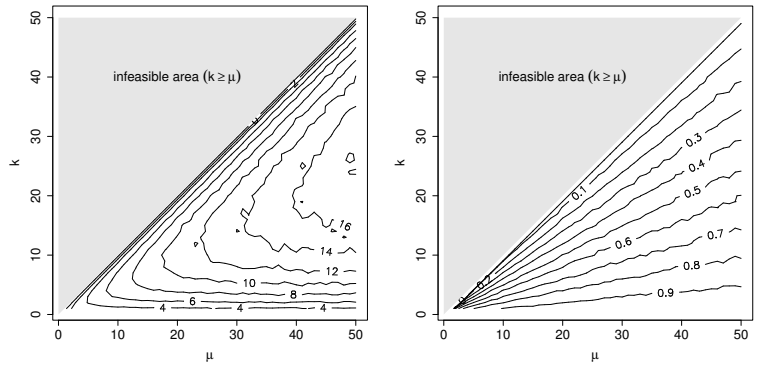


Figure 2. Extinction times (left) and probabilities (right) for a $(\mu + 7\mu)$-ES with Gaussian mutation on the two sphere model (with 2 dimensions) averaged from 5000 runs.

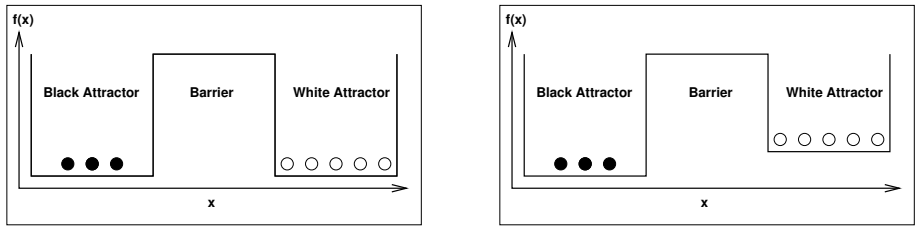


Figure 3. Schematic draw of the instantiations of the *two-attractor model*. The left figure describes the case with equal function values in both attractor basins and the right figure describes the case of better function values for the black individuals than that for the white individuals.

3.1 Deriving the Transition Probabilities

Let k denote the number of black individuals in the initial population. Then we are interested in the probability $p_j(k)$ for j black individuals in the subse-

quent population. This can be obtained by dividing the EA into two steps - the generation of individuals and the selection of P_{t+1}. A possible generational transition could be described as

$$P_t = \{\underbrace{\bullet,\ldots,\bullet}_{k},\underbrace{\circ,\ldots,\circ}_{\mu-k}\} \quad \overrightarrow{\text{generate}} \quad Q_t = \{\underbrace{\bullet,\ldots,\bullet}_{l},\underbrace{\circ,\ldots,\circ}_{\lambda-l}\} \tag{1}$$

$$\overrightarrow{\text{replace}} \quad P_{t+1} = \{\underbrace{\bullet,\ldots,\bullet}_{j},\underbrace{\circ,\ldots,\circ}_{\mu-j}\} \tag{2}$$

Then the transition matrix of the whole evolution step reads:

$$\mathbf{P} := (p_{k,j})_{k\in\{0,\ldots,\mu\},j\in\{0,\ldots,\mu\}}\,, \tag{3}$$

with

$$p_{k,j} = \sum_{l=0}^{\lambda} p_l^{\text{gen}}(k)\cdot p_j^{\text{sel}}(l,k)\ . \tag{4}$$

Here $p_l^{\text{gen}}(k)$ describes the transition probabilities of the procedure gen(P_t)

$$p_l^{\text{gen}}(k) = \Pr(\text{black}(Q_t) = l | \text{black}(P_t) = k) \tag{5}$$

and $p_j^{\text{sel}}(l,k)$ describes the transition probabilities for the procedure sel(M_t)

$$p_j^{\text{sel}}(l,k) = \Pr(\text{black}(P_{t+1}) = j | \text{black}(P_t) = k \wedge \text{black}(Q_t) = l)\ . \tag{6}$$

The transition probabilities $p_l^{\text{gen}}(k)$ for the generate function are the same for all selection schemes studied here:

$$p_l^{\text{gen}}(k) = \left(\frac{k}{\mu}\right)^l \cdot \left(\frac{\mu-k}{\mu}\right)^{\lambda-l} \cdot \binom{\lambda}{l}\ . \tag{7}$$

Table 1 shows the transition probabilities that are instantiated for different selection methods and assumptions about the function values on the two attractors.

3.2 Markov Chain Analysis

Now, we can apply Markov chain analysis [7] in order to analyze the dynamics of the system. Recall from probability theory, for $t > 0$ and any given state vector $\mathbf{p}_t$ we can calculate the probability distribution for the resulting subsequent state j by means of $\mathbf{p}_{t+1} = \mathbf{P}\cdot\mathbf{p}_t$. The limit value for $\mathbf{p}_t$ as $t \to \infty$ can be obtained by means of the fundamental matrix:

The Markov process of the two-attractor model has absorbing boundaries $k = 0$ and $k = \mu$. If one of the absorbing states has been reached the system remains

Table 1. Selection probabilities $p_j^{sel}(l,k)$ (I is the indicator function).

selection method	*equal function values*	*black better than white*
$(\mu+\lambda)$	$\frac{\binom{k+l}{j}\cdot\binom{\mu+\lambda-k-l}{\mu-j}}{\binom{\mu+\lambda}{\mu}}$	$I(j=\min(\mu,k+l))$
(μ,λ)	$\frac{\binom{l}{j}\cdot\binom{\lambda-l}{\mu-j}}{\binom{\mu+\lambda}{\mu}}$	$I(j=\min(\mu,l))$

stable. The absorption probabilities and mean absorption times correspond to the extinction probabilities and mean extinction times. Both can be derived from the transition matrix $\mathbf{P}$ and the initial state k_0. First, let us partition the transition matrix as follows:

$$\mathbf{P} = \begin{pmatrix} 1 & 0 & \dots & 0 & \dots & 0 & 0 \\ a_{1,1} & c_{1,1} & \dots & c_{1,j} & \dots & c_{1,\mu-1} & a_{1,2} \\ \vdots & \vdots & & \vdots & & \vdots & \vdots \\ a_{k,1} & c_{k,1} & \dots & c_{k,j} & \dots & c_{k,\mu-1} & a_{k,2} \\ \vdots & \vdots & & \vdots & & \vdots & \vdots \\ a_{\mu-1,1} & c_{\mu-1,1} & \dots & c_{\mu-1,j} & \dots & c_{\mu-1,\mu-1} & a_{\mu-1,2} \\ 0 & 0 & \dots & 0 & \dots & 0 & 1 \end{pmatrix} . \quad (8)$$

Now, the fundamental matrix $\mathbf{T}$ of the transition matrix $\mathbf{P}$ reads:

$$\mathbf{T} := (\mathbf{I} - \mathbf{C})^{-1} \ , \quad (9)$$

and from Markov chain theory ([7], Chap. 3) an expression for the extinction of black individuals , i. e. for reaching the absorbing state $k = 0$, can be derived as

$$p_E(k_0) = \sum_{i=1}^{\mu-1} a_{i,1} t_{k_0,i}, \quad k_0 = 1, \dots, \mu-1 \quad (10)$$

It is also known that $t_{i,j}$ of $\mathbf{T}$ equals the mean number of iterations that the system is in state i when starting in state j before absorption takes place. Thus

$$E(k_0, \mu) = \sum_{i=1}^{\mu-1} t_{k_0,i}, \quad k_0 = 1, \dots, \mu-1 \quad (11)$$

is the mean absorption time, or - translated to our model - the average time that two species in the evolutionary system can coexists when working with the generational transition described by $\mathbf{P}$ and starting with k individuals.

4. Analysis of Selection Mechanisms

Now, we can use the Markov chain techniques proposed in the previous section to determine some characteristics of selection mechanisms on the two-attractor model. We start with the case of equal fitness.

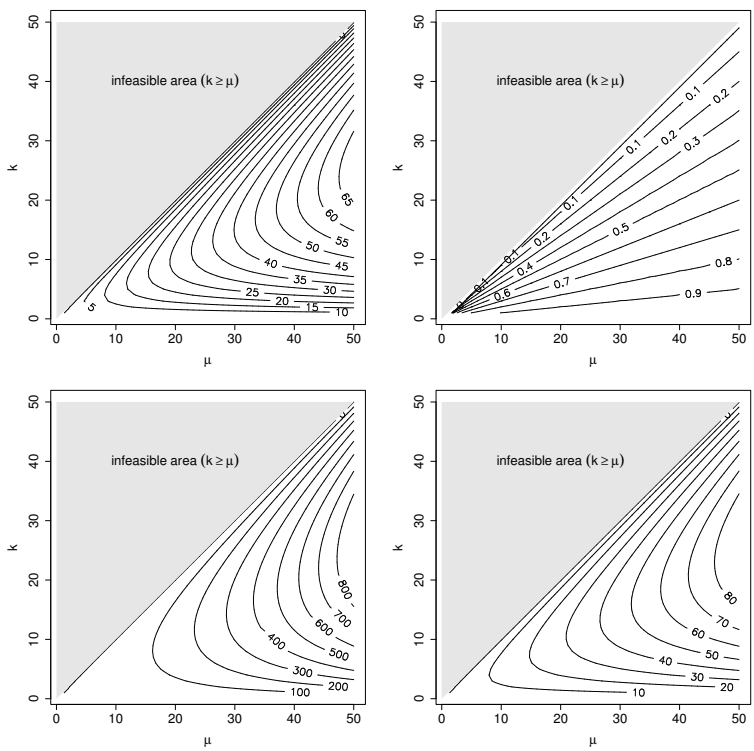


Figure 4. Expected extinction times (EET) and probabilities (PE) obtained by Markov theory. Upper left: EET $(\mu + 7\mu)$-EA, Upper right: PE $(\mu + 7\mu)$-EA, Lower left: EET $(\mu + 1)$-EA, Lower right: EET $(\mu + \mu)$-EA.

Figure 4 shows the mean EET and PE for some frequently used EA strategies. The figure reveals that the extinction times increase linearly with a growing μ if λ and k are set as a constant proportion of μ. Note, that the extinction times are measured in generations. In the case of $(\mu + 7\mu)$ selection and $\mu = 40, k = 20$ this means that one population dies out on average after 53 generations or about $15,000$ offspring. Contrary to this, for the $(\mu + 1)$ selection and the same settings for μ and k only 550 offspring are generated until one population dies out.

In addition to the mean extinction time we are interested in the probability of extinction for one population. As a closer look at the underlying data of the

upper right diagram in Fig. 4 reveals, the extinction probability could be quantified by $\frac{k}{\mu}$. This is an astonishing simple formula and especially independent from λ.

Investigating the model of equal function values gives fundamental insights into the behavior of EA on a bimodal fitness landscape. But it is assumed that a real EA will produce different offspring on a two-attractor landscape. Therefore, we study the two sphere model depicted in Fig. 1 as a more realistic case.

For this function, a Markov chain analysis can not be applied any more. Hence, we obtain the results presented in Fig. 2 by a monte carlo simulation with a real EA. In order to prevent an acceleration of the extinction process cause by recombination, the EA shall only apply a variation operator that works with the mutation operator only. The mutation operator adds a normal distributed random variate to the object variables. The small value of the mutation step size assures that no individual is produced that jumps from one attractor to the other.

The results show that due to the stochastic mutation the extinction times are smaller than in the former model. This is an expected result because this scenario is lying in between the two models of equal and different function values. In contrast to this, the probabilities of extinction are the same.

Our investigations show that a species can die out quickly even if it has equal function values as the other species. To guarantee the survival of a fitter species and to prevent the extinction of sub-populations located on equally shaped attractors one could use several techniques. One of it is niching [4, 6, 8]. Some of the former experiments were repeated with a simple niching technique: Attraction areas to which individuals belong are identified by some clustering approach (cf. [9]) and we generate the same number of offspring individuals. In contrast to many other niching techniques, the suggested selection process is panmictic (alternative selection schemes are presented in [1, Sect. C2]). It was observed that using this kind of simple niching, the sub-populations were able to coexist for a very much longer time (Table 2), even for the more realistic example of the 2-sphere function. Hence, the results in this paper affirm that for optimization on multimodal landscapes niching is a preferable technique to avoid the loss of information gained by sub-populations [9], even in the presence of equality.

The results for some selected test cases are shown in Table 2. It contains the mean extinction times as well as the extinction probabilities. The cases were chosen because they reflect some frequently observed situations and provide the reader with some borderline cases. The mean extinction time shows the number of reproduction cycles both sub-populations survive. In contrast to this p_E measures the probability that the species with the black individuals die out. Some remarks need to be spend on the results of the (16+112)-ES on the two-

sphere model. Due to time limitations the runs were limited to 10^7 generations. Until then, in only 15% of all runs one population died out. On average, a single run lasts longer than 10^6 generations. In the case that one population died out, the black sub-population was eliminated in 52%.

Table 2. Illustrative cases for extinction times and probabilities.

Strategy	*Model*	*Niching*	k	*Simulation Method*	$E(t)$	p_E
(16,112)	boolean	-	1	Markov	2.0	$7 \cdot 10^{-4}$
(16,112)	neutral	-	1	Markov	6.7	0.94
(16,112)	neutral	x	1	Markov	$1.1 \cdot 10^5$	0.5
(16,112)	boolean	x	1	Markov	1.0	0
(4,28)	neutral	x	1	Markov	10.2	0.5
(50+1)	boolean	-	1	Markov	224.0	0
(16,112)	two-sphere	x	1	Experiment	$9.8 \cdot 10^4$	0.48
(16+112)	two-sphere	x	1	Experiment	$> 10^6$	0.52

5. Conclusions

By means of this paper a better understanding of the extinction process on multimodal landscapes has been achieved. In detail our investigations give evidence for the following conjectures:

- For $(\mu \dot{+} 7\mu)$-EA and the equal fitness model the extinction time grows linearly with μ. Thus, even in the best-case scenario the co-existence of populations will not occur for a long time. This result can be interpreted also in a way that mating restrictions alone will not suffice to prevent sub-populations from extinction. This is because in the studies on the simplified model, the recombination has been ommitted and thus it cannot accelerate the extinction of species.

- For different function values and comma selection it was observed that better individuals survive with probability near 1. Hence, the effect of random genetic drift unlikely biases the direction of evolution, if fitness values are clearly different.

- The extinction time for a $(\mu + 1)$-EA are long even in the case of equal function values. However, if we regard the number of function evaluations as a criterion for the extinction time, proportions change and the mean extinction time for the $(\mu + 1)$-EA is significant smaller than that for, e.g., the $(\mu + 7\mu)$-EA.

- The extinction time is increased substantially by using the suggested niching technique.

In this paper only the bimodal case has been considered. However, we conjecture that similar results can be obtained for landscapes with more than two attractors. Future research will have to clarify this point. Furthermore, the effect of the recombination operator deserves further attention.

Acknowledgments This work was supported by the Deutsche Forschungsgemeinschaft (DFG) as part of the *Collaborative Research Center* "Computational Intelligence" (SFB 531).

References

[1] T. Bäck, D.B. Fogel, and Z. Michalewicz (eds.). *Handbook of Evolutionary Computations*. Oxford University Press, New York, and Institute of Physics Publishing, Bristol, 1997.

[2] H.-G. Beyer and H.-P. Schwefel. Evolution strategies: A comprehensive introduction. *Natural Computing*, 1:3–52, 2002.

[3] U. Chakraborty, K. Deb, and M. Chakraborty. Analysis of selection algorithms: A markov chain approach. *Evolutionary Computation*, 2:133–167, 1996.

[4] K. Deb and D.E. Goldberg. An investigation of niche and species formation in genetic function optimization. In J.D. Schaffer (ed.): *Proceedings of the 3rd International Conference on Genetic Algorithms ICGA 89*, San Mateo, USA, 1989, pp. 42–50.

[5] D.E. Goldberg and P. Segrest. Finite markov chain analysis of genetic algorithms. In J.J. Grefenstette (ed.): *Proceedings of the 2nd International Conference on Genetic Algorithms ICGA 87*, Cambridge, USA, 1987, pp. 1–8.

[6] J. Horn. Finite markov chain analysis of genetic algorithms with niching. In S. Forrest (ed.): *Proceedings of the Fifth International Conference on Genetic Algorithms ICGA 93*, San Mateo, USA, 1993, pp. 110–117.

[7] J.G. Kemeny and J.L. Snell. *Finite Markov chains*. D. Van Nostrand Company, Ltd., London, 1969.

[8] S.W. Mahfoud. Niching Methods for Genetic Algorithms. PhD thesis, University of Illinois at Urbana Champaign, 1995.

[9] F. Streichert, G. Stein, H. Ulmer, and A. Zell. A Clustering Based Niching EA for Multimodal Search Spaces. In *Proceedings of the 6th International Conference on Artificial Evolution*, Marseille, France, 2003, pp. 169–180.

PARAMETER CONTROL IN EVOLUTIONARY ALGORITHMS BY DOMAIN-SPECIFIC SCRIPTING LANGUAGE PPC$_{EA}$

Shih-Hsi Liu
Department of Computer and Information Sciences
University of Alabama at Birmingham, USA
liush@cis.uab.edu

Marjan Mernik
Faculty of Electrical Engineering and Computer Science
University of Maribor, Slovenia
marjan.mernik@uni-mb.si

Barrett R. Bryant
Department of Computer and Information Sciences
University of Alabama at Birmingham, USA
bryant@cis.uab.edu

Abstract Programmable Parameter Control for Evolutionary Algorithms (PPCEA), a domain-specific scripting language, solves the problems of control parameter settings in a programmable fashion. It keeps the evolutionary algorithm simple and lifts the problems of control parameter settings into a higher abstraction layer by using metaprogramming. From our experiments, PPCEA outperforms the trial-and-error approach and performs the adaptable, reusable and controllable solutions of control parameter settings for evolutionary algorithms in parameter tuning, deterministic, and adaptive aspects.

Keywords: Evolutionary algorithms, Parameter tuning, Parameter control, Domain-specific scripting language

1. Introduction

It has long been acknowledged that the parameters that control an Evolutionary Algorithm (EA) such as the population size, the probability of crossover,

etc., have significant impacts on EA's performance. Therefore, the designer of an EA has a problem with deciding what control parameter settings are likely to produce the best results. Choosing the right parameter values is a time-consuming and tedious task. Researchers can use experience from previous similar application scenarios or follow guidelines described in [10, 16] for particular numeric optimization problems. Most often, for new problems previous experience is not available nor are the above mentioned guidelines applicable. This leads researchers to choose the trial-and-error approach, spending a considerable amount of effort on this task. To make the problem even harder, different values of parameters might be optimal at different stages of the evolutionary process. For example, in the early stages the larger population is needed than in the later stages when fine tuning of sub-optimal solutions is done.

More recently, researchers recognized that specific problems require specific values of control parameters and that a general near-optimal control parameter setting is not appropriate. The problem is well known to the researchers working on EAs [5] and has not been solved sufficiently yet. Various EAs which differentiate substantially from standard EAs have been proposed (e.g., GAVaPS - genetic algorithm with varying population size [1], 1/5 success rule [2] in evolution strategies) to partially solve this problem. All these new ideas and approaches may be incorporated in one's own EA. However, in this case an EA becomes extraordinarily complex and needs to be adapted whenever we decide to implement different strategies for control parameter settings (e.g., from deterministic to adaptive scheme) or whenever a new method is incorporated into the existing algorithm.

In this paper, a novel solution to this problem is presented. We keep EA simple as before and lift the problem of control parameter settings into a higher abstraction layer. A domain-specific scripting language (DSSL) called Programmable Parameter Control for Evolutionary Algorithms (PPC$_{EA}$) has been implemented to address these needs, where control parameters are set in a programmable fashion with small programs which interact with the original EA.

The paper is organized as follows. Section 2 describes the related work. In Sect. 3, we introduce PPC$_{EA}$. Section 4 shows the examples and experimental results. Finally, Sect. 5 addresses the conclusion.

2. Related Work

There have been a variety of studies [5, 6, 7] on determining the best control parameter values to use for EAs. The main problem is to find control parameters, which optimally balance exploration and exploitation. Recommendations on control parameters for a particular set of problems can be found in [10, 16].

In [5] an overview of this problem has been given, where the authors distinguish between parameter tuning and parameter control. In parameter tuning,

the values of control parameters are set before the run of the EA (e.g., Matlab approach [4]), and in parameter control, values are changed during the run. It was argued also that parameter tuning is less appropriate and parameter control is preferred. One of the problems in parameter tuning of the trial-and-error approach is that one control parameter is usually tuned at a time. This leads to sub-optimal values since control parameters interact in a complex way. Furthermore, methods for parameter control have been classified into deterministic, adaptive, and self-adaptive categories [5]: the deterministic category adjusts parameters by deterministic rules; the adaptive category utilizes the feedback of the evolutionary process to control the direction and magnitude of parameters; the self-adaptive category encodes parameters into chromosomes and undergoes mutation and recombination.

Another approach is to consider the search for the best values of control parameters of an EA as an optimization problem itself which can be solved by another EA, leading to a meta-evolutionary approach. Several meta-evolutionary approaches have already been proposed. In [7] the meta-evolutionary approach is used to determine population size, crossover probability, mutation rate, generation gap, scaling window, and selection strategy. In [6] the meta-evolutionary approach is used to determine a large set of 20 components of genetic algorithms for the traveling-salesman problem (TSP). In [12] the meta-evolutionary approach in searching for the best combination of crossover operators for TSP is also described. One of the major shortcomings of the meta-evolutionary approach is too large processing time.

Interesting work is presented in [8] where a parameter-less genetic algorithm is presented. The objective of their work is to provide robust and simple genetic algorithms (GAs) where users are relieved from control parameter settings despite the fact that peak performance can not be achieved. In [8] only selection rate, crossover probability and population size are taken into account. This is achieved by setting the selection rate (s) and crossover probability (p_c) to the predefined values ($s = 4$, $p_c = 0.5$) to avoid very high or very low selection pressure. In some sense, this approach is similar to the De Jong work [10] except that settings are not based on experiments, yet on sound theoretical work on schema theorem. The population size parameter is simply eliminated by iteratively running the GAs with different population size, doubling the population size each time the population converges or by establishing a race among populations of various sizes. However, the authors leave integration of mutation probability into their framework as future work. Another problem with their approach is that specific problems most often require specific parameter settings.

3. PPCEA

In order to solve control parameter settings of EAs in programmable fashion, we have constructed an interpreter for PPCEA following the techniques of [13]. Figure 1 shows the framework of PPCEA. JLex [3] is a lexical analyzer generator for Java, and CUP (Constructor of Useful Parsers) [9] is the parser generator for Java. The cooperation of both JLex and CUP defines the grammar of PPCEA syntactically and semantically. This grammar consists of both EAs' specialized statements and common linguistic elements of programming languages. In addition, CUP helps to define how PPCEA's interpreter executes/interpretes the source code of PPCEA at runtime. Initializing from the bottom of the parse tree of PPCEA's source code, CUP traces up to check the syntax and execute the semantics. As CUP meets EAs' statements (e.g., call_EA), the interpreter processes the EA operations. As CUP encounters linguistic elements common to all programming languages (e.g., if-then-else statement), the interpreter executes the behaviors of these elements. Consequently, the users can acquire the results of control parameter settings of an EA by writing a PPCEA source code regarding desired control parameters. The typical scenario for choosing control parameters is as follows. From parameter tuning, intervals where control parameters produce best results are identified. In the selected intervals, control parameters are then adjusted using appropriate deterministic or adaptive rules expressed in a programmable fashion. Each run of the PPCEA program produces also diagrams that assist users in control parameter settings.

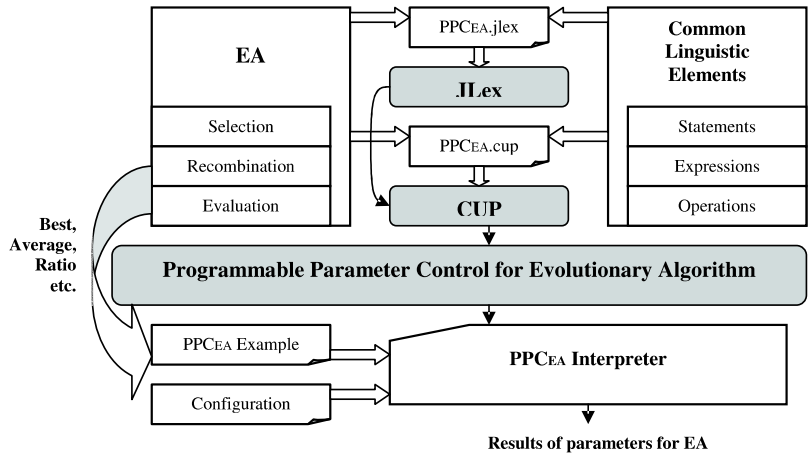


Figure 1. The framework of design and implementation of PPCEA.

PPCEA is in between the system programming language and scripting language [15]. PPCEA is plain, abstract and weakly-typed. Like many other scripting languages (e.g., shell script for UNIX), PPCEA includes simple expressions,

control structures, extended functionality and runtime type checking. However, PPCEA uniquely focuses on the domain of EAs. The approach of writing a PPCEA source code for the interpreter, called metaprogramming, facilitates EAs to hide the implementation details and lift the adaptation of the control parameter settings up to the DSSL level. The major advantage of employing the metaprogramming approach to assess control parameter settings is that EA parameters can be evaluated dynamically by PPCEA's interpreter at runtime. For example, users are able to write if-then-else or while-loop statements to control the parameters at the DSSL level before or during the evaluation process.

In addition, PPCEA provides the configuration mechanism for a fitness function and parameter assignments. The fitness function in general is embedded into the evaluation statement of PPCEA. We use the file processing method to insert the fitness function into an EA. Therefore, users can easily alter the fitness function externally. On the other hand, the configuration mechanism for parameter assignments is equivalent to the predefined identifiers of PPCEA. The configuration mechanism and implementation of DSSL encapsulate the source code of EAs safely and reduce the trial-and-error approach efficiently.

Table 1. Initial values of parameters in the following examples.

Parameter	*Value*	*Parameter*	*Value*	*Parameter*	*Value*
$Maxgen$	500	$Popsize$	50	$Pmutation$	0.1
$Pxover$	0.7	$Epoch$	10	t	0

4. Examples

We had applied the PPCEA approach to solve a routing problem [11] successfully. The routing problem determined the best route of a Mass Transit System from a set of roads. The best solution was decided by a fitness function within a limited budget. The fitness function was the sum of multiplication of weight and attribute (e.g., road length and construction cost) parameters. The PPCEA approach found the best route, which satisfied the constraint, from the roads, and appropriate control parameters. For simplicity, we provide a simpler algebraic fitness function. All examples are tested on the problem of the fitness function in [14],

$$f(x,y) = 0.5 + \frac{\sin^2\sqrt{x^2+y^2} - 0.5}{(1+0.001\cdot(x^2+y^2))^2} \tag{1}$$

where $-100 \leq$ x, and y ≤ 100. The initial values of required parameters are in Table 1. However, $Epoch$ is $Maxgen$ in the Example 1, and $Pxover$ is 0 in Examples 2 and 3. In the following examples, $Pxover$ and $Pmutation$ are the probability rates of crossover and mutation, respectively; $Epoch$ is the counter variable that an EA is executed $Epoch$-times without initialization

of population; Ratio is the successful mutation ratio for 1/5 success rule [2]; Best and Average parameters are the maximum and average fitness values, respectively. For the sake of brevity, we omit the source code of the initialization of these parameters in the following examples.

4.1 Parameter Tuning

Let us start with a simple example where the best probability of crossover and mutation are searched in the predefined range. Values of parameters ($Pmutation$ and $Pxover$) are not changed during the run of EA. This example corresponds to several and tedious runs of basic EAs. Figure 2 (parameter tuning) shows which parameters produce best results.

Example 1:

```
while Pxover ≤ 0.9 do
  Pmutation := 0.1
  while Pmutation ≤ 0.2 do
    init                                   /* initialize population */
    call_EA                                /* run EA for one epoch */
    writeresult
    Pmutation := Pmutation + 0.01
  end while
  Pxover := Pxover + 0.05
end while
```

4.2 Deterministic Control of Mutation Step

Deterministic parameter control employs certain deterministic rule to parameters. The example shows that the mutation step size is controlled deterministically by $P_m(t) = 1 - 0.9(t/T)$, where t is current generation number and T is maximum generation number.

Example 2:

```
init
while t ≤ Maxgen do
  Pmutation := 1 − 0.9(t/Maxgen)
  call_EA                                  /* run EA for one epoch */
  writeresult
  t := t + Epoch
end while
```

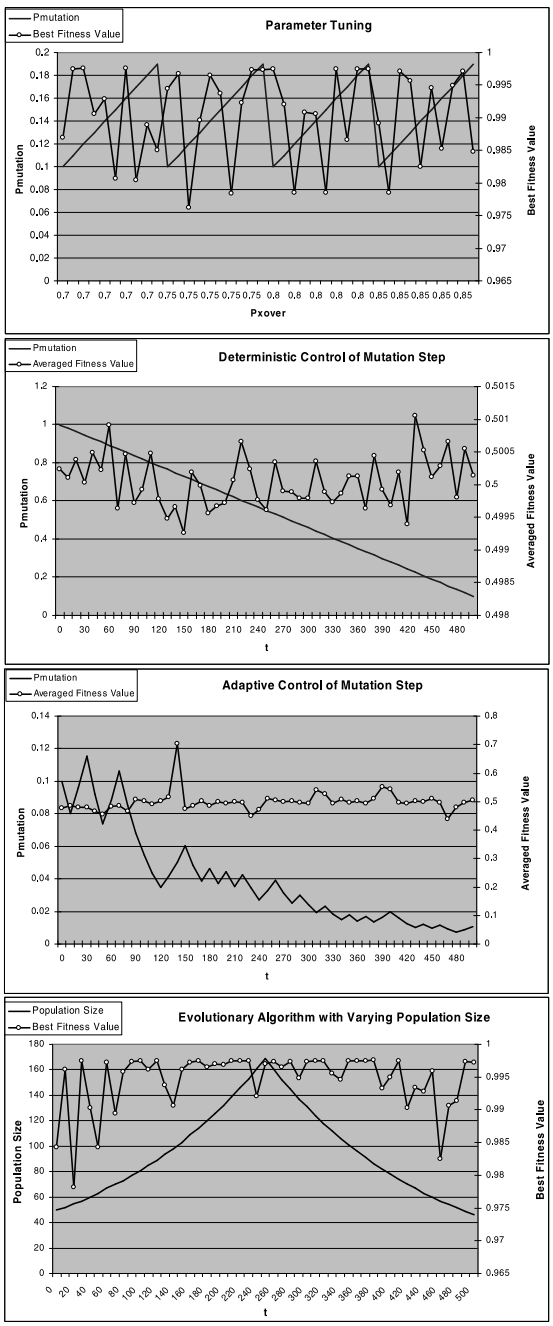


Figure 2. Experimental results of examples: Sect. 4.1, Sect. 4.2, Sect. 4.3, and Sect. 4.4.

4.3 Adaptive Control of Mutation Step

In adaptive control, the feedback from the evolutionary process controls the values of parameters. An example of simple adaptive control is 1/5 success rule [2] where the ratio of successful mutations to all mutations should be 1/5. If the ratio is greater (lower) then the mutation step should be increased (decreased). The example shows that PPCEA lifts the 1/5 success rule up to the DSSL level. $Pmutation$ is controlled adaptively according to the successful mutation ratio.

Example 3:

```
init
while t ≤ Maxgen do
  call_EA
  writeresult
  if Ratio > 0.2 then
    Pmutation := 1.2Pmutation
  else
    Pmutation := 0.8Pmutation
  end if
  t := t + Epoch
end while
```

Self adaptation of the mutation step is also possible. In this case the mutation step size is encoded into the chromosome and as such goes through evolution. Therefore self adaptation is not at the level of PPCEA, but at the level of basic EA. However, the meta-evolutionary approach can be expressed with PPCEA.

4.4 Evolutionary Algorithm with Varying Population Size

In [14, 17], an EA with varying population size is described where the concept of the chromosome age has been introduced. The similar effect can be achieved with the following program in PPCEA. The example shows that $Popsize$ is determined by the relationship of the best and averaged results of an EA. We eliminate the worst chromosomes by sorting if the new $Popsize$ is less than the current one. Yet new members will be generated randomly if the new $Popsize$ is larger than the present one. A more sophisticated formula for $Popsize$ can be programmed exploiting best and average fitness values as well.

Figure 2 shows the experimental results for all examples. Each figure includes average or best values of the fitness function. Necessary parameters such as $Pmutation$, $Pxover$, and $Popsize$ also appear in the figures to assist users with deciding appropriate control parameter settings. From Fig. 2, users can easily acquire the best control parameter setting by analyzing the diagrams. For Example 1, there are various combinations of $Pxover$ and $Pmutation$ to obtain the best fitness value. Example 2 exhibits that the best averaged fitness

Example 4:

```
init
while t ≤ Maxgen do
  call_EA
  writeresult
  if Best < Average * Maxgen/(t + 1) then
    Popsize := 1.05Popsize
  else
    Popsize := 0.95Popsize
  end if
  t := t + Epoch
end while
```

value is obtained when $Pmutation$ is 0.23 at 430^{th} generation. For Example 3, EA computes the highest averaged fitness value where $Pmutation$ and generation are about 0.05 and 130, respectively. Example 4 reveals that diverse population sizes affect the best fitness value at varied stages.

The main advantage of our approach is that the problem of control parameter settings is lifted to the higher abstraction layer while keeping an EA simple as before. All control parameters are treated uniformly (e.g. the probability of crossover, population size, epoch), hence achieving a good flexibility of the proposed approach.

5. Conclusion

PPCEA applies the advantages of domain-specific languages and scripting languages to solve the long-lasting control parameter setting problems. A domain-specific language provides the abstract and repeatable features, since a simple EA evaluation statement (call_EA) covers obligatory selection, recombination and evaluation processes. On the other hand, a scripting language endows EAs with adaptable and controllable characteristics. Therefore, the joint features of PPCEA empower the usage and performance of EAs. PPCEA not only outperforms the trial-and-error approach, but also performs the adaptable, reusable and controllable solutions of control parameter settings for EAs in parameter tuning, deterministic, and adaptive aspects.

References

[1] J. Arabas, Z. Michalewicz, and J. Mulawka. GAVaPS - a Genetic Algorithm with Varying Population Size. In *Proceedings of the 1^{st} IEEE International Conference on Evolutionary Computation*, Orlando, FL, 1994, pp. 73–78.

[2] T. Bäck and H.-P. Schwefel. Evolution strategies I: Variants and their computational implementation. In G. Winter, J. Périaux, M. Galán, P. Cuesta (eds.): *Genetic Algorithms in*

Engineering and Computer Science, Proceedings of the First Short Course EUROGEN'95, Wiley, New York, 1995, pp. 111–126.

[3] E. Berk. *JLex: A lexical analyzer generator for Java*, 1997. http://www.cs.princeton.edu/∼appel/modern/java/JLex/.

[4] A.J. Chipperfield and P.J. Fleming. *Genetic Algorithm Toolbox User's Guide*, 1995. http://www.shef.ac.uk/∼ gaipp/ga-toolbox/manual.pdf

[5] A. Eiben, R. Hinterding, and Z. Michalewicz. Parameter Control in Evolutionary Algorithms. *IEEE Transascition on Evolutionary Computation*, 3:124–141, 1999.

[6] B. Freisleben and M. Härtfelder. In Search of the Best Genetic Algorithm for the Traveling Salesman Problem. In *Proceedings of the 9^{th} International Conference on Control Systems and Computer Science*, Bucharest, Romania, 1993, pp. 485-493.

[7] J.J. Grefenstette. Optimization of Control Parameters for Genetic Algorithms. *IEEE Transaction on Systems, Man & Cybernetics*, 16:122-128, 1986.

[8] G.R. Harik and F.G. Lobo. A parameter-less genetic algorithm. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '99)*, Orlando, FL, 1999, Vol. 1, pp. 258–265.

[9] S.E. Hudson. *CUP parser generator for Java*. 1999. http://www.cs.princeton.edu/∼appel/modern/java/CUP/

[10] K. De Jong. The Analysis of the Behavior of a Class of Genetic Adaptive Systems. PhD thesis, Department of Computer Science, University of Michigan, Ann Arbor, Michigan, 1975.

[11] S.-H. Liu. *Parameterized Genetic Algorithm*. Domain-specific Language: Project Report, 2004. http://www.cis.uab.edu/liush/DSL/final_project.pdf

[12] M. Mernik, M. Črepinšek, and V. Žumer. A Meta-evolutionary Approach in Searching of the Best Combination of Crossover for the TSP. In *Proceedings of the International Joint Conference on Neural Networks (NN '2000)*, Pittsburgh, USA, 2000, pp. 32–35.

[13] M. Mernik, J. Heering, and A.M. Sloane. When and How to Develop Domain-Specific Languages. CWI Technical Report, SEN-E0309, 2003.

[14] Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs*. 3^{rd} Edition. Springer-Verlag, 1996.

[15] J.K. Ousterhout. Scripting: Higher Level Programming for 21^{st} Century. *IEEE Computer*, 31(3):23-30, 1998.

[16] J.D. Schaffer, R.A. Caruana, L.J. Eshelman, and R. Das. A Study of Control Parameters Affecting Online Performance of Genetic Algorithms for Function Optimization. In *Proceedings of the 3^{rd} International Conference on Genetic Algorithms*, Fairfax, USA, 1989, pp. 51–60.

[17] R. Smith and E. Smuda. Adaptively Resizing Populations: An Algorithm, Analysis, and First Results. *Complex Systems*, 1(9):47–72, 1995.

MOGA-II PERFORMANCE ON NOISY OPTIMIZATION PROBLEMS

Silvia Poles
ES.TEC.O S.r.l.
Trieste, Italy
poles@esteco.com

Enrico Rigoni
ES.TEC.O S.r.l.
Trieste, Italy
rigoni@esteco.com

Tea Robič
Department of Intelligent Systems
Jožef Stefan Institute, Ljubljana, Slovenia
tea.robic@ijs.si

Abstract Since the mid-fifties evolutionary algorithms (EAs) have been used in different optimization problems. In the last years their use was extended to the demanding field of multi-objective optimization. For this expansion, EAs themselves had to evolve to more complex forms. The question is whether an algorithm that is adapted to work well with multiple-objectives is still capable to handle single-objective optimization problems. In this paper we present a new EA for multi-objective optimization called MOGA-II. We test it on noisy single-objective problems and compare its performance with two algorithms for single-objective optimization. The results show that MOGA-II is a robust algorithm that can efficiently solve a palette of different optimization problems.

Keywords: MOGA-II, Genetic algorithms, Optimization, Noisy functions

1. Introduction

Evolutionary Algorithms (EAs) are widely used in several optimization problems that are too complex to be solved by traditional methods such as linear programming or gradient based algorithms. Their main advantage over tradi-

tional methods is robustness, which enables efficient optimization of different functions, including noisy ones. Being population-based, EAs can be easily parallelized and can construct multiple optimal solutions in a single run. The latter property is especially welcome in multimodal and multi-objective optimization.

Multi-objective Optimization Problems (MOPs) are more complex than Single-objective Optimization Problems (SOPs) because they involve optimization of several (usually conflicting) objectives. This yields not a single optimal solution but a set of equally important optima, called the *Pareto front*. In multi-objective optimization it is important to guide the search process toward the Pareto front and at the same time maintain adequate population variety to capture as many diverse optimal solutions as possible.

In recent years, EAs have been adjusted in numerous approaches to handle MOPs. Among many algorithms, the NSGA-II of Deb et al. [1] and SPEA2 of Zitzler et al. [10] are the most popular. One of the new EAs for multi-objective optimization is MOGA-II described by Poles in [6], which uses a directional crossover operator for fast convergence and a smart multisearch elitism for uniform spread of solutions. MOGA-II has proved to be very efficient in solving MOPs [7] and in this paper it is tested for robustness. We are raising the question: "Can an EA that was constructed to solve MOPs handle also (noisy) SOPs?" For this purpose we test MOGA-II on five benchmark problems and compare its results with the ones obtained by differential evolution (DE) [8] and a standard EA for single-objective optimization.

The rest of the paper is organized as follows: a detailed description of MOGA-II is presented in Sect. 2 followed by the specification of the experiments in Sect. 3. Section 4 is devoted to the presentation of the experimental results that are further discussed in Sect. 5. The paper ends with a conclusion in Sect. 6.

2. MOGA-II

MOGA-II is an improved version of MOGA (Multi-Objective Genetic Algorithm) by Poloni [5] and is not to be confused with MOGA by Fonseca and Fleming [2] with which it shares only the same acronym. MOGA-II uses a smart multisearch elitism for robustness and directional crossover for fast convergence. Its efficiency is ruled by its operators (classical crossover, directional crossover, mutation and selection) and by the use of elitism. In this paper only the features of MOGA-II that relate to its use in single-objective optimization are explained.

2.1 Encoding

Encoding in MOGA-II is done as in classical genetic algorithms [3]. Each variable is represented as a binary string where the length of the string depends on the base (the number of allowed values for the variable). For example, if only integer values in the interval $[0, 10]$ are to be allowed (11 possible values), the base is set to 11. Thus the length of the string is equal to 4 and the variable can take values from $[0000]$ to $[1011]$. In order to simulate a continuous variable, the base must be set to an appropriate high number.

2.2 Elitism

Elitism is very important in multi-objective optimization because it helps preserving the individuals that are closest to the Pareto front and the ones that have the best dispersion. When optimizing a single objective, the elitism embedded in MOGA-II reduces to copying the solution with the best fitness into the next generation.

2.3 Reproduction

MOGA-II uses four different operators for reproduction (one-point crossover, directional crossover, mutation and selection). At each step of the reproduction process, one of the four operators is chosen (with regard to the predefined operator probabilities) and applied to the current individual. Algorithm 1 shows the reproduction of MOGA in pseudo code.

Algorithm 1: Pseudo code of the reproduction used in MOGA-II

with (individual $Ind_i \in$ generation G) **do**
  choose reproduction operator
  if (operator is *one-point crossover*) **then**
    $j \leftarrow$ TournamentSelection, where $j \neq i$
    $NewInd_i \leftarrow$ OnePointCrossover(Ind_i, Ind_j)
  else if (operator is *directional crossover*) **then**
    $j \leftarrow$ RandomWalk(i)
    $k \leftarrow$ RandomWalk(i), where $k \neq j \neq i$
    $NewInd_i \leftarrow$ DirectionalCrossover(Ind_i, Ind_j, Ind_k)
  else if (operator is *mutation*) **then**
    $NewInd_i \leftarrow$ Mutation(Ind_i)
  else if (operator is *selection*) **then**
    $NewInd_i \leftarrow Ind_i$
  end if
end with

2.3.1 One-Point Crossover. One-point crossover is the most classical operator for reproduction. Two parents are chosen and some portion of the genetic material (the design variables) is exchanged between the parent variables vectors (see Fig. 1). The point of the crossing site is randomly chosen and the binary strings are cut at that point. The two head pieces are then swapped and rejoined with the two tail pieces. From the resulting individuals, usually called children, one is randomly selected to be the new individual.

Figure 1. One-point crossover.

In MOGA-II, one-point crossover starts by taking the current individual Ind_i as the first parent. The second parent Ind_j is chosen by means of a multi-objective tournament selection on a randomly selected population subset: this operator returns the first non-dominated solution in the subset.

2.3.2 Directional Crossover. Directional crossover is slightly different and assumes that a *direction of improvement* can be detected comparing the fitness values of two reference individuals. In [9] a novel operator called *evolutionary direction crossover* was introduced and it was shown that even in the case of a complex multimodal function this operator outperforms classical crossover.

The direction of improvement is evaluated by comparing the fitness of the individual Ind_i from generation t with the fitness of its parents belonging to generation $t-1$. The new individual is then created by moving in a randomly weighted direction that lies within the ones individuated by the given individual and his parents (see Fig. 2). A similar concept can be however applied on

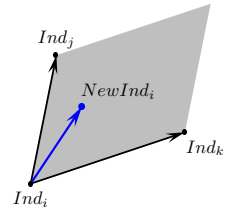


Figure 2. Directional crossover between individuals Ind_i, Ind_j and Ind_k.

the basis of directions not necessarily linked to the evolution but detected by selecting two other individuals Ind_j and Ind_k in the same generation (like shown in Algorithm 1).

Algorithm 2: Random walk from the i-th individual

Input: index i of the starting individual
  $S \leftarrow \emptyset$
  $m \leftarrow \lfloor\sqrt{popSize}\rfloor$;
  for all (N steps) **do**
    $k \leftarrow \lfloor 4 \cdot rand() + 1\rfloor$
    if $(k == 1)$ **then**
      $i \leftarrow i + 1$
    end if
    if $(k == 2)$ **then**
      $i \leftarrow i - 1$
    end if
    if $(k == 3)$ **then**
      $i \leftarrow i - m$
    end if
    if $(k == 4)$ **then**
      $i \leftarrow i + m$
    end if
    if $(i < 1)$ **then**
      $i \leftarrow i + popSize$
    end if
    if $(i > popSize)$ **then**
      $i \leftarrow i - popSize$
    end if
    $S \leftarrow S \cup Ind_i$
  end for
Output: j such that $f(Ind_j) = \min_{Ind \in S} f(Ind)$

The selection of individuals Ind_j and Ind_k can be done using any available selection schema. In MOGA-II local tournament with random steps in a toroidal grid is used. First of all, the individual subject to reproduction is chosen as the starting point. Other individuals met in a random walk of assigned number of steps from that starting point are then marked as possible candidates for the first "parent" Ind_j. The list of all possible candidates for the second "parent" Ind_k is selected in the same way in a successive (and generally different) random walk from the same starting point. When the set of candidates is generated, the candidate with the best fitness is chosen.

The number of steps N in the random walk remains fixed during the entire optimization run and is proportional to the population size. Algorithm 2 shows the random walk with individual Ind_i chosen as a starting point. The function *rand()* generates values in the interval $[0, 1)$ with a uniform distribution.

Directional crossover has demonstrated to help the algorithm convergence for a wide range of numerical problems.

2.3.3 Mutation. Mutation is an operator that ensures *diversity* from one generation to the next. Using plain words we can say that mutation guarantees the algorithm robustness. In MOGA-II it is possible to define the value of the so-called *DNA String Mutation Ratio*. This value gives the percentage of the binary string that is perturbed by the mutation operator.

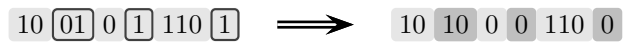


Figure 3. Mutation example with DNA string mutation ratio set to 40%.

3. Experiments

In our experiments we tested the performance of MOGA-II on five numerical single-optimization problems with and without noise. The used numerical benchmark problems are described in Table 1. In all problems the function is to be minimized.

To preserve consistency with the results in [4] we set up the following experiments. Each test function f_i was optimized with and without noise. The noise was introduced as

$$f_i^*(\mathbf{x}) = f_i(\mathbf{x}) + N(0,1)$$

where the $N(0,1)$ is the normal (or Gaussian) distribution with mean 0 and variance 1. The experiments on the noisy functions were run with 5 different number of resamples: $s = 1$, 5, 20, 50 and 100. This means that a solution was evaluated s times and the true value of f_i was estimated by the mean of the samples. In all runs the number of function evaluations was kept constant to provide a fair performance comparison and was calculated as

$$numEval = popSize \times numIt \times s - numUnchanged$$

where $popSize$ is the population size, $numIt$ is the number of iterations, s is the number of resamples and $numUnchanged$ is the number of unchanged individuals during the run. The latter refers to candidate solutions that were evaluated previously in the same run, which we did not re-evaluate if they remained unchanged, such as for example members of the elite.

Each experiment was repeated 3 times. We used $numEval = 100,000$ for low dimensional functions f_1 and f_2 and $400,000$ for the 50 dimensional

Table 1. Benchmark problems.

name	Schaffer F6
dimensions	2
definition	$f_1(\mathbf{x}) = 0.5 + \frac{\sin^2(\sqrt{x_1^2+x_2^2})-0.5}{(1+0.001(x_1^2+x_2^2))^2}$
constraints	$\mathbf{x_i} \in [-100, 100]$
optimum	$\mathbf{x}^* = (0,0)$, $f_1(\mathbf{x}^*) = 0$
name	Sphere
dimensions	5
definition	$f_2(\mathbf{x}) = \sum_{i=1}^{5} x_i^2$
constraints	$\mathbf{x_i} \in [-100, 100]$
optimum	$\mathbf{x}^* = (0,0,0,0,0)$, $f_2(\mathbf{x}^*) = 0$
name	Griewank
dimensions	50
definition	$f_3(\mathbf{x}) = 1 + \frac{1}{4000} \cdot \sum_{i=1}^{50}(x_i - 100)^2 - \prod_{i=1}^{50} \cos\left(\frac{x_i-100}{\sqrt{i}}\right)$
constraints	$\mathbf{x_i} \in [-600, 600]$
optimum	$\mathbf{x}^* = (100, \ldots, 100)$, $f_3(\mathbf{x}^*) = 0$
name	Rastrigin F1
dimensions	50
definition	$f_4(\mathbf{x}) = 500 + \sum_{i=1}^{50}(x_i^2 - 10 \cdot \cos(2\pi x_i))$
constraints	$\mathbf{x_i} \in [-5.12, 5.12]$
optimum	$\mathbf{x}^* = (0, \ldots, 0)$, $f_4(\mathbf{x}^*) = 0$
name	Rosenbrock
dimensions	50
definition	$f_5(\mathbf{x}) = \sum_{i=1}^{49}\left(100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2\right)$
constraints	$\mathbf{x_i} \in [-50, 50]$
optimum	$\mathbf{x}^* = (1, \ldots, 1)$, $f_5(\mathbf{x}^*) = 0$

Table 2. The values of population size and number of iterations for each experiment. The number of unchanged individuals is different in every run, therefore the number of iterations is calculated as $numEval/(popSize \times s)$. The non-noisy versions of the functions had the same values for $popSize$ and $numIt$ as the respective noisy ones with $s = 1$.

functions	s	1	5	20	50	100
$f_1^*,\ f_2^*$	$popSize$	50	50	50	50	25
	$numIt$	2000	400	100	40	40
$f_3^*,\ f_4^*,\ f_5^*$	$popSize$	100	100	100	100	100
	$numIt$	4000	800	200	80	40

functions f_3, f_4 and f_5. The values of $popSize$ and $numIt$ for each experiment are gathered in Table 2.

Table 3 shows the parameter setting for MOGA-II. The parameters were not tuned to any benchmark problem – we used the default values of the algorithm, which demonstrate to perform well in most real-world problems. MOGA-II

encodes its variables as binary strings and therefore needs a discretized space. In this experiments, the base for the encoding was set in such way, that the discretization was 10^{-6}. In the used benchmark problems, the bases have been set respectively to $200,000,001$ for f_1 and f_2, $1,200,000,001$ for f_3, $10,240,001$ for f_4 and $100,000,001$ for f_5. As said in the previous section, such high bases permit to simulate suitably continuous variables.

Table 3. Parameter settings for MOGA-II.

probability of directional crossover	50%
probability of classical crossover	35%
probability of selection	5%
probability of mutation	10%
DNA string mutation ratio	5%

4. Results

The results reported in this section refer to the "true" (non-noisy) evaluation of solutions. We compare the results of MOGA-II with the ones obtained by differential evolution and a standard EA that were published in [4] (see Table 4). The comparison is not completely fair. As said in the previous section, the experiments with MOGA-II were repeated 3 times while the results of DE and EA from Table 4 are the outcome of 30 runs. Moreover, with the high dimensional functions f_3, f_4 and f_5 MOGA-II was given 400,000 evaluations while DE and EA continued until 500,000 evaluations were reached. The reason for this inconsistence is the lack of time we had at our disposal. But it is important to emphasize that this comparison works in favour of DE and EA and not MOGA-II. Time limitations were due to the fact that MOGA-II is embedded into a commercial optimization environment called modeFRONTIER [11]; this Java software contains an advanced post-processing analysis tool, well suited for engineering issues, where the bottleneck is given by the external solvers. But in our case the main problem was handling such a big designs database, and this cannot be quickly done in a non-compiled programming language such as Java.

In Fig. 4 the charts represent the performance of MOGA-II on noisy benchmark problems. Note that there was not enough room to include also the graphs for non-noisy evaluations. Anyhow, with non-noisy functions the optimum was always reached very quickly.

Table 4. Mean and standard deviation of the final results for the benchmark problems (see Table 1). The results of the algorithms DE and EA are taken from [4].

	MOGA-II		*DE*		*EA*	
Function	*mean*	*st. dev.*	*mean*	*st. dev.*	*mean*	*st. dev.*
f_1	0	0	0	0	$3 \cdot 10^{-17}$	0
f_1^* $(s = 1)$	0.04285	0.03488	0.48998	0.00582	0.25829	0.03045
f_1^* $(s = 5)$	0.03533	0.04419	0.40360	0.03030	0.12859	0.01678
f_1^* $(s = 20)$	0.01054	0.00120	0.16597	0.02753	0.06730	0.01066
f_1^* $(s = 50)$	0.01012	0.00049	0.12729	0.01829	0.04769	0.00757
f_1^* $(s = 100)$	0.01927	0.01655	0.09795	0.01203	0.06277	0.00743
f_2	0	0	10^{-152}	0	$7 \cdot 10^{-20}$	0
f_2^* $(s = 1)$	0.01067	0.00581	0.25249	0.02603	0.04078	0.00543
f_2^* $(s = 5)$	0.01077	0.00188	0.13315	0.01266	0.02690	0.00363
f_2^* $(s = 20)$	0.00757	0.00903	0.07364	0.00811	0.02205	0.00290
f_2^* $(s = 50)$	0.00398	0.00238	0.07004	0.00686	0.01765	0.00233
f_2^* $(s = 100)$	0.04436	0.04290	0.08165	0.00800	0.03929	0.00396
f_3	$2 \cdot 10^{-12}$	$2 \cdot 10^{-12}$	0	0	0.00624	0.00138
f_3^* $(s = 1)$	3.29905	0.40864	3.31514	0.07388	1.14598	0.00307
f_3^* $(s = 5)$	2.40897	0.25458	2.42183	0.03616	1.10223	0.00342
f_3^* $(s = 20)$	1.78540	0.35279	2.67093	0.03895	1.44349	0.01381
f_3^* $(s = 50)$	3.78713	0.75639	46.8197	0.96449	3.69626	0.13127
f_3^* $(s = 100)$	14.5960	1.25293	233.802	6.25840	18.0858	0.99646
f_4	0.49748	0.70354	0	0	32.6679	1.94017
f_4^* $(s = 1)$	28.8315	2.33133	2.35249	0.06062	30.7511	1.32780
f_4^* $(s = 5)$	21.6573	2.68711	14.0355	0.47935	31.4725	2.02356
f_4^* $(s = 20)$	45.2687	4.17703	167.628	2.12569	39.1777	2.11529
f_4^* $(s = 50)$	104.415	19.7481	314.762	2.88650	74.8577	2.69437
f_4^* $(s = 100)$	177.847	13.7495	438.036	3.67504	147.800	2.93208
f_5	40.6641	50.5749	35.3176	0.27444	79.8180	10.4477
f_5^* $(s = 1)$	56.5750	39.8582	47.6188	0.15811	118.940	13.2322
f_5^* $(s = 5)$	160.737	56.7030	47.0404	0.13932	341.788	49.6738
f_5^* $(s = 20)$	1601.84	1081.10	7917.46	352.851	1859.06	261.844
f_5^* $(s = 50)$	$1.3 \cdot 10^5$	93260.1	$1.7 \cdot 10^7$	903677	35477.7	4656.17
f_5^* $(s = 100)$	$1.2 \cdot 10^6$	$4.3 \cdot 10^5$	$3.0 \cdot 10^8$	$1.0 \cdot 10^7$	257488	19371.2

5. Discussion

The results in the non-noisy cases are obviously better than the noisy experiments: the noise plays always the role of annoyance factor. As concerns the noisy cases, considering the different values $s = 1$, 5, 20, 50, and 100 for the resampling (and consequently the differences in the number of iterations and/or population size, in order to preserve the total number of evaluated designs), we can expect two different effects to come into play, for "low" and "high" values of s, respectively. Towards the low end, say $s = 1$, the noise gain in importance, so we could expect the results to deteriorate. But also towards the high end, i.e. $s = 100$, the results are expected to be worse, since the request for evaluating

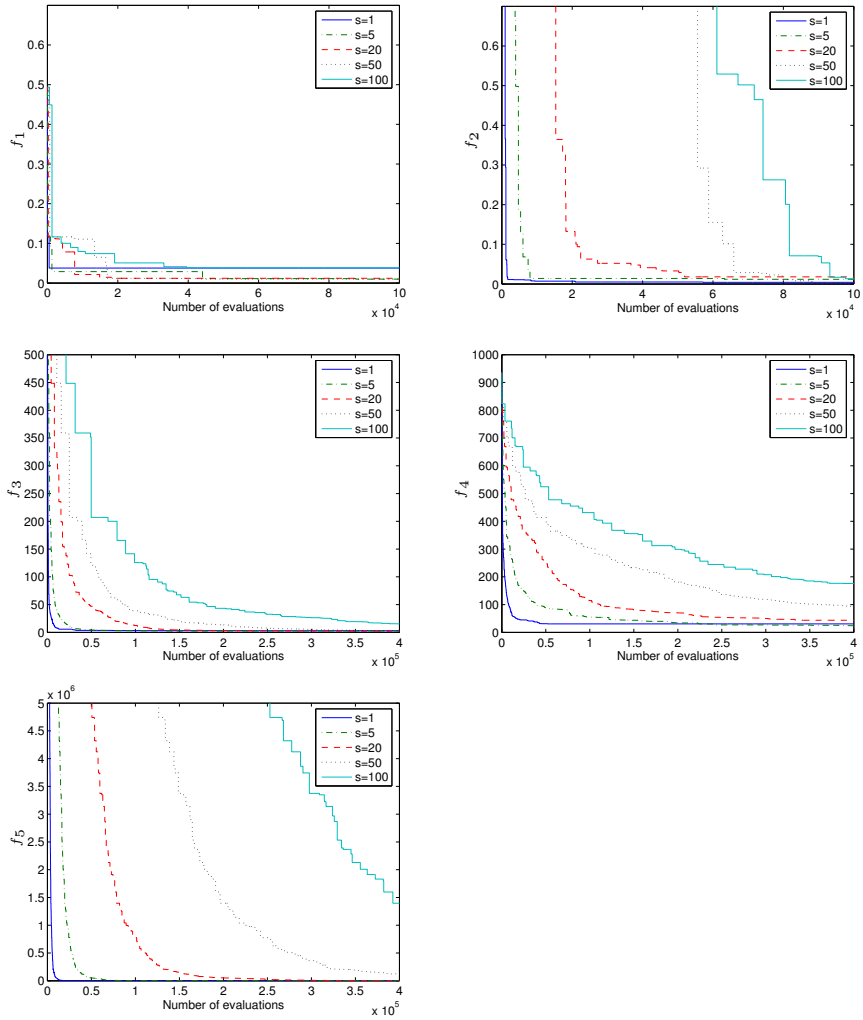


Figure 4. MOGA-II performance on noisy benchmark problems (see Table 1 for function definitions).

the same candidate solution many times, in order to reduce the noise effect, implies that we have to limit consequently the number of iterations, stopping prematurely the optimization process. This is the case for the low-dimensional noisy Schaffer F6 (2D) function (f_1) and Sphere (5D) function (f_2), but also for the high-dimensional Griewank (50D) function (f_3): the best results are found for $s = 20$ or $s = 50$, as a compromise between the low and high ends.

On the contrary, for the Rosenbrock (50D) function (f_5), the results get better monotonically as s decreases, showing no deterioration due to a stronger

noise effect. For Rastrigin F1 (50D) function (f_4) the behaviour is similar, but there is still a residual "low end effect", which settles the best compromise result in $s = 5$. In these two cases, as outlined by Krink et al. [4], the main problem resides in the difficulty of the function, and not in the noise effect: the performance of any EA is affected by the intrinsic critical aspects of the function, well before the noise effect can come into play, in terms of fitness contribution. This can be also seen considering the results for the non-noisy cases: the results achieved for f_4 and especially for f_5 are sensibly far from the optimal value 0.

The comparison of MOGA-II results with those of DE and the generic EA presented in [4], shows the good performance of MOGA-II (see Table 4). For both low-dimensional noisy functions f_1 and f_2 , MOGA-II performs better than DE and EA, for all values of s.

As concerns f_3, where EA is better than DE, for low s, i.e. $s = 1$ and $s = 5$, MOGA-II is comparable to DE, while for high s, i.e. $s = 20$, 50, and 100 it is comparable to the good results of EA. With the f_4 function, for low s (i.e. $s = 1$, 5), where DE performs better than EA, MOGA-II results are better than EA but worse than DE; for high s (i.e. $s = 20$, 50, and 100), where EA is better than DE, the MOGA-II results are roughly comparable to those of EA. Finally, as concerns f_5, for low s DE performs better than EA, and conversely for high s EA is better than DE, as in the previous case. The results of MOGA-II are roughly situated in an intermediate position between the results of DE and EA, but the large standard deviations prevent us from a detailed comparative analysis. Such large standard deviations could be indicative of a difficult problem, but can also be due to the low number of repetitions. Unfortunately, as said in the previous section, we could repeat the experiments only 3 times, for intrinsic limitations.

It should be noted that in general MOGA-II converges to the optimal solution faster than DE, in terms of number of function evaluations: this is an important point in case of realistic applications, where the evaluation time for a single design can be very long.

6. Conclusion

In this paper, we have presented MOGA-II, a new evolutionary algorithm for multi-objective optimization, and tested it on single-objective optimization problems (with and without noise). Although compared to two very successful algorithms (differential evolution and a standard evolutionary algorithm), MOGA-II sometimes performed better and never worse than both algorithms, which were constructed for single-objective optimization.

The current results motivate further work in testing MOGA-II, especially to see its performance on real-world problems. We can conclude that MOGA-II

is a competent algorithm that is robust and efficient enough to handle different optimization problems, including noisy ones.

References

[1] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. In *Proceedings of the 6-th International Conference Parallel Problem Solving from Nature (PPSN-VI)*, 2000, pp. 849–858.

[2] C.M. Fonseca and P.J. Fleming. Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization. In *Proceedings of the Fifth International Conference on Genetic Algorithms*, San Mateo, USA, 1993, pp. 416–423.

[3] D.E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, Reading, Mass., USA, 1989.

[4] T. Krink, B. Filipič, G.B. Fogel, and R. Thomsen. Noisy optimization problems - A particular challenge for differential evolution? In *Proceedings of the 2004 IEEE Congress on Evolutionary Computation*, Portland, USA, 2004, pp. 332–339.

[5] C. Poloni and V. Pediroda. GA coupled with computationally expensive simulations: tools to improve efficiency. In *Genetic Algorithms and Evolution Strategies in Engineering and Computer Science*, John Wiley and Sons, England, 1997, pp. 267–288.

[6] S. Poles. MOGA-II An Improved Multi-Objective Genetic Algorithm. Technical report 2003-006, Esteco, Trieste, 2003.

[7] S. Poles. Bench-marking MOGA-II. Technical report 2004-001, Esteco, Trieste, 2004.

[8] K.V. Price and R. Storn. Differential evolution – a simple evolution strategy for fast optimization. *Dr. Dobb's Journal*, 22(4):18–24, 1997.

[9] K. Yamamoto and O. Inoue. New evolutionary direction operator for genetic algorithms. *AIAA Journal*, 33:1990–1993, 1995.

[10] E. Zitzler, M. Laumanns, and L. Thiele. SPEA2: Improving the Strength Pareto Evolutionary Algorithm. TIK-Report 103, Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH), Zürich, Switzerland, May 2001.

[11] Web page of the modeFRONTIER optimization software (http://www.esteco.com/Products/modeFrontier/index.html).

ANT ALGORITHM FOR THE MULTIDIMENSIONAL KNAPSACK PROBLEM

Inès Alaya
SOIE
Institut Supérieur de Gestion de Tunis, Tunisia
ines.alaya@isg.rnu.tn

Christine Solnon
LIRIS
University Lyon, France
christine.solnon@liris.cnrs.fr

Khaled Ghédira
SOIE
Institut Supérieur de Gestion de Tunis, Tunisia
khaled.ghedira@isg.rnu.tn

Abstract We propose a new algorithm based on the Ant Colony Optimization (ACO) meta-heuristic for the Multidimensional Knapsack Problem, the goal of which is to find a subset of objects that maximizes a given objective function while satisfying some resource constraints. We show that our new algorithm obtains better results than two other ACO algorithms on most instances.

Keywords: Ant colony optimization, Multidimensional knapsack problem

1. Introduction

The Multidimensional Knapsack Problem (MKP) is a NP-hard problem which has many practical applications, such as processor allocation in distributed systems, cargo loading, or capital budgeting. The goal of the MKP is to find a subset of objects that maximizes the total profit while satisfying some

resource constraints. More formally, a MKP is stated as follows:

$$\begin{array}{ll} \text{maximize} & \sum_{j=1}^{n} p_j \cdot x_j \\ \text{subject to} & \sum_{j=1}^{n} r_{ij} \cdot x_j \leq b_i, \forall i \in 1..m \\ & x_j \in \{0,1\}, \forall j \in 1..n \end{array}$$

where r_{ij} is the consumption of resource i for object j, b_i is the available quantity of resource i, p_j is the profit associated with object j, and x_j is the decision variable associated with object j and is set to 1 (resp. 0) if j is selected (resp. not selected).

In this paper, we describe a new algorithm for solving MKPs. This algorithm is based on Ant Colony Optimization (ACO) [4, 6, 5], a stochastic metaheuristic that has been applied to solve many combinatorial optimization problems such as traveling salesman problems [3], quadratic assignment problems [9], or vehicule routing problems [1]. The basic idea of ACO is to model the problem to solve as the search for a minimum cost path in a graph, and to use artificial ants to search for good paths. The behavior of artificial ants is inspired from real ants: they lay pheromone trails on components of the graph and they choose their path with respect to probabilities that depend on pheromone trails that have been previously laid; these pheromone trails progressively decrease by evaporation. Intuitively, this indirect stigmergetic communication mean aims at giving information about the quality of path components in order to attract ants, in the following iterations, towards the corresponding areas of the search space.

To solve MKPs with ACO, the key point is to decide which components of the constructed solutions should be rewarded, and how to exploit these rewards when constructing new solutions. A solution of a MKP is a set of selected objects $S = \{o_1, \ldots, o_k\}$ (we shall say that an object o_i is selected if the corresponding decision variable x_{o_i} has been set to 1). Given such a solution $S = \{o_1, \ldots, o_k\}$, one can consider three different ways of laying pheromone trails:

- A first possibility is to lay pheromone trails on each object selected in S. In this case, the idea is to increase the desirability of each object of S so that, when constructing a new solution, these objects will be more likely to be selected;
- A second possibility is to lay pheromone trails on each couple (o_i, o_{i+1}) of successively selected objects of S. In this case, the idea is to increase the desirability of choosing object o_{i+1} when the last selected object is o_i.
- A third possibility is to lay pheromone on all pairs (o_i, o_j) of different objects of S. In this case, the idea is to increase the desirability of

choosing together two objects of S so that, when constructing a new solution S', the objects of S will be more likely to be selected if S' already contains some objects of S. More precisely, the more S' will contain objects of S, the more the other objects of S will be attractive.

To solve MKP with ACO, Leguizamon and Michalewizc [8] have proposed an algorithm based on the first possibility, whereas Fidanova [7] has proposed another algorithm based on the second possibility. In this paper, we propose a new ACO algorithm for solving MKPs that is based on the third possibility. Our intuition is that this strategy should attract ants in a more precise way as the desirability of an object depends on the objects that already belong to the partial solution under construction.

2. Ant-Knapsack Description

We define the construction graph, on which ants lay pheromone trails, as a complete graph that associates a node to each object of the MKP. The quantity of pheromone laying on an edge (o_i, o_j) is denoted by $\tau(o_i, o_j)$. Intuitively, this quantity represents the learnt desirability of selecting together objects o_i and o_j.

Algorithm 1: Ant-knapsack

Initialize pheromone trails to τ_{max}
repeat
  for each ant k **in** $1..nbAnts$ **do**
    /* construct a solution $\mathcal{S}_k$ as follows: */
    Randomly choose a first object $o_1 \in 1..n$
    $\mathcal{S}_k \leftarrow \{o_1\}$
    $Candidates \leftarrow \{o_i \in 1..n / o_i$ can be selected without violating resource constraints$\}$
    while $Candidates \neq \emptyset$ **do**
      Choose an object $o_i \in Candidates$ with probability $p_{S_k}(o_i)$
      $\mathcal{S}_k \leftarrow \mathcal{S}_k \cup \{o_i\}$
      remove from $Candidates$ every object that violates some resource constraints
    end while
  end for
  Update pheromone trails w.r.t. $\{\mathcal{S}_1, \ldots, \mathcal{S}_{nbAnts}\}$
  if a pheromone trail is lower than τ_{min} **then**
    set it to τ_{min}
  end if
  if a pheromone trail is greater than τ_{max} **then**
    set it to τ_{max}
  end if
until maximum number of cycles reached **or** optimal solution found

The proposed ACO algorithm for solving MKPs is described by Algorithm 1 and more particularly follows the $\mathcal{MAX} - \mathcal{MIN}$ Ant System [10] : we ex-

plicitly impose lower and upper bounds τ_{min} and τ_{max} on pheromone trails (with $0 < \tau_{min} < \tau_{max}$), and pheromone trails are set to τ_{max} at the beginning of the search.

At each cycle of this algorithm, every ant constructs a solution. It first randomly chooses an initial object, and then iteratively adds objects that are chosen within a set *Candidates* that contains all the objects that can be selected without violating resource constraints. Once each ant has constructed a solution, pheromone trails are updated. The algorithm stops either when an ant has found an optimal solution (when the optimal bound is known), or when a maximum number of cycles has been performed.

2.1 Definition of Transition Probabilities

At each step of the construction of a solution, an ant k randomly selects the next object o_i within the set *Candidates* with respect to a probability $p_{S_k}(o_i)$. This probability is defined proportionally to a pheromone factor and a heuristic factor, i.e.,

$$p_{S_k}(o_i) = \frac{[\tau_{S_k}(o_i)]^\alpha \cdot [\eta_{S_k}(o_i)]^\beta}{\sum_{o_j \in Candidates} [\tau_{S_k}(o_j)]^\alpha \cdot [\eta_{S_k}(o_j)]^\beta}$$

where $\tau_{S_k}(o_i)$ is the pheromone factor of o_i, $\eta_{S_k}(o_i)$ is its heuristic factor, and α and β are two parameters that determine the relative importance of these two factors.

The pheromone factor $\tau_{S_k}(o_i)$ depends on the quantity of pheromone laid on edges connecting the objects that already are in the partial solution S_k and the candidate vertex o_i, i.e.,

$$\tau_{S_k}(o_i) = \sum_{o_j \in S_k} \tau(o_i, o_j)$$

Note that this pheromone factor can be computed in an incremental way: once the first object o_i has been randomly chosen, for each candidate object o_j, the pheromone factor $\tau_{S_k}(o_j)$ is initialized to $\tau(o_i, o_j)$; then, each time a new object o_l is added to the solution S_k, for each candidate object o_j, the pheromone factor $\tau_{S_k}(o_j)$ is incremented by $\tau(o_l, o_j)$.

The heuristic factor $\eta_{S_k}(o_i)$ also depends on the whole set S_k of selected objects. Let $c_{S_k}(i) = \sum_{g \in S_k} r_{ig}$ be the consumed quantity of the resource i when the ant k has selected the set of objects S_k. And let $d_{S_k}(i) = b_i - c_{S_k}(i)$ be the remaining capacity of the resource i. We define the following ratio:

$$h_{S_k}(j) = \sum_{i=1}^{m} \frac{r_{ij}}{d_{S_k}(i)}$$

which represents the tightness of the object j on the constraints i relatively to the constructed solution S_k. Thus, the lower this ratio is, the more the object is profitable.

We integrate the profit of the object in this ratio to obtain a pseudo-utility factor. We can now define the heuristic factor formula as follows:

$$\eta_{S_k}(j) = \frac{p_j}{h_{S_k}(j)}$$

2.2 Pheromone Updating

Once each ant has constructed a solution, pheromone trails laying on the construction graph edges are updated according to the ACO meta-heuristic. First, all amounts are decreased in order to simulate evaporation. This is done by multiplying the quantity of pheromone laying on each edge of the construction graph by a pheromone persistence rate $(1 - \rho)$ such that $0 \leq \rho \leq 1$.

Then, the best ant of the cycle deposits pheromone. More precisely, let $\mathcal{S}_k \in \{\mathcal{S}_1, \ldots, \mathcal{S}_{nbAnts}\}$ be the best solution (with maximal profit) constructed during the cycle, and $\mathcal{S}_{best}$ be the best solution built since the beginning of the run. The quantity of pheromone laid by ant k is inversely proportional to the gap of profit between $\mathcal{S}_k$ and $\mathcal{S}_{best}$, i.e., it is equal to $1/(1 + \mathrm{profit}(S_{\mathrm{best}}) - \mathrm{profit}(S_k))$. This quantity of pheromone is added on each edge connecting two different vertices of S_k.

3. Parameters Setting

When solving a combinatorial optimization problem with a heuristic approach such as evolutionary computation or ACO, one usually has to find a compromise between two dual goals. On one hand, one has to intensify the search around the most "promising" areas, that are usually close to the best solutions found so far. On the other hand, one has to diversify the search and favor exploration in order to discover new, and hopefully more successful, areas of the search space. The behavior of ants with respect to this intensification/diversification duality can be influenced by modifying parameter values. In particular, diversification can be emphasized either by decreasing the value of the pheromone factor weight α —so that ants become less sensitive to pheromone trails— or by decreasing the value of the pheromone evaporation rate ρ —so that pheromone evaporates more slowly. When increasing the exploratory ability of ants in this way, one usually finds better solutions, but as a counterpart it takes longer time to find them.

This is illustrated in Fig. 1 on a MKP instance with 100 objects and 5 resource constraints. When emphasizing pheromone guidance, by choosing values such as $\alpha = 2$ and $\rho = 0.02$, Ant-knapsack quickly finds good solutions but it may fail in finding the optimal (or the best) solution. On the contrary, when choosing

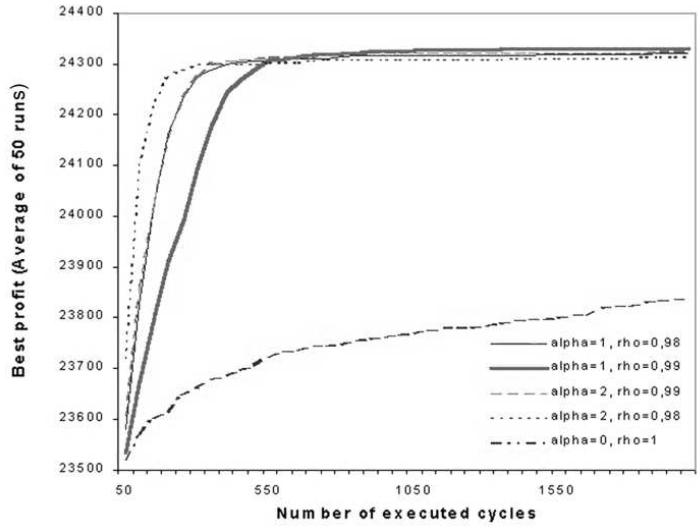


Figure 1. Influence of α and ρ on solution quality: each curve plots the evolution of the profit of the best solution when the number of cycles increases, for a given setting of α and ρ. The other parameters have been set to $\beta = 5$, $nbAnts = 30$, $\tau_{min} = 0.01$, and $\tau_{max} = 6$.

values for α and ρ that emphasize exploration, such as $\alpha = 1$ and $\rho = 0.01$, ants find better solutions, but they need more cycles to converge towards these solutions. A good compromise between solution quality and computation time is reached when α is set to 1 and ρ to 0.01.

For all experiments reported below, we have set α to 1, β to 5, ρ to 0.01, the number of ants nbAnts to 30, and the pheromone bounds τ_{min} and τ_{max} to 0.01 and 6. Finally, we limited the number of cycles to 2000.

4. Experiments and Results

The Ant-knapsack has been tested on benchmarks of MKP from OR-Library (available at `http://mscmga.ms.ic.ac.uk/`). We compare the results of Ant-knapsack with the two ACO algorithms of Leguizamon and Michalewicz [8] and Fidanova [7], and the genetic algorithm of Chu and Beasly [2].

Table 1 displays the results for 30 instances with 100 objects and 5 constraints ($n = 100$ and $m =$5). On these instances, Ant-knapsack clearly outperforms Fidanova's algorithm. It also obtains better results than the algorithm of Leguizamon and Michalewicz: the best solutions found are always larger or equal, and the average solutions found are larger for 7 instances, and smaller for 3 instances. Ant-knapsack finds the best known results of Chu and Beasley for 26 instances over the 30 tested instances.

Table 1. Results on 5.100 instances. For each instance, the table reports the best solutions found by Chu and Beasley as reported in [2](C. & B.), the best and average solutions found by Leguizamon and Michalewicz as reported in [8](L. & M.), and the best solutions found by Fidanova as reported in [7]. It then reports results obtained by Ant-knapsack: best and average solutions over 50 runs, followed by standard deviation in brackets, and the average number of cycles needed to find the best solution (C*).

N°	*C. & B.*	*L. & M.*		*Fidanova*	*Ant-knapsack*		
	Best	*Best*	*Avg*	*Best*	*Best*	*Avg (sdv)*	*C**
00	24381	24381	24331	23984	24381	24342 (29.3)	522
01	24274	24274	24245	24145	24274	24247 (38.5)	469
02	23551	23551	23527	23523	23551	23529 (8.0)	483
03	23534	23527	23463	22874	23534	23462 (32.6)	500
04	23991	23991	23949	23751	23991	23946 (31.8)	589
05	24613	24613	24563	24601	24613	24587 (31.3)	535
06	25591	25591	25504	25293	25591	25512 (43.8)	480
07	23410	23410	23361	23204	23410	23371 (30.3)	509
08	24216	24204	24173	23762	24216	24172 (32.9)	571
09	24411	24411	24326	24255	24411	24356 (44.3)	588
10	42757			42705	42757	42704 (14.3)	537
11	42545			42445	42510	42456 (15.8)	577
12	41968			41581	41967	41934 (22.3)	635
13	45090			44911	45071	45056 (24.0)	627
14	42218			42025	42218	42194 (33.2)	512
15	42927			42671	42927	42911 (33.3)	484
16	42009			41776	42009	41977 (45.2)	458
17	45020			44671	45010	44971 (32.5)	490
18	43441			43122	43441	43356 (38.5)	514
19	44554			44471	44554	44506 (25.2)	517
20	59822			59798	59822	59821 (3.2)	261
21	62081			61821	62081	62010 (47.1)	387
22	59802			59694	59802	59759 (21.7)	450
23	60479			60479	60479	60428 (21.8)	368
24	61091			60954	61091	61072 (20.0)	298
25	58959			58695	58959	58945 (14.5)	356
26	61538			61406	61538	61514 (24.0)	407
27	61520			61520	61520	61492 (25.6)	396
28	59453			59121	59453	59436 (40.5)	395
29	59965			59864	59965	59958 (8.4)	393

Table 2 displays the results for 30 instances with 100 objects and 10 constraints ($n = 100$ and $m = 10$). On these instances, Ant-knapsack also obtains better results than the algorithm of Leguizamon and Michalewicz: the best solutions found are larger or equal for 9 instances over 10, and the average solutions found are larger for 8 instances, and smaller for 2 instances. Ant-knapsack finds for this set also the best known results of Chu and Beasley for 25 instances over 30.

We also tested Ant-knapsack on larger MKP instances with 500 objects and 5 constraints (Table 3). The best known results for this set are obtained by

Table 2. Results on 10.100 instances. For each instance, the table reports the best solutions found by Chu and Beasley as reported in [2](C. & B.), and by Leguizamon and Michalewicz as reported in [8](L. & M.). It then reports results obtained by Ant-knapsack: best and average solutions over 50 runs, followed by standard deviation in brackets, and the average number of cycles needed to find the best solution (C*).

N°	C. & B.	L. & M.		Ant-knapsack		
	Best	Best	Avg	Best	Avg (sdv)	C*
00	23064	23057	22996	23064	23016 (42.2)	538
01	22801	22801	22672	22801	22714 (67.2)	575
02	22131	22131	21980	22131	22034 (66.9)	598
03	22772	22772	22631	22717	22634 (60.6)	700
04	22751	22654	22578	22654	22547 (66.3)	640
05	22777	22652	22565	22716	22602 (63.3)	645
06	21875	21875	21758	21875	21777 (44.9)	552
07	22635	22551	22519	22551	22453 (89.2)	586
08	22511	22418	22292	22511	22351 (69.4)	534
09	22702	22702	22588	22702	22591 (88.5)	588
10	41395			41395	41329 (48.5)	501
11	42344			42344	42214 (49.5)	559
12	42401			42401	42300 (58.1)	584
13	45624			45624	45461 (73.6)	562
14	41884			41884	41739 (57.3)	536
15	42995			42995	42909 (76.3)	525
16	43559			43553	43464 (71.7)	597
17	42970			42970	42903 (47.7)	439
18	42212			42212	42146 (48.0)	598
19	41207			41207	41067 (89.7)	548
20	57375			57375	57318 (59.5)	330
21	58978			58978	58889 (40.2)	504
22	58391			58391	58333 (29.5)	513
23	61966			61966	61885 (42.4)	427
24	60803			60803	60798 (5.0)	316
25	61437			61437	61293 (52.7)	502
26	56377			56377	56324 (35.7)	453
27	59391			59391	59339 (53.3)	445
28	60205			60205	60146 (62.6)	360
29	60633			60633	60605 (36.1)	360

Vasquez and Hao [11]. They proposed an hybrid algorithm that combines tabu search and linear programming. On these difficult instances, we find worse results than those of Vasquez and Hao.

5. Conclusion

In this paper, we propose an ACO algorithm for the multidimensional knapsack problem. This algorithm differs from many ACO algorithms in the fact that pheromone trails are laid not only on the edges of the visited paths, but on all edges connecting any pair of nodes belonging to the solution. In addition, when adding a node to the solution under construction, the probability of choos-

Table 3. Results on 5.500 instances. For each instance, the table reports the best solutions found by Vasquez and Hao as reported in [11](V. & H.). It then reports results obtained by Ant-knapsack: best and average solutions over 50 runs, followed by standard deviation in brackets, and the average number of cycles needed to find the best solution (C*).

N°	*V. & H.*	*Ant-knapsack*		
	Best	*Best*	*Avg (sdv)*	*C**
00	120134	119893	119658 (135.8)	1625
01	117864	117604	117423 (130.4)	120
02	121112	120846	120622 (121.4)	1600
03	120804	120534	120279 (152.3)	124
04	122319	122126	121829 (135.2)	191

ing a node not only depends on the pheromone trail between the last visited node and the candidate node but on the trails laying on all edges connecting the candidate node and all visited nodes in the solution. The proposed algorithm finds most of the best known results for the tested MKP benchmarks. This algorithm improves also many results found by other ACO algorithms.

References

[1] B. Bullnheimer, R.F. Hartl, and C. Strauss. Applying the Ant System to the vehicle routing problem. In S. Voß, S. Martello, I.H. Osman, and C. Roucairol (eds.): *Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization*, Kluwer Academic Publishers, Dordrecht, 1999, pp. 285–296.

[2] P.C. Chu and J.E. Beasley. A genetic algorithm for the multidimentional knapsack problem. *Journal of Heuristics*, 4:63–86, 1998.

[3] M. Dorigo, A. Colorni, and V. Maniezzo. The Ant System: Optimization By a Colony of Cooperating Agents. *IEEE Transactions on Systems, Man, and Cybernetics-Part B*, 26:1–13, 1996.

[4] M. Dorigo and G. Di Caro. The Ant Colony Optimization Meta-Heuristic. In D. Corne, M. Dorigo and F. Glover (eds.): *New Ideas in Optimization*, McGraw Hill, London, 1999, pp. 11–32.

[5] M. Dorigo, G. Di Caro, and L.M. Gambardella. Ant Algorithms for Discrete Optimization. *Artificial Life*, 5:137–172, 1999.

[6] M. Dorigo and T. Stützle. *Ant Colony Optimization*. MIT Press, Cambridge, MA, 2004.

[7] S. Fidanova. Evolutionary Algorithm for Multidimensional Knapsack Problem. In *Proceedings of the Seventh International Conference on Parallel Problem Solving from Nature (PPSN VII), Workshop on Real World Optimization Using Evolutionary Computing*, Granada, Spain, 2002.

[8] G. Leguizamon and Z. Michalewicz. A new version of Ant System for Subset Problem. In *Proceedings of the Congress on Evolutionary Computation*, Washington, 1999, pp. 1459-1464.

[9] T. Stützle and M. Dorigo. ACO Algorithms for the Quadratic Assignment Problem. In D. Corne, M. Dorigo, and F. Glover (eds.): *New Ideas in Optimization*, McGraw Hill, London, 1999, pp. 33–50.

[10] T. Stützle and H.H. Hoos. MAX-MIN Ant System. *Future Generation Computer Systems*, 16:889–914, 2000.

[11] M. Vasquez and J.K. Hao. A Hybrid Approach for the 0-1 Multidimensional Knapsack Problem. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI-01)*, Washington, 2001.

MULTILEVEL OPTIMIZATION OF GRAPH BISECTION WITH PHEROMONES

Peter Korošec
Computer Systems Department
Jožef Stefan Institute, Ljubljana, Slovenia
peter.korosec@ijs.si

Jurij Šilc
Computer Systems Department
Jožef Stefan Institute, Ljubljana, Slovenia
jurij.silc@ijs.si

Abstract We present a multiple ant-colony algorithm (MACA) for the graph bisection problem. The aim of this paper is to compare the performance of the MACA with results on the benchmark graphs from Graph partitioning Archive at the University of Greenwich. Experimental results show that the MACA is comparable with the state-of-the-art graph bisection algorithms.

Keywords: Ant colony algorithm, Multilevel optimization, Graph bisection, Performance evaluation

1. Introduction

Let $G = (V, E)$ be an undirected and unweighted graph with $|V| = n$. Generalizing the standard definition for odd n, we define:

- A *bisection* is a partition (D_1, D_2) of V with $|D_1| = \lceil \frac{n}{2} \rceil$.
- The *bisection width* ω is defined as the minimum number of edges between domains D_1 and D_2 among all possible bisections (D_1, D_2).
- MINBISECTION is the NP-hard problem of finding a bisection with a minimum bisection width ω.

The graph bisection is an elementary and important problem in graph theory and have many real world applications, such as parallel scientific computing and VLSI design.

For example, in scientific computing, it is common to use parallel computers to perform sparse matrix-vector multiplication. Typically each processor owns some fraction of the rows of the matrix, and is responsible for computing only those components of the result corresponding to rows it owns. In order to compute the result, however, it must have a valid entry in any component of the vector for which there is a nonzero entry in the corresponding column of the matrix in any row it owns. Thus the amount of communication which is necessary to perform a matrix-vector multiplication in parallel depends on how effectively the rows of the matrix are distributed to the processors. If there are two processors, it is desirable to split the number of rows that each processor owns roughly in half (for load balancing), and to assign rows so that the number of nonzero matrix entries a_{ij} with i owned by one processor and j owned by the other is minimized (to minimize communication). If there are more than two processors, one typically uses graph bisection recursively until a good assignment of rows to processors is found.

Another application of graph bisection is found in VLSI placement. In designing VLSI layouts, the divide-and-conquer approach is often utilized. Typically, the circuit is split in half by removing wires connect the halves. Each half is recursively laid out and then the wires connecting the two halves are put back. The quality of the final layout depends greatly on the number of wires that are removed. The problem of minimizing the number of wires that go between the two halves is equivalent to the graph bisection problem if one consider the components to be the vertices in a graph and the wires to be edges. By producing a bisection of the graph with a minimal cut size, we minimize the number of wires between the two halves.

Since the space of feasible solutions for the MINBISECTION problem is prohibitively large we are forced to recourse to heuristic approaches, which reduce or bound the space to be searched. A promising alternative is to use stochastic heuristics, some of which are based on various fundamental principles observed in nature. Recently, a number of studies have shown that such techniques have great potential for solving the MINBISECTION problem. Examples include simulated annealing [5], genetic algorithms [2, 6, 11], and ant-colony algorithms [3, 8, 10].

2. Multiple Ant Colony Approach

The basic idea of the multiple ant-colony algorithm **MACA** is very simple [7]. We have two or more colonies of ants that are competing for food. In our case food are the vertices of the graph.

First we map the graph onto the grid, which represents the ants' habitat (a place where the ants can move). There are many possibilities as to how a graph can be mapped, but, for our example, we will consider a random mapping.

Ants are placed into their nest locus from where they start their foraging and gathering of food.

Algorithm 1: MACA

```
initialize();
while ending condition not satisfied do
  for all ants of colony do
    for all colonies do
      if carrying food then
        if in nest locus then
          drop_food()
        else
          move_to_nest()
        end if
      else if food here then
        pick_up_food()
      else if food ahead then
        move_forward()
      else if in nest locus then
        move_to_away_pheromone()
      else if help signal then
        move_to_help()
      else
        follow_strongest_forward_pheromone()
      end if
    end for
  end for
  for all grid cells do
    evaporate_pheromone()
  end for
end while
End pseudo-code.
```

The ants on the grid move in three possible directions (forward, left and right). The decision in which direction an ant will move is defined by the probability of movement. A cumulative probability distribution is used to decide which direction is chosen. When an ant tries to move off the grid, it is forced to move left or right with equal probability. When an ant finds food it tries to pick it up. At first it checks if the quantity of the temporarily gathered food in its nest is not on the limit (the capacity of storage is limited due to the problem constraints). If the limit is not reached, then the weight of the food is calculated from the number of cut edges created by assigning the selected vertex to the partition associated with the nest of the current ant, otherwise the ant moves in a randomly selected direction. If the weight of the food is too heavy for one ant to pick it up (and not too heavy for a few ants to lift it up) then an ant sends a

help signal within the radius of a few cells. So if ants are in the neighborhood, they will help this ant to carry the food to the nest locus. On the way back to the nest locus an ant deposits pheromones on the trail that it is making, so the other ants can follow its trail and gather more food from that, or a nearby, cell. When an ant reaches the nest locus it drops the food in the first possible place around the nest (in a clockwise direction). After an ant drops its food it starts a new round of foraging. Of course ants can also gather food from other nests. When an ant tries to pick up a food from the other nests it a performs the same procedure as if it was gathering for food, except when the food is to heavy to pick it up, it does not send a help signal but moves on. With this, we significantly improve our temporary solution.

3. Multilevel Optimization

An effective way to speed up and globally improve any partitioning method is the use of multilevel techniques [1]. The basic idea is to group vertices together to form clusters that define a new graph. This procedure is applied until the graph size becomes small enough. Each step is followed by a successive refinement of the graph.

The implementation of these ideas consists of two parts: graph contraction and partition expansion. In *graph contraction* a coarser graph $G_{\ell+1}(V_{\ell+1}, E_{\ell+1})$ is created from $G_\ell(V_\ell, E_\ell)$ by finding the largest independent subset of graph edges and then collapsing them. Each selected edge is collapsed and the vertices $u_1, u_2 \in V_\ell$ that are at either end of it are merged into a new vertex $u \in V_{\ell+1}$ with weight $|v| = |u_1| + |u_2|$. Edges that have not been collapsed are inherited by the new graph $G_{\ell+1}$, and the edges that become duplicated are merged and their weight is summed. Because of inheritance the total weight of the graph remains the same and the total edge weight is reduced by an amount equal to the weight of the collapsed edges, which has no impact on the graph imbalance or the edge-cut. In the second part, *graph expansion with partitioning*, an already optimized partition of graph G_ℓ is expanded. The optimized partition must be interpolated onto its parent graph $G_{\ell-1}$. Because of the simplicity of coarsening in the first part, the interpolation itself is very trivial. So, if vertex $v \in V_\ell$ belongs to subdomain D_i, then after refinement the matched pair $u_1, u_2 \in V_{\ell-1}$ that represents v will also be in D_i. The graph is expanded to its original size and a partitioning algorithm is run on every level ℓ of the expansion.

Due to large graphs and an increased number of levels, the number of vertices in a single cell increases rapidly. For this reason we suggested and applied a *bucket sort* procedure that accelerates and improves the algorithm's convergence by choosing the most "promising" vertex from the cell. The basic idea of the bucket sort procedure is that all the vertices of a given gain g are put together in a "bucket" ranked g. The problem of finding a vertex with the maximum gain is

then reduced to finding the non-empty bucket with the highest rank, and picking a vertex from it. If a chosen vertex migrates from one subdomain to another, then only its gain and the gains of all its neighbors have to be recalculated and put back into appropriate buckets. In our implementation each bucket is represented by a double-linked list of vertices. Because of the multilevel process, it often happens that the potential gain values are dispersed over a wide range. For this reason we introduced a 2-3 tree, and so eliminated large and sparse arrays of pointers. The non-empty buckets are stored in the 2-3 tree, so each leaf in the tree represents a bucket. For even faster searching we made one 2-3 tree for each colony on every cell that has vertices on it. With this we speeded up the search, as well as the add and delete operations.

4. Performance Evaluation

The MACA was implemented in Borland® Delphi™. The experiments were made on a computer with an AMD Athlon™XP 1800+ processor running the Microsoft® Windows® XP operating system. The implementation also includes a visualization tool to assist the user in selecting the appropriate parameters of the algorithm (see Fig. 1).

The benchmark graphs used in our experiment were taken from the Graph Partitioning Archive and are described in Table 1.

Table 1. Benchmark suite from the Graph Partitioning Archive at the University of Greenwich.*

Graph	*Number of nodes*	*Number of edges*
add20	2 395	7 462
data	2 851	15 093
3elt	4 720	13 722
uk	4 824	6 837
add32	4 960	9 462
bcsstk33	8 738	291 583
whitaker3	9 800	28 989
crack	10 240	30 380
wing_nodal	10 937	75 488
fe_4elt2	11 143	32 818
4elt	15 606	45 878
fe_sphere	16 386	49 152
cti	16 840	48 232
cs4	22 499	43 858

*http://staffweb.cms.gre.ac.uk/~c.walshaw/partition/

The results of our experiment are shown in Table 2. It is clear that the MACA performed very well. Notice that our MACA is superior to the classical k-METIS and Chaco algorithms [8]. The MACA also returned some solutions that are better than currently available solutions (Table 2). Furthermore, the

MACA is even comparable to the combined evolutionary/multilevel scheme used in the JOSTLE Evolutionary algorithm [11], which is currently the most promising partitioning algorithm.

Table 2. The best bisections found to date.*

Graph	*Bisection width* ω		*Algorithm*
	Produced with MACA	*The best known*	
add20	603	612	MQI
data	199	191	mpM4.0
3elt	90	90	JE
uk	20	20	mpM4.0
add32	11	11	Ch2.0
bcsstk33	10 224	10 172	mpM4.0
whitaker3	127	127	JE
crack	184	184	JE
wing_nodal	1 709	1 707	JE
fe_4elt2	130	130	MRSB
4elt	139	139	JE
fe_sphere	402	386	JE
cti	334	334	JE
cs4	391	372	JE

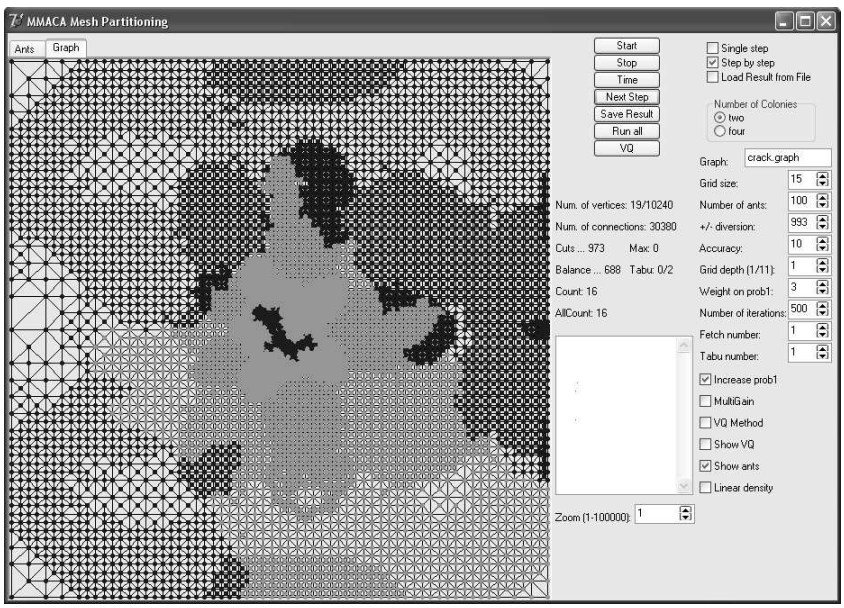
*Graph Partitioning Archive (Summer 2004)

- MQI – Max-flow Quotient-cut Improvement, a bisection algorithm from Lang and Rao, which uses many multiple tries and improves an initial partition provided by METIS [9]
- mpM4.0 – Multiple runs of a randomized version of p-Metis; results provided by Lang and Rao.
- JE – JOSTLE Evolutionary - combined evolutionary/multilevel scheme [11]
- Ch2.0 – CHACO - multilevel Kernighan-Lin (recursive bisection); version 2.0 (October 1995) [4]
- MRSB – Barnard and Simon's Multilevel Recursive Spectral Bisection [1]

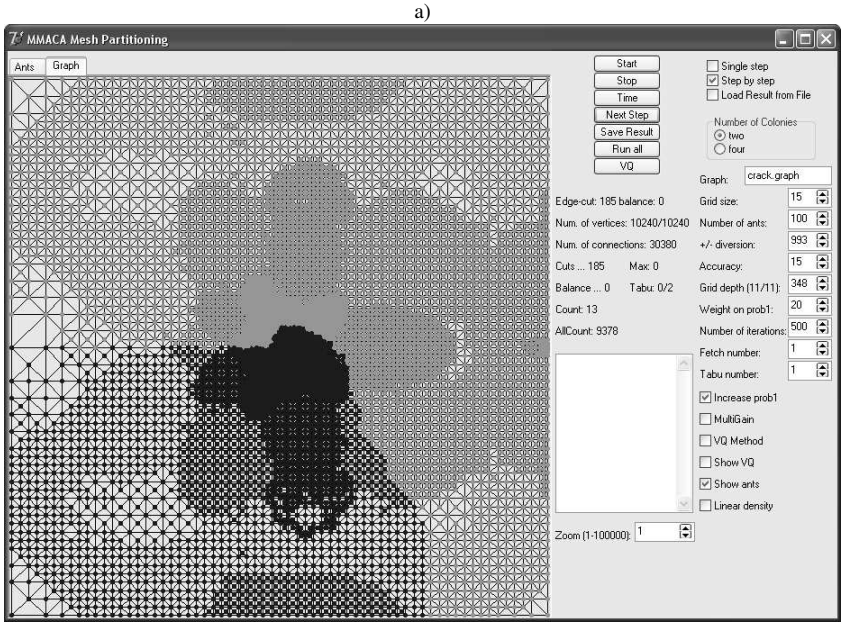
5. Conclusions

This paper introduce the graph bisection problem as well as the ant-colony optimization technique. A multilevel multiple ant-colony algorithm (MACA) was developed for solving graph bisection problem. The results achieved were close to the best known results for the set of benchmark graphs.

There is a wide range of possibilities to be considered in the future. One of the most appealing is a merger of the MACA with some other method through daemon actions and parallel implementation of the MACA. This could make the algorithm even more competitive when compared with other heuristics algorithms.



a)



b)

Figure 1. Bisection of the graph `crack` with the MACA: a) after 16 steps ($\omega = 973$); b) final solution ($\omega = 185$).

Acknowledgment

The work presented in the paper was supported by the Slovenian Ministry of Education, Science and Sport (research programme P2-0098 *Computer Structures and Systems*).

References

[1] S.T. Barnard and H.D. Simon. A fast multilevel implementation of recursive spectral bisection for partitioning unstructured problems. *Concurrency: Practice and Experience*, 6:101–117, 1994.

[2] T.N. Bui and B.R. Moon. Genetic algorithm and graph partitioning. *IEEE Transactions on Computers*, 45:841–855, 1996.

[3] T.N. Bui and L.C. Strite. An ant system algorithm for graph bisection. In *Proceedings of the Genetic and Evolutionary Computation Conference*, New York, USA, 2002, pp. 43–51.

[4] B. Hendrickson and R. Leland. A Multilevel Algorithm for Partitioning Graphs. In *Proceedings of the Supercomputing '95*, San Diego, USA, 1995.

[5] M. Jerrum and G.B. Sorkin. The Metropolis algorithm for graph bisection. *Discrete Applied Mathematics*, 82:155–175, 1998.

[6] K. Kohmoto, K. Katayama, and H. Narihisa. Performance of a genetic algorithm for the graph partitioning problem. *Mathematical and Computer Modelling*, 38:1325–1332, 2003.

[7] P. Korošec. Metaheuristic solving of the optimization problem with ant colonies. M.Sc. Thesis, Faculty of Computer and Information Science, University of Ljubljana, Slovenia, 2004.

[8] P. Korošec, J. Šilc, and B. Robič. Solving the mesh-partitioning problem with an ant-colony algorithm. *Parallel Computing*, 30:785–801, 2004.

[9] K. Lang and S. Rao. A flow-based method for improving the expansion or conductance of graph cuts. In *IPCO Summer School*, New York, USA, 2004.

[10] A.E. Langham and P.W. Grant. Using competing ant colonies to solve k-way partitioning problems with foraging and raiding strategies. *Lecture Notes in Computer Science*, 1674:621–625, 1999.

[11] A.J. Soper, C. Walshaw, and M. Cross. A combined evolutionary search and multilevel optimisation approach to graph partitioning. In *Proceedings of the Genetic and Evolutionary Computation Conference*, Las Vegas, USA, 2000, pp. 674–681.

III

APPLICATIONS

IN SEARCH FOR AN EFFICIENT PARAMETER TUNING METHOD FOR STEEL CASTING

Tea Robič
Department of Intelligent Systems
Jožef Stefan Institute, Ljubljana, Slovenia
tea.robic@ijs.si

Bogdan Filipič
Department of Intelligent Systems
Jožef Stefan Institute, Ljubljana, Slovenia
bogdan.filipic@ijs.si

Abstract We deal with the optimization of process parameters in industrial continuous casting of steel. The process requires fine-tuning of numerous parameters with respect to the metallurgical cooling criteria to achieve the highest possible quality of the cast steel. We tackle the problem with various optimization methods: local optimization, conjugate gradient, downhill simplex and several types of evolutionary algorithms. They search the parameter space and evaluate candidate settings using a numerical simulator of the process and a cost function defined over the metallurgical criteria. We analyze the performance of the methods with respect to effectiveness and efficiency, and compare the optimized parameter settings with the manual setting used previously at the steel plant. The best results are achieved by local optimization and conjugate gradient what suggests that the applied cost function is not complex.

Keywords: Continuous casting of steel, Process simulator, Numerical optimization methods, Local optimization, Conjugate gradient, Downhill simplex, Evolutionary algorithms, Differential evolution, Comparative study

1. Introduction

Continuous casting of steel is broadly used at modern steel plants to produce steel semi-manufactures. The quality of the cast steel is subject to many process parameters, such as the casting temperature and speed, coolant temperatures and flows, etc. The number of possible parameter settings grows exponentially with the number of parameters considered. Consecutively, finding the parameter

setting that will result in high-quality steel is a demanding task. Because of high cost and safety risks, the optimization cannot be carried out on a real caster and we therefore use a numerical simulator of the casting process. However, the evaluation of solutions on the simulator is a time-consuming process. For this reason, we need to find the best parameter setting in as few process simulations as possible. In other words, we search for an optimization method that is effective and efficient.

Over the last years, several advanced computer techniques have been used in attempts to enhance the process performance and product properties in continuous casting of steel. Cheung and Garcia [3] combined a numerical model of the process with a heuristic search technique to find parameter values that reduce the proportion of defects in steel production. Filipič and Šarler [4] optimized 14 process parameters of an industrial steel casting machine with an automated software environment consisting of a numerical process simulator and evolutionary algorithm (EA). Chakraborti and coworkers [1] found genetic algorithms to be the most suitable technique for optimizing the settings of the continuous casting mold. In a follow-up study [2] based on heat transfer modeling, they used genetic algorithms to determine the maximum casting speed.

An important question in tuning the casting process parameters is which optimization algorithm to use. In this paper, we extend the collection of applied methods from generational EA and steady-state EA, to differential evolution and other optimization methods (local optimization, conjugate gradient and downhill simplex). By investigating a variety of approaches, we hope to enhance the knowledge on suitability of the optimization methods for this problem.

In Sect. 2 we outline the process of continuous casting of steel, present the simulation-based optimization procedure, and give an example of the optimization problem. The applied methods are described in Sect. 3. The numerical experiments and results are presented in Sect. 4 and discussed in Sect. 5. The paper concludes with a summary of the work done and directions for further investigation.

2. Optimization of Continuous Casting of Steel

2.1 The Process

The process of continuous casting of steel (schematically shown in Fig. 1) is a complex metallurgical process where molten steel is cooled and shaped into semi-manufactures of desired dimensions. The main components of the casting system are the ladle, tundish, mold and cooling subsystems. The ladle is used to transfer batches of molten steel from a steel-making furnace into the tundish. The tundish holds steel while casting is carried out. It ensures the continuity of steel flow into the mold. The mold is the heart of the casting system. It extracts heat from the molten steel and initiates the formation of a solid shell on

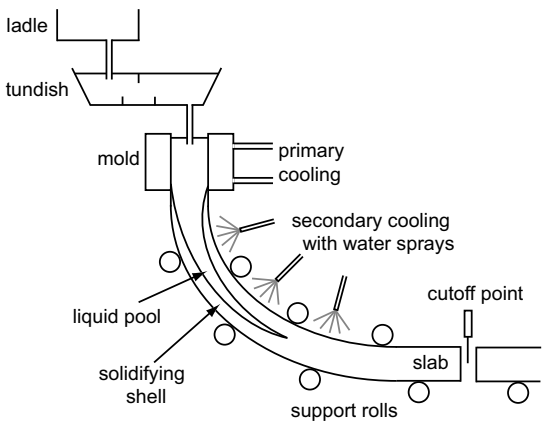


Figure 1. Schematic view of the continuous casting of steel.

the slab coming out of the mold. The mold oscillates to prevent the steel from sticking to the copper-alloy plates of the mold. Heat extraction is performed by coolant flowing through channels built in the mold. This represents the primary cooling subsystem of the caster. The heat extraction and solidification continue as the slab, led by support rolls, passes through the caster. Along the moving slab water sprays are located which form the secondary cooling subsystem. Cooling in this region results in complete solidification and the solidified slab is finally cut into pieces of the ordered length.

2.2 Simulation-Based Optimization Procedure

To tune the process parameters in continuous casting of steel, we have implemented an optimization environment consisting of the process simulator [12], a cost function and various optimization algorithms. Initially, given the process parameter values, the simulator computes the temperature field in the slab and extracts the metallurgical criteria of critical importance for the steel quality. Afterwards, the cost function value is calculated from the obtained criteria. The cost value is used by the applied optimization algorithm to generate new parameter setting and send it to the simulator. This represents one step in the simulation-based optimization procedure that operates in this manner until the stopping criterion is met.

In this study, five of the metallurgical criteria c_i provided by the simulator [12] are considered: maximum cooling and reheating rates on the slab surface in the secondary cooling zone, maximum surface temperature in the slab unbending point, and maximum negative and positive temperature deviations on the slab

surface. They are taken into account in the cost function f as

$$f = \sum_{i=1}^{5} \frac{c_i - c_i^{\min}}{c_i^{\max} - c_i^{\min}}, \tag{1}$$

where $c_i^{\min}$ and $c_i^{\max}$ are the lower and upper bounds for the i-th criterion. The bounds have been determined empirically. The criteria are defined in such a way that a lower value indicates a more satisfied criterion. Thus the optimization task is to find a parameter setting that will result in the minimum value of the cost function f.

As the evaluation of parameter settings with the simulator is time-consuming, a database of the evaluated solutions is maintained. When a solution is to be evaluated, the optimization algorithm first checks for the presence of the solution in the database and activates the simulator only if the solution has not yet been evaluated.

2.3 An Example of the Optimization Problem

Let us consider an example of the optimization problem, where 12 spray coolant flows in the secondary cooling zone are subject to optimization. Table 1 shows the parameter search space for this problem. The total number of possible parameter settings equals to $5^{12} \approx 2.4 \cdot 10^8$. The task is to find the parameter setting $x^* = (x_1, \ldots, x_{12})$ that minimizes the cost function f.

Table 1. An example of the optimization problem: the search space.

Coolant flow number	*Min. value* [l/min]	*Max. value* [l/min]	*Discretization step* [l/min]	*Number of values*
1	120	160	10	5
2	65	85	5	5
3	200	280	20	5
4	190	270	20	5
5	160	240	20	5
6	150	230	20	5
7	120	160	10	5
8	140	180	10	5
9	120	160	10	5
10	120	160	10	5
11	130	170	10	5
12	120	160	10	5

3. Optimization Methods

For the optimization of continuous casting of steel, six optimization methods were tested. They include single-point iterative procedures and population-

based algorithms, gradient and evolutionary methods, as well as stochastic and deterministic approaches. All methods use real vector representation of candidate solutions $x = (x_1, \ldots, x_n)$, where n is the number of parameters to be optimized. The search space is discretized. The stopping criterion is the predefined number of the examined solutions. The methods are described in the following subsections.

3.1 Local Optimization

Local optimization (LO) is a simple optimization method that searches for an optimum by examining the neighborhood of the current solution. The points $x = (x_1, \ldots, x_n)$ and $y = (y_1, \ldots, y_n)$ are defined to be neighbors if, according to the given discretization, they differ by one step in exactly one dimension:

$$x_k = y_k \pm d_k \text{ for an arbitrary } k \in \{1, \ldots, n\}, \text{ and } x_i = y_i \text{ for all } i \neq k. \quad (2)$$

Therefore, every point of the n-dimensional search space (except for the border points) has $2n$ neighbors. The LO procedure is presented in more detail in Algorithm 1.

Algorithm 1: Local optimization (LO)

```
randomly select an initial solution x
initiate direction d := 0
while stopping criterion not met do
  for i := 1 to num_neighbors do
    if d = 0 then
      randomly select neighbor y from the neighborhood of x
    else
      get neighbor in the given direction y := x + d
    end if
    if y is better than x then
      remember direction d := y − x
      replace solution x := y
      break
    else
      reset direction d := 0
    end if
  end for
  if stuck in a local optimum then
    randomly select a new solution x
  end if
end while
```

3.2 Conjugate Gradient

The conjugate gradient method (CG) is used for minimizing functions f, for which the gradients f' and f'' can be computed. The method consists of iterative steps in which the search of the space is made in conjugate directions.

Two vectors (or directions) d_i and d_j are conjugate with regard to the symmetric positive definite matrix A if

$$d_i^T A\, d_j = 0. \tag{3}$$

In an n-dimensional space, there are n conjugate directions. In CG the directions are conjugate with regard to the Hessian matrix f'' (that can be approximated by its diagonal). The CG method is outlined in Algorithm 2. Further details on the method can be found in [11].

Algorithm 2: Conjugate Gradient (CG)

randomly select a starting point $x_{(0)}$
calculate the preconditioner M (the diagonal of the Hessian matrix f'')
compute the first search direction as $d_{(0)} = -M^{-1} f'(x_{(0)})$
while stopping criterion not met **do**
  for $i := 0$ **to** $n - 1$ **do**
    with the Newton-Raphson method find $\alpha_{(i)}$ that minimizes $f(x_{(i)} + \alpha_{(i)} d_{(i)})$
    the next point is $x_{(i+1)} = x_{(i)} + \alpha_{(i)} d_{(i)}$
    the next search direction is $d_{(i+1)} = -M^{-1} f'(x_{(i+1)}) + \beta_{(i+1)} d_{(i)}$
    (the coefficient $\beta_{(i+1)}$ is calculated with the Polak-Ribiere method)
  end for
  restart from the best point found in the direction $d_{(0)} = -M^{-1} f'(x_{(n)})$
end while

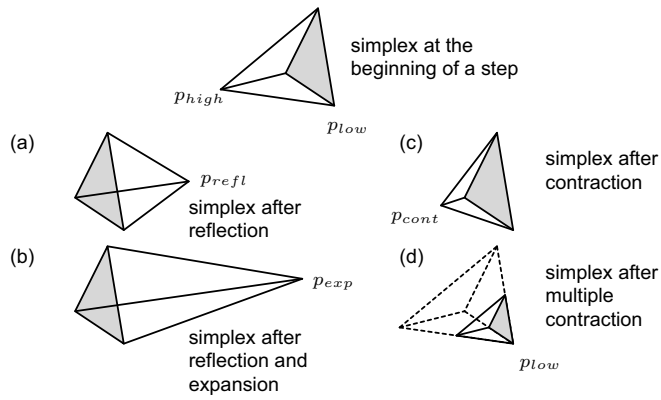


Figure 2. Simplex at the beginning of a step and possible outcomes for a step of DS [8].

3.3 Downhill Simplex

The downhill simplex method (DS), also named the Nelder-Mead method after its authors [7], searches for a minimum of an n-dimensional function by making use of a simplex. A simplex is a geometrical figure consisting, in n dimensions, of $n + 1$ vertices and all their interconnecting line segments, polygonal faces, etc.

DS starts with a randomly chosen simplex and takes a series of steps (reflections, expansions and contractions) which transform the simplex and move it towards the minimum. At every step the point of the simplex with the highest function value (p_{high} – "highest point") is transformed into a lower point (see Fig. 2 and Algorithm 3).

Algorithm 3: Downhill Simplex (DS)

randomly choose the vertices of an initial simplex
while stopping criterion not met **do**
  if simplex too small **then**
    construct a new simplex with p_{low} and n random points
  end if
  reflect p_{high} through the opposite face of the simplex (Fig. 2(a))
  if p_{refl} is lower than p_{low} **then**
    expand the simplex in the same direction (Fig. 2(b))
    if p_{exp} is lower than p_{refl} **then**
      replace p_{high} with p_{exp}
    else
      replace p_{high} with p_{refl}
    end if
  else if p_{refl} is lower than p_{high} **then**
    replace p_{high} with p_{refl}
  else
    contract the simplex (Fig. 2(c))
    if p_{cont} is lower than p_{high} **then**
      replace p_{high} with p_{cont}
    else
      contract the simplex around p_{low} (Fig. 2(d))
    end if
  end if
end while

3.4 Generational Evolutionary Algorithm

The generational evolutionary algorithm (GEA) is an optimization method that imitates the principles of Darwinian theory of evolution. By applying selection, crossover and mutation to a population of solutions, it creates better

and better offspring populations (Algorithm 4). This method was originally studied in [6] and made popular by [5].

Algorithm 4: Generational Evolutionary Algorithm (GEA)

```
fill up the initial population pop with random solutions
while stopping criterion not met do
  create an empty population new_pop
  repeat
    select two parents from pop
    create two offspring by crossing the parents
    mutate the offspring
    add the offspring into the new population new_pop
  until new_pop full
  copy new_pop into pop
end while
```

3.5 Steady-State Evolutionary Algorithm

The steady-state evolutionary algorithm (SSEA) is similar to GEA, with the exception of maintaining a single population of solutions. Like in GEA, at every step two offspring are created by applying the evolutionary operators. But instead of filling a new population, the offspring replace the worst two individuals in the current population (Algorithm 5).

Algorithm 5: Steady-State Evolutionary Algorithm (SSEA)

```
fill up the population pop with random solutions
while stopping criterion not met do
  select two parents from pop
  create two offspring by crossing the parents
  mutate the offspring
  replace the two worst individuals in pop with the offspring
end while
```

3.6 Differential Evolution

Differential evolution (DE) is a population-based algorithm for optimizing functions on totally ordered spaces. It was developed by Price and Storn [9] as a variant of an evolutionary algorithm. The basic idea of DE is outlined in Algorithm 6.

Algorithm 6: Differential Evolution (DE)

evaluate the initial population P of random solutions
while stopping criterion not met **do**
  for $i := 1$ **to** *pop_size* **do**
    randomly select three different individuals $I_1, I_2, I_3 \in P$
    generate the candidate solution as $C := I_1 +$ weight$\cdot(I_2 - I_3)$
    alter the candidate by binomial crossover with the i-th individual
    evaluate the candidate
    if the candidate is better than the i-th individual **then**
      replace in P the i-th individual with the candidate
    end if
  end for
end while

4. Numerical Experiments and Results

The optimization methodology was experimentally applied to continuous casting of the construction steel AC-0113 at the Acroni steel plant in Jesenice, Slovenia. The computation was performed for a slab with the cross-section of 1.03 m $\times$ 0.20 m. Out of more than 20 influential process parameters, 12 spray coolant flows in the secondary cooling zone were subject to optimization (see Sect. 2.3). The task was to check whether the manual coolant flow setting used at the plant could be improved and which optimization method is the most suitable for this problem. The calculations were run on a 1.8 GHz Pentium IV computer where the execution time to evaluate a solution through numerical simulation was 2.5 minutes.

All applied methods used real vector representation of candidate solutions. Every method was run 5 times and in each run 400 solutions were evaluated (calculated with the simulator or read from the database). LO had no additional parameters. CG was implemented as shown in Algorithm 2, i.e. with preconditioning, Newton-Raphson and Polak-Ribiere methods. DS used reflection factor 1, contraction factor 0.5 and expansion factor 2. The evolutionary methods (GEA, SSEA and DE) operated on populations of 20 individuals. Both GEA and SSEA used tournament selection with the size of tournament 2, crossover probability 0.8 and mutation probability 0.05. DE applied the strategy *DE/rand/1/bin* [10] with crossover probability 0.5 and multiplication factor 0.5.

The results of the applied optimization methods are presented statistically in Table 2, while Fig. 3 shows the improvement of the best solution cost during the optimization process. The plots represent averages over five algorithm runs and are compared with the cost of the manual setting. They are divided into two graphs to enable a better view of the results.

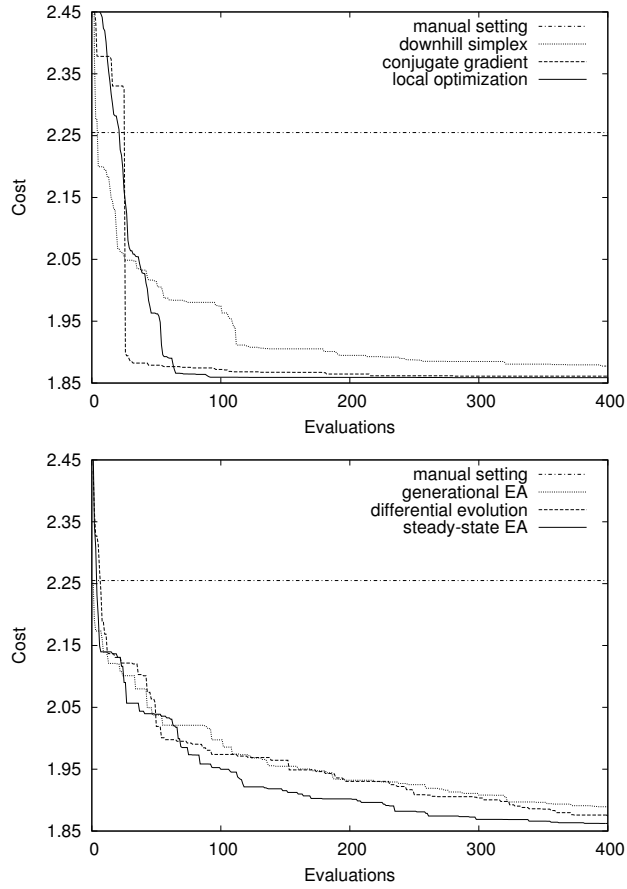


Figure 3. Performance of the optimization methods averaged over five runs.

Table 2. Results of the applied optimization methods in terms of cost given by Eq. (1).

Method Name	*Best*	*Average*	*Worst*	*St. dev.*
Generational EA	1.8638	1.8892	1.9137	0.0192
Downhill Simplex	1.8598	1.8775	1.8879	0.0137
Differential Evolution	1.8641	1.8741	1.8935	0.0116
Steady-state EA	1.8587	1.8622	1.8654	0.0028
Conjugate Gradient	1.8587	1.8612	1.8645	0.0027
Local Optimization	1.8587	1.8587	1.8587	0

All methods significantly improve the performance of the manual setting. LO outperforms all other methods by always reaching the best solution in less than

300 evaluations. The second best method is CG that makes a huge improvement in its first step (for one step the method needs 25 evaluations). SSEA is the best evolutionary method. The other methods (DE, GEA and DS) performed a little worse.

5. Discussion

The analysis of the solutions shows that the best result (cost value 1.8587) is always reached in the same point of the search space. This fact and superior performance of LO and CG over other methods indicate that the optimized function is probably unimodal. Although this outcome was not expected, it can be explained through the underlying physics. The 12 spray coolant flows subject to optimization are namely highly independent and in either monotonic or unimodal relationshis with the metallurgical criteria. Consequently, the resulting 12-dimensional cost function is also not very complex.

Although not as successful as LO, the applied evolutionary methods produce good results too, but they need more evaluations to converge (this is especially true for DE). They serve as a good comparison to LO and CG, since they are more robust and achieve good results also on more complex functions.

Our findings could be applied to similar optimization tasks in material processing. The physics behind such problems usually makes the search space simpler than expected. It is therefore a good idea to try methods like LO or CG in addition to EAs.

6. Conclusion

In the presented study, we have compared the performance of different optimization methods on process parameter tuning in continuous casting of steel. We have used an automatic optimization procedure based on a process simulator, compound cost function and various numerical optimization methods. All applied methods considerably improved the manual setting of process parameters. The best results were achieved by local optimization and the conjugate gradient method. These findings suggest that the cost function is of low complexity, most probably unimodal. However, our results are founded on certain assumptions, including the parameter intervals and discretization of the search space. With a finer discretization, the applied methods would probably perform differently. Testing the methods on different discretizations remains a task for further investigation. In addition, the improved coolant flow settings need to be practically evaluated at the plant.

Acknowledgment

The work presented in the paper was supported by the Slovenian Ministry of Education, Science and Sport under Research Programme P0-0541-0106

Intelligent Systems, and the European Commission under project COST 526: *Automatic Process Optimization in Materials Technology* (APOMAT).

References

[1] N. Chakraborti, R. Kumar, and D. Jain. A study of the continuous casting mold using a pareto-converging genetic algorithm. *Applied Mathematical Modelling*, 25:287–297, 2001.

[2] N. Chakraborti, R.S.P. Gupta, and T.K. Tiwari. Optimisation of continuous casting process using genetic algorithms: Studies of spray and radiation cooling regions. *Ironmaking and Steelmaking*, 30:273–278, 2003.

[3] N. Cheung and A. Garcia. The use of a heuristic search technique for the optimization of quality of steel billets produced by continuous casting. *Engineering Applications of Artificial Intelligence*, 14:229–238, 2001.

[4] B. Filipič and B. Šarler. Evolving parameter settings for continuous casting of steel. In: *Proceedings of the 6th European Conference on Intelligent Techniques and Soft Computing EUFIT'98*, Aachen, Germany, 1998, Vol. 1, pp. 444–449.

[5] D.E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, Reading, 1989.

[6] J.H. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, 1975.

[7] J.A. Nelder and R. Mead. A simplex method for function minimization. *Computer Journal*, 7:308–313, 1965.

[8] W.H. Press. *Numerical Recipes in Pascal: The Art of Scientific Computing*. Cambridge University Press, New York, 1989.

[9] K.V. Price and R. Storn. Differential evolution – a simple evolution strategy for fast optimization. *Dr. Dobb's Journal*, 22(4):18–24, 1997.

[10] K.V. Price and R. Storn. Differential evolution homepage (http://www.icsi.berkeley.edu/~storn/code.html).

[11] J.R. Shewchuk. An introduction to the conjugate gradient method without the agonizing pain. Technical report, School of Computer Science, Carnegie Mellon University, Pittsburgh, 1994 (http://www.cs.cmu.edu/~jrs/jrspapers.html).

[12] B. Šarler. Numerical procedure for calculating temperature field in continuous casting of steel. *Metals, Alloys, Technologies*, 30:217–223, 1996.

OPTIMIZATION OF GAS TURBINE BLADE CASTING USING EVOLUTION STRATEGIES AND KRIGING

Jürgen Jakumeit
ACCESS e. V.
Aachen, Germany
jakumeit@access.rwth-aachen.de

Michael Emmerich
System Analysis Group
University of Dortmund, Germany
michael.emmerich@cs.uni-dortmund.de

Abstract Blade casting is the a costly process in gas turbine manufacturing. This makes a reduction of the production costs by optimization very interesting for the industry. Today, high accuracy computer simulation codes are available for the casting process. This makes it possible to optimize this process by means of automatically scheduled computer experiments. This paper reports on recent studies on the optimization of the control parameters of the casting process, thereby comparing different objective and constraint functions.

Difficulties, like a large number of restrictions, time consuming computer experiments (evaluations), non-linear input-output mappings, and the need for flexible parallelisation schemes are faced when choosing the appropriate optimisation tool. As a robust parallel optimization technique evolutionary strategies (ES) have been suggested for such problems. However, these strategies typically need many function evaluations. Thus, metamodel-assisted ES are proposed that make extensive use of fast approximate evaluations obtained by means of Kriging surrogate models.

Keywords: Bridgeman Casting, Directional Solidification, Kriging Models, Numerical Optimisation, Constraint Handling, Evolution Strategy

1. Introduction

The highest gas turbine efficiency is achieved today with single-crystal (SX) or directionally solidified (DS) blading material, commonly cast in a Bridg-

man furnace (Fig. 1). The Bridgman process is controlled by time dependent parameters (withdrawal speed, heater temperatures), which are ideal for the application of numerical optimization [7, 3]. In addition, the blade casting is the most expensive process during the manufacturing of a turbine. This makes a reduction of the production costs by optimization very interesting for the industry.

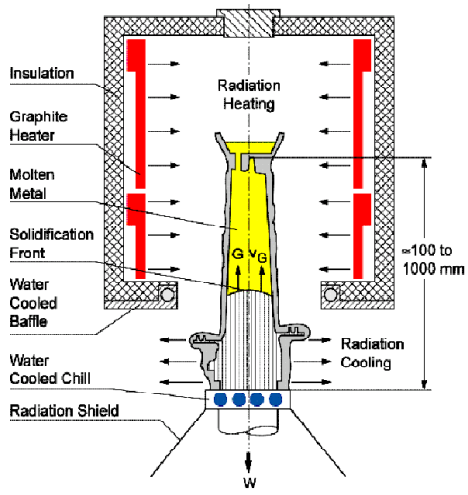


Figure 1. Schematic describtion of a Bridgman furnace used for directional solidification.

The hybrid FE/CV simulation program CASTS (Computer Aided Solidification TechnologieS) [6], under development at ACCESS e.V. since the late eighties, is used to predict numerically the transient temperature response during the Bridgman casting process. CASTS calculates transient temperature distributions in mold, core and alloy, taking into account both latent heat release as a function of fraction solid, and heat transfer resistance at material interfaces. The main output of the present release is the temperature and heat flux field. Based on this data, temperature gradients and defect maps can be calculated for each set of input process parameters, which are the basis for the evaluation of the turbine blade.

From the data obtained after a successful simulation for a suggested process schedule, various constraint values and an objective function value can be extracted. The following list gives an informal description of the optimization criteria:

- The process time should be minimized;

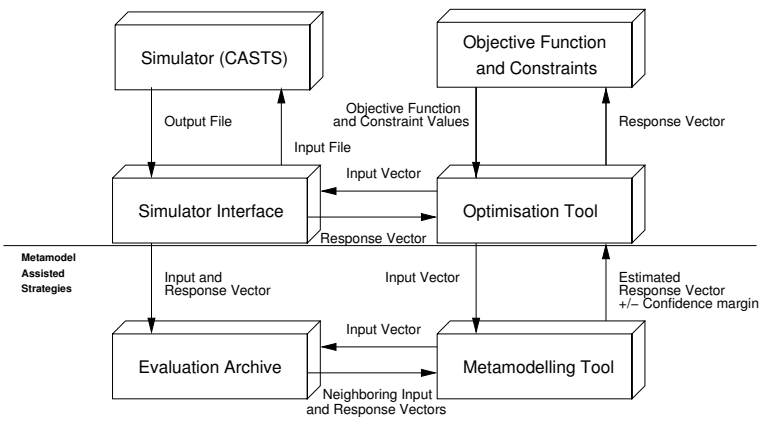


Figure 2. Data-flow in the metamodel-assisted optimization.

- A sufficiently low probability of local freckle formation, which is governed, in a first approximation, by the cooling rate at the liquidus isotherm has to be achieved;
- The degree of curvature of the solidification front has to be in a predefined range;
- The ratio G/v (temperature gradient over solidification speed) must be greater than a critical value (600 Ks/cm^2), describing the transition from columnar dendritic growth to an equiaxed grain structure.

In order to apply numerical optimization strategies, this problem has to be formulated as a black-box optimisation problem with implicit constraints:

$$\begin{aligned} &\min f(\mathbf{y}(\mathbf{x})), \\ s.t.\ g_1(\mathbf{y}(\mathbf{x})) \leq 0, \ldots,\ &g_{n_g}(\mathbf{y}(\mathbf{x})) \leq 0 \end{aligned} \tag{1}$$

Here, $f : \mathbb{R}^m \rightarrow \mathbb{R}$ denotes the objective function (i.e. the process time) and $g_1 : \mathbb{R}^m \rightarrow \mathbb{R}, \ldots, g_{n_g} : \mathbb{R}^m \rightarrow \mathbb{R}$ denote implicit constraint functions (attributed to the measured local defects). The vector valued function $\mathbf{y} : \mathbb{R}^n \rightarrow \mathbb{R}^m$ denotes the response function, which contains all criteria of interest that can be obtained by the (costly) evaluation of a given input vector $\mathbf{x} \in \mathbb{S}$ with $\mathbb{S}$ denoting the search space (typically a subset of $\mathbb{R}^n$). It is assumed that the evaluation of $\mathbf{y}$ is very (time) expensive in comparison to the evaluation of f or $g_1, \ldots, g_{n_g}$.

The choice of the mapping between the output of the simulator $\mathbf{y}(\mathbf{x})$ and the constraint function values and objective functions deserves a great deal of attention for this problem class. This is mainly because we deal with *local*

constraints. This means that for each point on the discretised surface of the turbine blade, we get a record of constraint criteria (the freckles probability, G/v ratio and degree of curvature). These values have to be summarised to a few constraint functions, which can be handled by the numerical optimization strategy. Besides the performance assessment of the optimisation strategy the choice of the constraint functions will be an important issue in this paper.

Once the objective and constraint criteria have been formulated in the standard form, numerical optimisation strategies can be applied to search for optimized input variables due to the objectives and constraints. The non-linear nature of this problem and the high dimensional search space demands for robust optimisation techniques [4]. Evolution strategies (ES) proved to be very effective in such scenarios and have been choosen here. In order to decrease the number of time consuming evaluations, statistical interpolation techniques that approximate responses based on data from previously evaluated points have been applied (cf. Fig. 2). Within the *metamodel-assisted evolution strategy* (MAES) these approximations are used in order to pre-screen solution candidates. It is well known that this kind of pre-screening makes ES much more effective in the presence of time-consuming evaluations. In order to decide, whether to evaluate a point precisely or by means of approximation, error estimations for the approximation are used. The Kriging method has been chosen, because it allows for such an error estimation. The Kriging method is based on statistics. It allows to specify confidence margins for each prediction, which can be used to formulate pre-screening criteria.

The rest of the paper is organized as follows: After a brief introduction to metamodelling with Kriging (Sect. 2), in Sect. 3 the MAES optimization tool is introduced. Then in Sect. 4 different formulations of the objective and constraint functions for the gas turbine blade casting problem are specified and discussed. Results are reported in Sect. 5 and finally we summarize and state questions for future research.

2. Kriging Models

Given a database of precise evaluations $\mathbf{y}^{(1)} = f(\mathbf{x}^{(1)}), \ldots, \mathbf{y}^{(m)} = f(\mathbf{x}^{(m)})$ that have been obtained from costly computer experiments, we may ask for a tool that can utilise this data in order to predict results at new points. This tool should (1) be considerably faster than the precise evaluation and (2) give a continuous interpolation of the multivariate response function $\mathbf{y}$ for irregulary distributed input data $\mathbf{X}$.

Kriging metamodels can be used for this purpose: They provide exact interpolations of black-box functions after they have been trained with the known results $\mathbf{X}, \mathbf{y}$. Beside a prediction $\hat{\mathbf{y}}(\mathbf{x})$ of the true response, they also provide an error estimate for the prediction of $\hat{\mathbf{s}}(\mathbf{x})$.

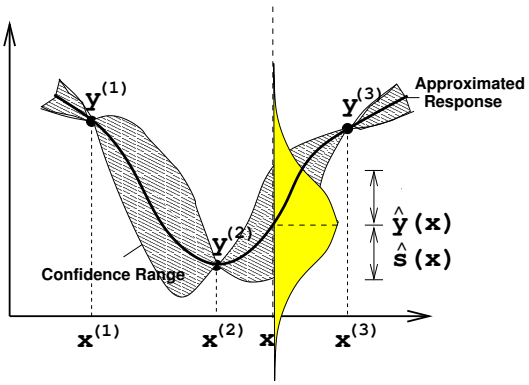


Figure 3. Understanding interpolation with Kriging for a problem with an 1D input vector and 1D response: With three training patterns $x^{(i)}$, $i = 1, \ldots, 3$, the thick line corresponds to the approximate response $\hat{y} = \hat{f}(x)$. The two thin lines confine the confidence interval of the response. The former is equal to the expected value of the random variable $\mathcal{F}$ at a new point x.

Taking a Bayesian stance, the output of Kriging, as used in this paper, is the conditional Gaussian distribution with mean $\hat{\mathbf{y}}(\mathbf{x})$ and standard deviation $\hat{\mathbf{s}}(\mathbf{x})$ describing the likelihood for the true realisation of the function value at site $\mathbf{x}$ given the prior information $\mathbf{X}, \mathbf{y}$ and an assumption of the correlation structure of the input-output mapping. The assumption that is made in Kriging is that the measured response is part of a sample path of a Gaussian process correlated in the input space via a distance-based correlation function. Figure 3 visualises this for the simple case of 1D input and 1D output vectors.

Kriging is a standard tool for metamodelling and statistical interpolation. The Kriging tool used in this paper has been proposed by Sacks et al. [9] for metamodelling purposes. A detailed description can be found in the given reference.

3. Metamodel-Assisted Optimization Tools

A simple strategy for metamodel assisted optimisation is to initialise the metamodel by a design of experiments (DoE) and then use a numerical optimisation strategy to find the optimum of the metamodel [5]. This is done in an iterative way in the *Kriging Monte Carlo Strategy (KMCS)*. However, first tests reported for the Bridgeman process [3] indicated that this strategy suffers from premature convergence. Within the same study it turned out that the metamodel-assisted derandomized evolution strategy (MA-DES) is a very promising strategy for this problem class.

The MA-DES features the selection scheme of a $(1, \lambda)$-ES [1]. In order to achieve an accelerated adaptation of step-sizes, the strategy has been equipped

with the derandomized self-adaptation of step-sizes as suggested by Ostermeier et al. [8].The main idea in the $(1 + \nu < \lambda)$-MA-DES is not to evaluate all λ generated solution candidates (individuals) by means of precise evaluations, but to consider only the ν most promising solutions with regard to a pre-screening criterion based on approximate evaluations with the metamodel. Thus, the number of precise evaluations per iteration reduces from λ to $\nu \ll \lambda$. For the given problem class $\nu = 4$ and $\lambda = 20$ turned out to be good settings. A detailed description of the MA-DES can be found in a previous publication by the authors [3].

In order to avoid premature convergence, lower confidence bounds (cf. Fig. 3) have been calculated for all optimisation criteria, including constraints (cf. Sect. 4): $\hat{y}_{i,lb}(\mathbf{x}) = \hat{y}_i(\mathbf{x}) - \omega\hat{s}_i(\mathbf{x})$, $i = 1, \ldots, n_g + 1$. Here ω denotes a confidence factor (here: $\omega = 2.0$) and $\hat{y}_i(\mathbf{x})$ and $\hat{s}_i(\mathbf{x})$are mean values and standard deviations specifying the response approximations at point $\mathbf{x}$.

4. Objective Function Formulation

The freckle probability, the curvature of the solidification front and the G/v ratio (cf. Sect. 1) are evaluated by counting the number of "bad" nodes, i.e. nodes with freckle probability, the curvature of the solidification front is above 20° or the G/v ratio is below 600 Ks/cm^2. The criteria can be tuned by changing the limits (0,20°,600 Ks/cm^2).

A great advantage of this criteria formulation is that these three criteria can now be easily combined due to their similar definition by the number of "bad" nodes. Figure 4 shows in the three left plots the nodes with too high curvature, too low G/v ratio or freckle tendency for the optimal withdrawal profile found by the MA-DES.

The individual criteria can be combined into one plot by giving nodes with freckle tendency a white color, nodes with to high curvature a lavender color and those with to low G/v ratio an orange color, while all "good" nodes are blue (right plot in Fig. 4). Beside the usefulness for numerical optimization such a combined visualization of different casting quality criteria can be helpful for any casting result evaluation.

For the optimization of an industrial turbine blade, those nodes that do not fulfill the criteria mentioned above, were weighted by their volume. A volume can be assigned to the node by the CASTS control volume approach. An additional factor was introduced to normalize the three criteria. This normalization reflects that a few nodes with freckles are equally bad as several nodes with a too high G/v value and many nodes with a wrong curvature. Figure 5 shows the influence of the weighting and normalization of the criteria.

Finally the process time has to be integrated into the objective function formulation. Here the first goal was to achieve a blade with no bad nodes with a

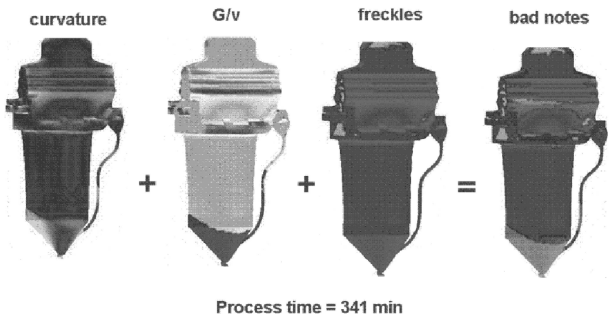


Figure 4. Solidification front for a gas turbine blade. The surface of the turbine blade is colored from blue to white with increasing curvature, G/v ratio or freckle tendency. The right plot shows the combination of these 3 criteria. Nodes with freckle tendency have a white color, nodes with to high curvature a lavender color and those with to low G/v ratio are orange, while all "good" nodes remain blue.

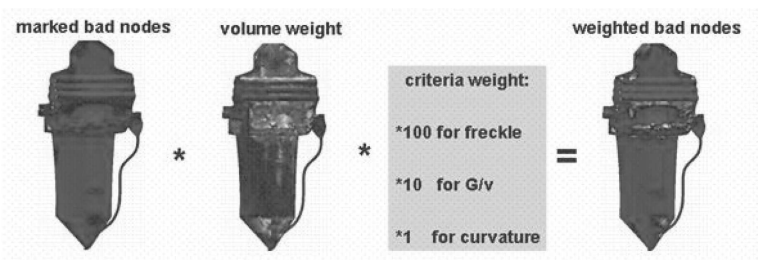


Figure 5. Improved optimization criteria by a weighting of nodes by the assigned control volume and an additional criteria weighting factor.

process time below an acceptable time. Only when all bad nodes are removed the optimization should try to reduce the process time to a minimum. Therefore, the objective function was defined as follows:

$$f = \begin{cases} \text{weight of bad nodes + process time} & \text{if weight } > 0 \quad \text{and time } < 10000 \\ \text{weight of bad nodes + 10000} & \text{if weight } > 0 \quad \text{and time } < 10000s \\ \text{process time in } [s] & \text{if weight = 0} \end{cases} \tag{2}$$

As long as the process time is above the acceptable time and bad nodes exist, the weighted sum of the freckle, G/v and curvature criteria are added to the process time to give the objective value. If the process time is below the acceptable time, the weighted sum plus a constant value of the acceptable time in seconds is used as objective function. This makes the optimization focus on the improvement of the blade quality by reducing the number of bad nodes. If a high quality blade with no bad nodes is achieved the optimization can again

try to reduce the process time. There for the process time in seconds becomes the objective function, when all bad nodes are removed.

5. Results on the Industrial Test-Case

Before applying the metamodel based ES to an industrial gas turbine balde the two metamodel based strategies were compared with the standard downhill simplex algorithm [10] and the derandomised evolution strategy (DES) for the optimization of a simplified blade geometry [3]. The MA-DES variants clearly outperform the conventional DES and the downhill simplex algorithm. The iterative Kriging finds a rather good solution in the first sampling but no further improvements can be found by the reduction of the sampling range in the following iterations. The MA-DES was applied to the optimization of an industrial turbine blade, the cluster of 3 SX blades mentioned above. An acceptable process time of 10000 seconds was used in the optimization. Figure 6 summarizes the result of the optimization of cluster of SX blades. The withdrawal profile was discretised using 6 velocity values at the withdrawal positions: 6 cm, 20 cm, 28 cm, 36 cm, 44 cm and 60 cm. The left upper plot shows the convergence of the MA-DES. The dotted red line gives the result of each simulation, while the full red line shows the objective value of the best solution so far. After a significant improvement within the first 12 simulations the optimization could not yield further improvements after 40 simulations and the process was stopped. A withdrawal profile which yields no bad nodes could not be obtained within 45 simulations. For the main improvements found, the blades with the marked bad nodes are plotted below the convergence plot (the color of the frame of the plots gives the position on the convergence curve). The improvements found in the objective function can obviously not be visualized by simply marking the bad nodes on the surface of the blade. For comparison the turbine blade was also optimized by a downhill simplex (DS) algorithm. The DS starts with a rather good solution but finds no further improvements.

The right plot of Fig. 6 shows the resulting withdrawal profiles. Starting from an initial guess the MA-DES finds the main form of the withdrawal profile with 20 simulations. The withdrawal velocity should be small around 3 mm/min at the beginning of the process. The velocity can be increased to 6 mm/min in the range of the blade itself. For the solidification of the thick basis of the blade slow velocities around 3 mm/min are again necessary. For the last centimeter of the process, the increased velocity leads to a significant reduction of the total process time. Compared to a withdrawal profile designed by a casting engineer, the numerically optimized withdrawal profile leads to a higher quality of the blade within a shorter process time.

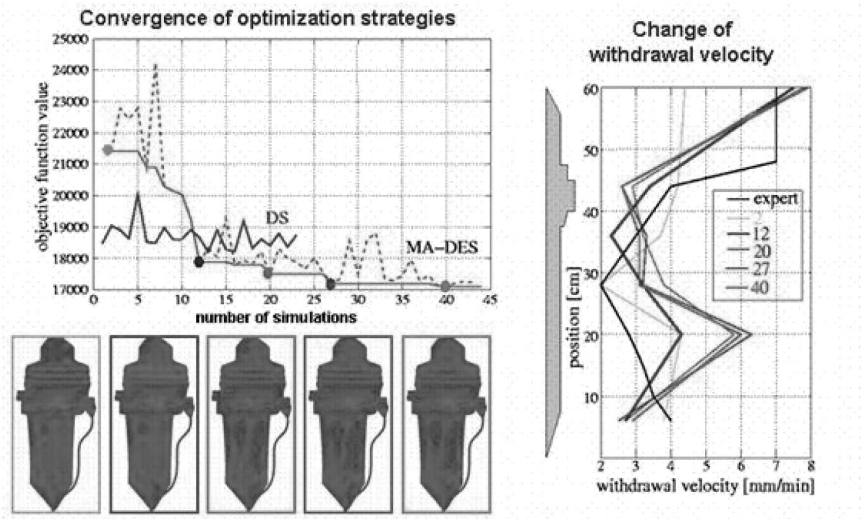


Figure 6. Result of the optimization of an industrial turbine blade.

6. Summary and Outlook

The MA-DES was successfully used to optimize the withdrawal profile of the Bridgman process for casting industrial turbine blades. The results for a simplified test case show that it outperforms classical methods with respect to the results obtained with the same number of precise evaluations. The MA-DES was finally applied to a industrial turbine blade. The result of the optimization is a withdrawal profile which leads to a higher quality of the turbine blade and a shorter process time compared to a withdrawal profile designed by hand. However, an withdrawal profile which yields a blade with no bad nodes could not be found. It is very likely change that blades with no bad nodes can be achieved, by solely variegating the withdrawal profile. For the future the objective function definition has to be further adjusted to the needs of the casting engineers. A discretization of the withdrawal profile with more than 6 velocities would be desirable. Moreover, multi-objective optimisation algorithms could be utilized for detecting trade-offs among criteria [2].

Acknowledgments This work was supported by the Deutsche Forschungsgemeinschaft (DFG) as part of the *Collaborative Research Center* "Computational Intelligence" (SFB 531).

References

[1] H.-G. Beyer and H.-P. Schwefel. Evolution strategies - A comprehensive introduction. *Natural Computing*, 1:3–52, 2002.

[2] K. Deb. *Multi-objective optimisation using evolutionary algorithms*, J. Wiley and Sons, Chichester, UK, 2001.

[3] M. Emmerich and J. Jakumeit. Metamodel-Assisted Optimisation with Constraints: A Case Study in Material Process Design. In *Proceedings of the Conference on Evolutionary Methods for Design, Optimization and Control with Application to Industrial Problems (EUROGEN 2003)*, Barcelona, Spain, 2003.

[4] M. Emmerich, M. Schallmo, and T. Bäck. Industrial Applications of Evolutionary Algorithms: A comparison to traditional methods. In: I. Parmee et al.: *Optimisation in Industry*, Springer, Berlin, 2001, pp. 304–314.

[5] J. Jakumeit, M. Herdy, and M. Nitsche. Parameter optimization of the sheet metal forming process using an iterative parallel Kriging algorithm. *Design Optimization: International Journal for Product & Process Improvement*, submitted in 2003.

[6] G. Laschet, J. Neises, and I. Steinbach. *Micro-Macrosimulation of casting processes*. 4ième école d'été de Modélisation numérique en thermique, C8 1-42, Porquerolles, 1998.

[7] G. Laschet, M. Schallmo, and N. Hofmann. Optimization tools for Bridgman casting process. In B. Thomas and C. Beckermann (eds.): *Proc. 7th Conf, on Casting, Welding and advanced Solidification*, TMS editions, San Diego, 1998, pp. 1095–1102.

[8] A. Ostermeier, A. Gawelczyk, and N. Hansen. Step-size adaptation based on non-local use of selection information. In Davidor et al. (eds.): *Parallel Problem Solving from Nature - PPSN III, Lecture Notes in Computer Science*, 866:189–198, 1994.

[9] J. Sacks, W.J. Welch, W.J. Mitchell, and H.-P. Wynn. Design and analysis of computer experiments. *Statistical Science*, 4:409–435, 2000.

[10] H.-P. Schwefel. *Evolution and Optimum Seeking*, Wiley, NY, 1995.

ELECTRICAL ENGINEERING DESIGN WITH AN EVOLUTIONARY APPROACH

Gregor Papa
Computer Systems Department
Jožef Stefan Institute, Ljubljana, Slovenia
gregor.papa@ijs.si

Barbara Koroušić Seljak
Computer Systems Department
Jožef Stefan Institute, Ljubljana, Slovenia
barbara.korousic@ijs.si

Jurij Šilc
Computer Systems Department
Jožef Stefan Institute, Ljubljana, Slovenia
jurij.silc@ijs.si

Abstract This paper presents two engineering design problems, both of which were solved by evolutionary algorithms. The evolutionary approach is used in universal electro-motor geometry optimization and integrated circuits area/time optimization. In the first case we improve the efficiency of a universal motor; where the goal is to find a new set of independent geometrical parameters for the rotor and the stator with the aim of reducing the motor's power losses, which occur in the iron and the copper. In the second case we improve some parts of the high-level synthesis process of integrated circuits by considering the concurrency of operation scheduling and resource allocation constraints to ensure a globally optimal solution in a reasonable time.

Keywords: Engineering design, Electro-motor, Integrated circuit, Evolutionary optimization

1. Introduction

Evolutionary techniques are used in various search methods for a range of different optimization areas. Their undetermined approach gives them an advantage when it comes to multi-criteria problems and problems with more local

optima [8]. For this reason we adopted an artificial approach to the solving of our design problems using a genetic algorithm (GA) [2, 7]. The GA, as a frequent implementation of evolutionary techniques, is an optimization method based on the mechanism of evolution and natural genetics. This algorithm has already proved to be very efficient in a wide range of different optimization procedures where the exact equations are not available or some non-linearities are present [4]. In spite of its simplicity, the GA has proved to be an efficient method for solving various optimization and classification problems, in areas ranging from economics and game-theory to control-system design [3, 5, 9, 11, 12].

This paper presents two engineering design problems, both of which were solved with evolutionary algorithms. The evolutionary approach is used in *universal electro-motor geometry optimization* (UM design), and *integrated circuits area/time optimization* (IC design).

2. Problems in Engineering Design

2.1 Universal Motor Design

Many common home appliances, such as vacuum cleaners and mixers, as well as power tools, such as drills and saws, are generally powered by a universal motor (UM) [16]. This type of motor has many advantages that make the UM such a popular choice for home appliances and power tools: a large output power in relation to its small size, high starting and running torque, variable speed that can be regulated in a simple way, and low manufacturing costs.

Home appliances and power tools need as low an energy consumption, i.e., input power, as possible, while still satisfying the needs of the user by providing sufficient output power. The ratio of the output power to the input power defines the efficiency of the motor, which can be improved by reducing some of the main power losses in the motor, i.e., those that originate in the iron and the copper. This can be done by optimizing the geometry of both the rotor and the stator (see Fig. 1). Because of the high magnetic saturation of the iron in a UM the problem is a highly non-linear one.

The rotor-and-stator unit of a UM is constructed by stacking the rotor/stator iron laminations. The shape and the profile of the rotor/stator lamination are described by several two-dimensional geometrical parameters. There are two types of parameter: the *invariable* and the *variable*. Invariable parameters are fixed; they cannot be altered, either for technical reasons or because of the physical constraints of the motor. Variable parameters are those that do not have predefined optimum values. Some of these variable parameters are mutually independent and without any constraints. In our case we optimize 12 mutually independent variable parameters.

The efficiency of a UM is defined as the ratio of the output power P_{out} to the input power P_{inp}, and it depends on various power losses, which include:

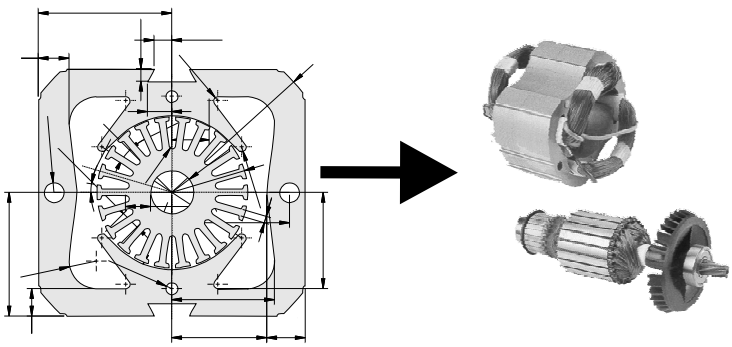


Figure 1. Geometrical parameters of the stator and rotor of a UM.

copper losses P_{Cu}, iron losses P_{Fe}, brush losses P_b, ventilation losses P_v, and friction losses P_f. When considering all the mentioned losses and the output power, the overall efficiency η of a UM can be defined as

$$\eta = \frac{P_{out}}{P_{inp}} = \frac{P_{out}}{P_{out} + P_{Cu} + P_{Fe} + P_b + P_v + P_f}.$$

2.2 Design of Integrated Circuits

High-level synthesis [6] is an automatic design process that transforms the initial behavioral description of the circuit (especially ASICs) into the final specification of the RTL. The process consists of the following: compilation, transformation, scheduling, allocation and binding. Of these, the operation scheduling and the resource allocation are the most important tasks of the high-level synthesis because they are at the core of the design and crucially influence both the design and the final layout (see Fig. 2).

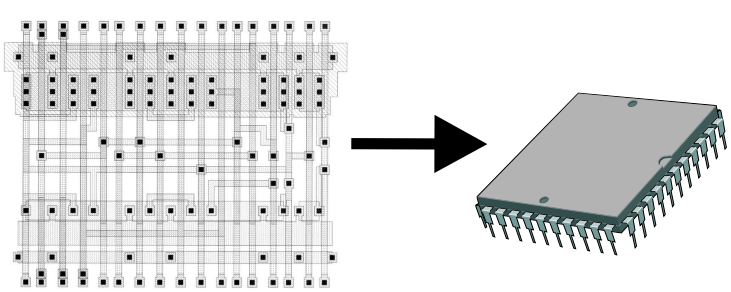


Figure 2. Time/area optimized layout of an IC.

Due to the interdependence of these two tasks, the solution of one task depends on an estimation of the solution of the other task, which is not solved yet. The scheduling of the operation into different control steps therefore affects the allocation of operations to different units. The interaction of these two tasks presents formidable obstacles to the goal of optimization [1]. There are, however, some approaches to concurrent solving, but their solutions, to some extent, are less than optimal [10].

3. The Genetic Algorithm in UM and IC Design

When reducing the main power losses in a UM design by optimizing the geometry of the rotor/stator lamination we have to deal with a complex search space and its non-linear behavior. In IC design we apply the concurrency of interdependent tasks with their opponent constraints. Because traditional search-and-optimization methods have proved to be inefficient at finding the solution under such conditions, we decided to apply a GA. This heuristic method requires only a little information to provide a robust, yet flexible, search in a wide and complex search space.

3.1 UM Design Procedure

Conventional motor design can be upgraded with a genetic algorithm. The advantages of this approach are that: there is no need for an experienced engineer to be present during the whole process, except at the beginning to decide on the initial design, and there is no need to know the mechanical and physical details of the problem. The problem can be solved without any knowledge of the problem, we only need some finite-element program to evaluate each solution.

3.1.1 Experimental Results. The proposed evolutionary design approach is evaluated by estimating the actual improvement in the efficiency of an initial UM that is designed using the conventional and evolutionary design approaches.

We optimized the UM twice. The first one (*Opt 1*) was full optimization, where all parameters were optimized, while in the second case (*Opt 2*) we fixed the outer boundaries of the UM to get the design with the same amount of material and therefore the same initial material costs.

We made prototypes of both the optimized and the costs-optimized motors and measured the real power losses and the efficiencies of the motors. These values are shown in Table 1. The results are only slightly different from those calculated with a finite-element program. The main reason for this difference can be explained by the non-exact calculation of the iron losses, due to a variation in the material's properties.

Table 1. UM evaluation results.

Feature	*Analytical calculation*			*Prototype calculation*		
	Initial	*Opt 1*	*Opt 2*	*Initial*	*Opt 1*	*Opt 2*
P_{inp} [W]	1 044	970	982	1 050	990	1 000
P_{out} [W]	731	731	731	730	730	730
$\Delta P = P_{inp} - P_{out}$ [W]	313	239	251	320	260	270
$\eta = \frac{P_{out}}{P_{inp}}$ [%]	70.0	75.8	74.8	69.5	73.7	73.0
ΔP improvement [W]		74	62		60	50
η improvement [%]		5.8	4.8		4.2	3.5

3.2 IC Design Procedure

The promising results of different evaluations [5, 11, 12] led us to the *Evolutionary Concurrent Scheduling and Allocation* (ECSA) design approach [10]. This approach considers scheduling and allocation constraints, allows a short design time and can find globally optimal solutions. The input description of the circuit is transformed into two basic (initial) schedules, obtained with the ASAP and ALAP algorithms. The functional units used in the first case are those that are the fastest for each operation, and in the second case are those that are the slowest for each operation. These two schedules present some kind of boundary solutions, since all the other solutions are executed in between the time limits defined by these two schedules. In other words, no other solution can be faster or slower, irrespective of the combinations of used units.

Each solution has to be properly encoded, i.e., each operation's start time and functional unit have to exist in the chromosome. The initial population is built upon the two initial solutions, which are multiplied to form the population with the so-called boundary solutions. The optimal solution has to be somewhere in-between the boundaries, therefore genetic operators (crossover, mutation, variation) transform those encoded solutions. With the transformations their start times and allocated functional units are changed. The appropriateness of the proposed approach is tested by a computer implementation of the ECSA algorithm, which is used with test-bench ICs.

In addition to simple GA operators also the independent GA approach [10] was used. There is no need to preset some working parameters, e.g., the number of generations, the population size, and the probabilities of crossover, mutation and variation. These parameters are set automatically during the optimization phase, depending on the progress and the speed of the optimization.

- *Setup.* If the chromosome that represents a solution is large, then the population size also has to be large enough to ensure that many different

chromosomes will be involved in a search. The population size therefore depends on the size of the chromosome or the complexity of the problem.

- *Crossover.* Considering four candidates-two parents and their two offspring-only the first and the third, rated according to their fitness, pass to the next generation. This forces at least one of the offspring to be passed to the next generation in addition to the best candidate. Otherwise the offspring have only a small influence on new generations, since the crossing of two good parents probably produces offspring that are not so good. They might, however, be good after a few more transformations.

- *Mutation.* Chromosomes with low fitness are mostly exposed to mutation. Each position in the chromosome string is mutated if that position of the chromosome is of the same value in the majority of chromosomes in the population. This is the way to change the bad characteristics in "poorly fitted" chromosomes and to redirect the search to another direction. In the case of "well-fitted" chromosomes, values are mutated if they differ from the majority of values in other good chromosomes at the same position. This ensures faster convergence in the final stages of the optimization.

- *Variation.* The interchange of the values of two positions, as described for the basic operators, is performed if the frequency of the value in that position in the population of one position is high and the frequency of another bit is low.

3.2.1 Experimental Results. The ICs used for the evaluation are chosen on the basis of their appearance in the literature and similar studies. They differ in terms of size and the number of operation types. The ECSA algorithm is evaluated by a comparison with nearly optimal [15] force-directed scheduling (FDS) [14]. FDS tries to optimally schedule the DFG considering a uniform distribution the operations of the same type over the available control steps.

Tables 2 and 3 show the results of the following evaluations: FDS with fast units, FDS with slow units, and ECSA with basic and independent genetic operators. There are two types of DFGs for each circuit. The first, or plain, is an ordinary data-flow graph with nodes that represent operations, as described in similar studies (Table 2); and the second, or improved [10], considers the input variables (start registers) via some additional nodes to ensure a more accurate estimation of the registers and the buses needed to implement the circuit (Table 3).

Differential equation: Because of the small circuit size there is no improvement in the solutions obtained with the ECSA algorithm (either basic or independent) when considering an ordinary DFG-all the solutions are of a larger

Table 2. The evaluation results of the ECSA algorithm with different test-bench ICs.

Algorithm	*Size* [gates]	*Registers*	*Buses*	*Delay* [steps]	*Runtime* [s]
			Differential equation		
FDS-fast	23 249	17	6	6	0.01
FDS-slow	7 173	18	6	20	0.01
ECSA-basic	23 914	18	8	6	0.11
ECSA-independent	23 914	17	6	6	0.09
			Fifth-order elliptic filter		
FDS-fast	23 883	26	8	17	0.02
FDS-slow	10 970	30	6	78	0.03
ECSA-basic	18 962	29	4	21	3.80
ECSA-independent	31 844	30	8	17	1.60
			Bandpass filter		
FDS-fast	31 179	34	10	10	0.01
FDS-slow	10 600	35	8	44	0.04
ECSA-basic	23 883	33	8	11	1.70
ECSA-independent	34 219	33	8	10	1.30
			Least-mean-square filter		
FDS-fast	45 771	68	12	13	0.40
FDS-slow	13 270	72	8	70	2.79
ECSA-basic	42 123	67	10	14	6.30
ECSA-independent	89 889	69	30	15	8.05

size. But when we consider the start registers (input variables) there are some ECSA solutions with a slightly larger size and a smaller number of buses.

Fifth-order elliptic filter: The evolutionary method with a basic approach finds a smaller circuit with a smaller number of buses and a slightly longer execution time for the ordinary DFG, whereas the independent approach could not find any improved solution. When dealing with the improved DFG, both approaches (basic and independent) find considerably smaller circuits with a slight increase in the execution time, while the independent approach also finds the solution with a substantial decrease in the required number of registers and buses.

Bandpass filter: Both ECSA methods find, when dealing with the ordinary DFG, the solutions with a smaller number of registers and buses; the basic approach also finds the smaller circuit, but with a slightly longer execution time. When dealing with the improved DFG, both approaches find the solutions with the same circuit size and execution time as the comparable FDS solution, but the required number of registers and buses is considerably smaller for the ECSA solutions.

Table 3. The evaluation results of the ECSA algorithm with different test-bench ICs.

Algorithm	*Size* [gates]	*Registers*	*Buses*	*Delay* [steps]	*Runtime* [s]
		Differential equation with start registers			
FDS-fast	23 249	10	9	6	0.01
FDS-slow	7 173	11	9	20	0.01
ECSA-basic	31 210	10	7	6	0.15
ECSA-independent	23 914	10	7	6	0.35
		Fifth-order elliptic filter with start registers			
FDS-fast	23 883	21	16	17	0.02
FDS-slow	10 970	25	16	78	0.04
ECSA-basic	18 962	24	16	19	4.80
ECSA-independent	15 922	18	9	21	3.40
		Bandpass filter with start registers			
FDS-fast	31 179	25	23	10	0.02
FDS-slow	10 600	26	23	44	0.04
ECSA-basic	31 179	23	19	10	2.40
ECSA-independent	31 179	23	19	10	2.90
		Least-mean-square filter with start registers			
FDS-fast	45 771	33	29	13	0.48
FDS-slow	13 270	37	27	70	3.52
ECSA-basic	45 220	32	25	13	9.20
ECSA-independent	63 517	33	25	13	12.30

Least-mean-square filter: At the expense of a small increase in the delay, the basic ECSA is able to decrease the size and lower the number of registers and buses of the ordinary DFG; but the independent ECSA is not able to improve any parameter. When dealing with the improved DFG, the basic ECSA is able to keep the initial delay, to decrease the circuit size and to lower the number of required registers and buses. The independent ECSA is only able to decrease the number of buses while increasing the circuit size.

There are slightly longer runtimes when the ECSA algorithm is used. But considering the speed (a few seconds) and the evaluation presented in [13], where the runtimes for larger circuits increase enormously (exponentially) when the FDS algorithm is used, we can conclude that small and large circuits can be designed and optimized with the use of the proposed evolution-based algorithm, which exhibits a linear increase in the design time with an increase in circuit size.

4. Conclusions

In the UM optimization we used an evolutionary approach to improve the efficiency of an electro-motor. The goal of our optimization was to find the new

set of independent geometrical parameters of the rotor and the stator with the aim of reducing the motor's power losses, which occur in the iron and the copper. The approach proves to be a simple and efficient search-and-optimization method for solving this day-to-day design problem in industry. It outperforms, by a significant improvement of the motor's efficiency, a conventional design procedure that was used previously. By using the GA we are able to reduce the iron and the copper losses of an initial UM by at least 20%, and increasing the GA running time or setting its parameters more appropriately could improve on this result.

In the IC area/time optimization we used an evolutionary approach to some parts of IC design. The work was focused on ASICs that need an even more sophisticated design due to their specific use. Optimally scheduled operations are not necessarily optimally allocated to functional units. To ensure optimum allocation we need to consider some allocation criteria while the scheduling is being done. The evolutionary approach considers scheduling and allocation constraints and ensures a globally optimal solution in a reasonable time. To evaluate our method we built an algorithm and implemented it with a computer. It is used with a group of test-bench ICs. These circuits are chosen because the same types were used in similar studies. The results of the evaluation of a computer-implemented algorithm show that the evolutionary methods are able to find a solution that is more appropriate in terms of all the considered and important objectives than is the case when working with classical deterministic methods.

Acknowledgment

The work presented in the paper was supported by the Slovenian Ministry of Education, Science and Sport (research programme P2-0098 *Computer Structures and Systems*), and partially by DOMEL, Electromotors and household devices, d. d., Železniki, Slovenia.

References

[1] J.R. Armstrong and F.G. Gray. *VHDL Design: Representation and Synthesis*. Prentice Hall PTR, 2000.

[2] T. Bäck. *Evolutionary Algorithms in Theory and Practice*. Oxford University Press, New York, 1996.

[3] D. Dasgupta and Z. Michalewicz. *Evolutionary Algorithms in Engineering Applications*. Springer-Verlag, 1997.

[4] R. Drechsler. *Evolutionary Algorithms for VLSI CAD*. Kluwer Academic Publishers, 1998.

[5] B. Filipič and J. Štrancar. Tuning EPR spectral parameters with a genetic algorithm. *Applied Soft Computing*, 1:83–90, 2001.

[6] D.D. Gajski, N. Dutt, A. Wu, and S. Lin. *High-Level Synthesis: Introduction to Chip and System Design*. Kluwer Academic Publishers, 1992.

[7] D.E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, Reading, 1989.

[8] C.L. Karr, I. Yakushin, and K. Nicolosi. Solving inverse initial-value, boundary-value problems via genetic algorithm. *Engineering Applications of Artificial Intelligence*, 13:625–633, 2000.

[9] B. Koroušić-Seljak. Heuristic methods for a combinatorial optimization problem - real-time task scheduling problem. In *Proceedings of the Artificial Networks in Engineering Conference (ANNIE'99)*, St. Louis, USA, 1999, pp. 1041–1046.

[10] G. Papa. Concurrent operation scheduling and unit allocation with an evolutionary technique in the process of integrated-circuit design. *Ph.D. thesis*, Faculty of Electrical Engineering, University of Ljubljana, Slovenia, 2002.

[11] G. Papa, B. Koroušić-Seljak, B. Benedičič, and T. Kmecl. Universal motor efficiency improvement using evolutionary optimization. *IEEE Transactions on Industrial Electronics*, 50:602–611, 2003.

[12] G. Papa and J. Šilc. Automatic large-scale integrated circuit synthesis using allocation-based scheduling algorithm. *Microprocessors and Microsystems*, 26:139–147, 2002.

[13] G. Papa, J. Šilc, and R. Wyrzykowski. Scheduling algorithms based on genetic approach. In *Proceedings of the 4th Conference on Neural Networks and Their Application*, Zakopane, Poland, 1999, pp. 469–474.

[14] P.G. Paulin and J.P. Knight. Force-directed scheduling in automatic data path synthesis. In *Proceedings of the 24th ACM/IEEE Design Automation Conference*, Miami, USA, 1987, pp. 195–202.

[15] P.G. Paulin and J.P. Knight. Scheduling and binding algorithms for high-level synthesis. In *Proceedings of the 26th ACM/IEEE Design Automation Conference*, Las Vegas, USA, 1989, pp. 1–6.

[16] P.C. Sen. *Principles of Electric Machines and Power Electronics*. John Wiley & Sons, 1996.

TEST PATTERN GENERATOR STRUCTURE DESIGN BY GENETIC ALGORITHM

Tomasz Garbolino
Institute of Electronics
Silesian University of Technology, Gliwice, Poland
tomasz.garbolino@polsl.pl

Gregor Papa
Computer Systems Department
Jožef Stefan Institute, Ljubljana, Slovenia
gregor.papa@ijs.si

Andrzej Hławiczka
Institute of Electronics
Silesian University of Technology, Gliwice, Poland
hlawicz@boss.iele.polsl.gliwice.pl

Abstract A new type of a deterministic test pattern generator (TPG) called DTI-MFSR is presented in the paper. It is based on a feedback shift register composed of D- and T-type flip flops and inverters. Moreover, it is equipped with a non-linear combinational function that can invert, using XOR gates connected to the inputs of the flip-flops, any bit in any pattern generated by the register. Thus, any arbitrary test sequence can be produced at the outputs of the new TPG. The optimization algorithm, which for the given deterministic test set minimizes the area overhead of the DTI-MFSR generating this set of vectors, is also proposed in the paper. The optimization is based on genetic algorithm (GA). The initial structure of the TPG is encoded and multiplied with some variations to form the initial population. The search for the optimal structure of the TPG is performed by selection, crossover, and mutation operators, while each solution is evaluated by the external evaluation tool. Quality of the proposed TPG is proven by experimental results obtained using ISCAS benchmarks.

Keywords: TPG, Structure, Genetic algorithm

1. Introduction

There are different techniques of combinational circuit testing. In general, the outputs of the testing structure are connected to the inputs of the circuit under test (CUT) to simulate possible combinations of input signals and therefore to test the behavior of the circuit. The exhaustive or pseudoexhaustive test provides 100% fault coverage but testing time is prohibitively long in the case of large circuits. On the other hand, the pseudorandom testing requires much shorter time for test application but it doesn't achieve full fault coverage. The method that combines advantages of the above techniques is deterministic testing. It provides 100% fault coverage in very short time. Moreover, energy and average power consumption during deterministic test are much lower than in the case of previously mentioned methods. The last feature is particularly important in battery-supplied and low-power systems. The drawback of deterministic testing is large area overhead of test pattern generator (TPG) that is necessary to produce vectors provided by automatic TPG tool. In order to decrease complexity of TPG, built-in self test (BIST) designers usually try to embed deterministic test patterns into the vector sequence generated by some linear register. Such embedding can be done either by re-seeding a TPG or modifying its feedback function [17]. These both techniques may be used in parallel, too [11]. In some other approaches, binary counters [14], or folding counters [2, 10] are used to reproduce deterministic test patterns. There are also solutions that modify or transform a vector sequence produced by a LFSR (Linear Feedback Shift Register) is such a way that it contains deterministic test patterns [1, 6, 21].

In the last decade a new linear shift register composed of T-type flip-flops in addition to D-type ones gained growing popularity [12, 13, 17]. It posses good properties as both pseudo-random [7, 8] and weighted [17] test pattern generator. Such type of linear registers is also attractive due to its low area overhead and high operating speed [7, 8], particularly if the specially designed T-type flip-flop is used [9].

In the paper an idea of deterministic TPG based on feedback shift register that is composed of D- and T-type flip-flops and inverters is developed. Its structure has a form of ring. The vector sequence produced by this TPG is modified in some clock cycles in such way that it contains deterministic test patterns only. Modification is done by nonlinear combinational function that controls, using XOR gates connected to the inputs of flip-flops, inverting of some bits of the contents of the register. Because the number of deterministic test patterns that are produced by the proposed TPG, say n-bit long, is only a small fraction of all possible n-bit vectors, the modification function does not introduce large area overhead. Proposed TPG operates in test-per-clock scheme thus it generates deterministic test patterns one after another in consecutive clock cycles.

The content of the remaining part of the paper is organized as follows. The new type of TPG called DTI-MFSR is introduced in Sect. 2. Section 3 discusses the optimization procedure for the new register, while Sect. 4 describes the evaluation tool. Experimental results are presented in Sect. 5 and we conclude in Sect. 6.

2. TPG Structure

The overall schematic of one of the proposed structures of TPG is presented in Fig. 1a. The register, which is called henceforth DTI-MFSR (Modified Feedback Shift Register composed of D- and T-type flip-flops and Inverters), takes form of a ring of connected D- and/or T-type flip-flops. Some or all of the flip-flops may have inverters at their data inputs. The value of the next state of each flip-flop can be inverted by the XOR gate, which is controlled from the output of the modifying function. Thus, in every clock cycle the content of the register can be modified in the way that we obtain one of the deterministic test patterns at the outputs of the DTI-MFSR. The exemplar structure of 3-bit DTI-MFSR is shown in Fig. 1b.

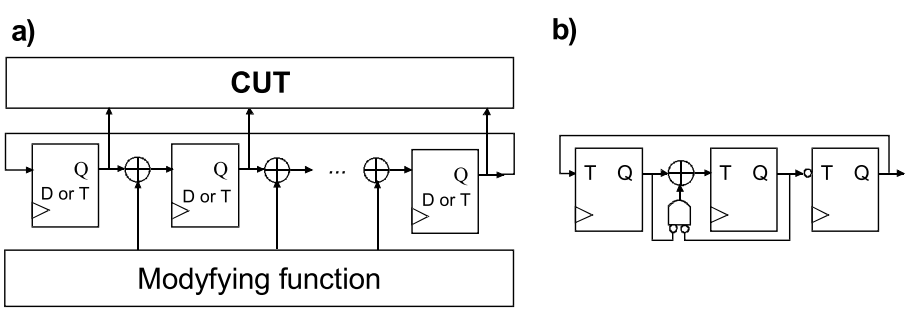


Figure 1. DTI-MFSR: a) overal schematic; b) example of three-bit register.

The main goal of design process of the DTI-MFSR is to find the order of deterministic test patterns, to select the type of each flip-flop, and to determine the flip-flops with inverted inputs in such a way that leads to minimal or quasi-minimal structure of a modifying function. The way of designing such register is shown using following example.

Let us assume that we want to design 3-bit DTI-MFSR that produces at its outputs the following set S of four vectors: V_1=100, V_2=111, V_3=001, V_4=110. At the beginning, for each ordered pair of test vectors V_i, V_j from the set S, where $i, j = 1, 2, 3, 4$ and $i \neq j$, we determine configuration of each m-th stage F_m of DTI-MFSR, where $m = 0, 1, 2$, that is able to produce this pair in two consecutive clock cycles. Some of these configurations are shown in the

form of the table of possible configurations (TPC) in Fig. 2. Symbols used in the figure have following meaning:

- D ... D-type flip-flop;
- T ... T-type flip-flop;
- $\overline{D}$ ... D-type flip-flop with inverted input;
- $\overline{T}$ ... T-type flip-flop with inverted input;

where inverting of the flip-flop input is done by the XOR gate which one input is set to value 1 by modification function. They denote four possible configurations of each stage of the DTI-MFSR. Whether certain bit transition on m-th position of the pair V_i, V_j can be realized by the given stage one can determine according the following equations, which describe previous to next state transitions of D and T flip-flops:

$$D : Q_m = q_{m-1} \qquad \overline{D} : Q_m = \overline{q}_{m-1}$$
$$T : Q_m = q_m \oplus q_{m-1} \qquad \overline{T} : Q_m = q_m \oplus \overline{q}_{m-1}$$

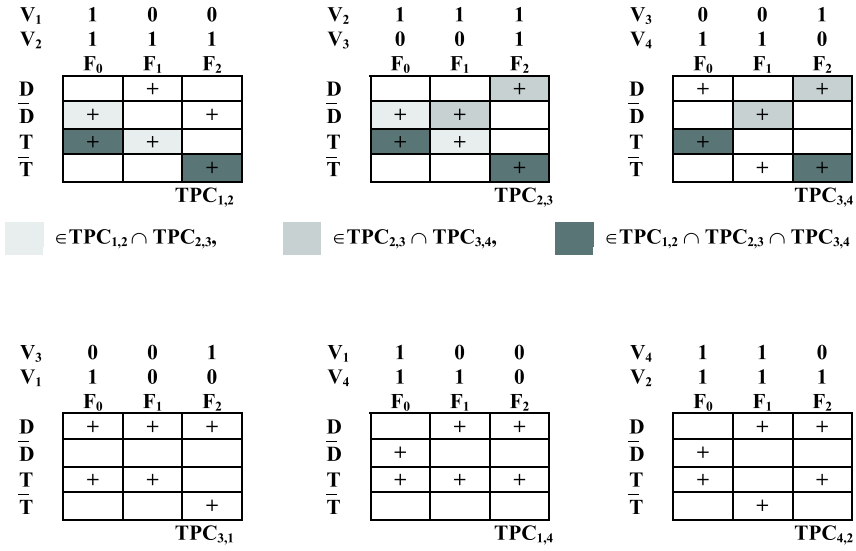


Figure 2. Some tables of possible configurations.

For example, according to the TPC1,2 in Fig. 2, the ordered pair of vectors V_1, V_2 can be produced by the DTI-MFSR which stages are configured as follows:

- stage $F_0 \ldots \overline{D}$ or T;
- stage $F_1 \ldots D$ or T;
- stage $F_2 \ldots \overline{D}$ or $\overline{T}$.

Configuration of DTI-MFSR that produces given sequence of vectors from the set S can be found on the basis of intersection of TPCs corresponding to the ordered pair of vectors that the sequence is composed of. For example, sequence of vectors V_1, V_2, V_3, V_4 is composed of three ordered pairs of vectors: $\{V_1, V_2\}$, $\{V_2, V_3\}$, $\{V_3, V_4\}$. TPCs for these pairs are shown in Fig. 2. The stage configurations that belong to particular intersections $\text{TPC}_{1,2} \bigcap \text{TPC}_{2,3}$, $\text{TPC}_{2,3} \bigcap \text{TPC}_{3,4}$ and $\text{TPC}_{1,2} \bigcap \text{TPC}_{2,3} \bigcap \text{TPC}_{3,4}$ are dashes by different graphical patterns. There is no common configuration for stage F_1 for all three TPCs. Thus, the configuration of this stage needs to be changed during the generation of the test sequence. The structure of the DTI-MFSR that produces the considered test sequence is shown in Fig. 1b. The modification function for this circuit contains one NOR gate, which detects vector V_3 in the test sequence and changes configuration of stage F_1 from T to $\overline{T}$ during vector transition V_3 to V_4. Moreover, this function has constant value 0 for stage F_0 and constant value 1 for stage F_2. Owing this fact, two XOR gates are reduced to the wire in the stage F_0 and inverter (active low input of the flip-flop) in stage F_2. The order of the vectors in the considered test sequence has been chosen arbitrarily. Thus, one can expect that some other order may lead to the cheaper modification function. The same applies to the order of bits in test vectors. Finding the optimal or nearly optimal structure of DTI-MFSR is the subject of the next section.

3. Genetic Algorithm

Traditional search and optimization methods are slow in finding the solution in a complex problem's search space. For this reason, we decided to apply the GA to find the optimal structure of the TPG. The GA is based on a heuristic method, which requires little information to search effectively in a large search space.

The GA codes parameters of the problem's search space as finite-length strings over some finite alphabet. It works with a coding of the parameter set, not the parameters themselves. The algorithm employs an initial population of strings, which evolve to the next generation by probabilistic transition rules such as selection, crossover and mutation. The objective function evaluates the quality (fitness) of solutions coded as strings. This information is used to perform an effective search for better solutions. There is no need of other auxiliary knowledge. The GA tends to take advantage of the fittest solutions

by giving them greater weight, and concentrating the search in the regions of the search space with likely improvement.

The GA is different from the traditional techniques because of its intrinsic parallelism (in evaluation function, selections) that allows working from a broad database of solutions in the search space simultaneously, climbing many peaks in parallel. Thus, the risk of converging to a local optimum is low. The random decisions made in the GA can be modeled using Markov chain analysis to show that each finite GA will always converge to its global optimum region [15]. In spite of its simplicity, the GA has proved to be an efficient method for solving various optimization and classification problems, in areas ranging from economics and game theory to control system design [16, 18, 19, 20].

3.1 Encoding

Parameters of the problem's search space were coded as integer values. We used three different chromosomes to concurrently optimize the structure of the TPG, the order of test vectors, and the order of test bit streams in the test sequence, which determines the order of columns in a test patterns set. The presentation of the first chromosome, which encodes the structure of n-bit TPG, looks like

$$C_1 = t_1 i_1 t_2 i_2 \dots t_n i_n,$$

where t_j $(j = 1, 2, \dots, n)$ represents the type of the flip-flop (either D or T) and i_j $(j = 1, 2, \dots, n)$ represents the presence of the inverter on the output of the j-th flip-fop.

The presentations of the second and third chromosome, which encodes the order of test vectors in test sequence, and the order of test bit streams in the test sequence, look like

$$C_2 = o_1 o_2 \dots o_m,$$

where m is the number of test vectors and o_j $(j = 1, 2, \dots, m)$ is the ordered number of the test vector from the vector list

$$C_3 = o_1 o_2 \dots o_k,$$

where k is the number of flip-flops in the structure and o_j $(j = 1, 2, \dots, k)$ is the ordered number of test bit streams in the test sequence.

3.2 Initial Population

The structure of the initial TPG was used to form a starting chromosome, which was reproduced $(n - 1)$-times to generate an initial population of n-strings. To ensure versatile population one quarter of the population is the same as the initial chromosome; one quarter with mirrored numbers in the chromosome (the last one on the first place, ..., the first one on the last place);

one quarter with mirrored D/T flip-flop type positions; and one quarter with mirrored positions of inverter presence.

The second and the third population (formed by the second and the third chromosome, respectively) consists of $n/2$-times reproduction of the initial chromosome and the second half of the population is made with mirrored order of test vectors and test bit streams, respectively.

3.3 Genetic Operators

To evolve the best solution candidate, the GA employed the genetic operators of selection, crossover and mutation for manipulating the strings in a population. The GA used these operators to combine the strings of the population in different arrangements, seeking a string that maximizes the objective function. This combination of strings resulted in a new population.

The first genetic operator used by the GA for creating a new generation was selection. To create two offspring two strings had to be selected from the current population as parents. Most fit strings were selected for reproduction. We had applied the elitism strategy, where a randomly selected number of least-fit members of the current population were interchanged with the equal number of the best-ranked strings.

Crossover proceeded in two steps. First, strings were mated randomly, using a given probability p_c to pair off the couples. Second, mated string couples crossed over, using a random probability to select the two-point crossing sites. An integer positions k and m were selected between 1 and the string length less one $[1, l-1]$. Swapping all characteristic values between the positions $k+1$ and m inclusively created two new strings. For example, considering two strings with $k = 1$ and $m = 4$:

$$\begin{array}{llll} D\,1 & \boxed{D\,0\;\;T\,0\;\;T\,0} & D\,1 & \\ D\,0 & \boxed{T\,1\;\;D\,1\;\;T\,0} & D\,1 & \end{array} \quad > \quad \begin{array}{llll} D\,1 & \boxed{T\,1\;\;D\,1\;\;T\,0} & D\,1 & \\ D\,0 & \boxed{D\,0\;\;T\,0\;\;T\,0} & D\,1 & \end{array}$$

Moreover, we might use a constant probability p_r to select a case, in which only the values of inverters were swapped:

$$\begin{array}{llll} D\,1 & \boxed{D\,0\;\;T\,0\;\;T\,0} & D\,1 & \\ D\,0 & \boxed{T\,1\;\;D\,1\;\;T\,0} & D\,1 & \end{array} \quad > \quad \begin{array}{llll} D\,1 & \boxed{D\,1\;\;T\,1\;\;T\,0} & D\,1 & \\ D\,0 & \boxed{T\,0\;\;D\,0\;\;T\,0} & D\,1 & \end{array}$$

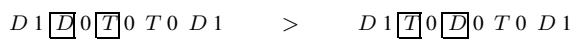
The crossover in case of test vectors order and test bit streams order was performed with the interchange of positions that store the ordered numbers within the range $[2, 4]$

$$\begin{array}{llllllll} \boxed{3} & 7 & \boxed{2} & 6 & 1 & 5 & \boxed{4} & 8 \\ 8 & 1 & \boxed{2} & 6 & \boxed{4} & 5 & \boxed{3} & 7 \end{array} \quad > \quad \begin{array}{llllllll} \boxed{2} & 7 & \boxed{4} & 6 & 1 & 5 & \boxed{3} & 8 \\ 8 & 1 & \boxed{3} & 6 & \boxed{2} & 5 & \boxed{4} & 7 \end{array}$$

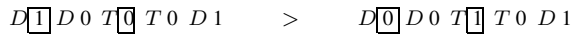
Mutation was a process by which strings resulting from selection and crossover were perturbed. It served to create random diversity in the population. Each

string was subjected to the mutation operator. Mutation was performed on characteristic-by-characteristic basis, each characteristic mutating with a probability p_m. However, since a high mutation rate resulted in a random walk through the GA search space, p_m had to be chosen to be somewhat low. Three different types of mutation were present:

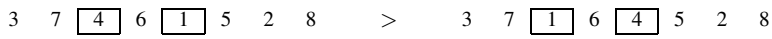
- D/T change, where only flip-flop types were changed



- inverter change, where inverter presences were changed



- order change, where pattern orders and test bit streams order were changed



There was also a possibility of annealing the mutation rate, where p_m was a variable mutation probability. It was decreasing linearly with each new population. Namely, we assumed that each new population generally was more fit than the previous one. Such an approach was used to overcome a possible disruptive effect of mutation, and to speed up the convergence of the GA to the optimal solution in the final stages of the optimization.

3.4 Fitness Evaluation

Following selection, reproduction, crossover, and mutation, the new population was ready to be evaluated. Therefore, each new string (solution candidate) created by the GA was evaluated by the external evaluation tool (see Sect. 4).

3.5 Termination Criteria

The GA operated repetitively, with an idea that, on average, solutions of the population defining the current generation had to be as good (or better) at maximizing the fitness function as those of the previous generation. When a certain number of populations had been generated and evaluated, the system was assumed to be in a non-converging state. The fittest member within all generations was taken to be the solution of the design problem.

4. Evaluation Tool

Operation of the jth cell of the DTI-MFSR register during one clock cycle can be expressed by the following equation:

$$Q_j = t_j q_j \oplus q_{j-1} \oplus i_j \oplus f_j$$

$$Q_1 = t_1 q_1 \oplus q_n \oplus i_1 \oplus f_1$$

where q_{j-1} is the current state of the cell number $j-1$, q_j is the current state of the jth cell, Q_j is the next state of the jth cell, t_j is the coefficient determining type of the flip-flop in the jth cell ($0 \ldots D$ flip-flop, $1 \ldots T$ flip-flop), i_j is the coefficient determining whether there is a negation at the input of the flip-flop in the jth cell ($0 \ldots$ no negation, $1 \ldots$ there is negation), and f_j is the value of the jth output of the modifying function.

Thus, the value of the jth output of the modifying function is:

$$f_j = t_j q_j \oplus q_{j-1} \oplus i_j \oplus Q_j$$

$$f_1 = t_1 q_1 \oplus q_n \oplus i_1 \oplus Q_1$$

On the basis of these equations one can derive values of the outputs of the modifying function for each vector but last in the test sequence. In that way ON-set and OFF-set of the modifying function are defined. They are further minimized and the cost of the function is estimated. We use ESPRESSO software [5] for boolean minimization of the modifying function and its approximate cost evaluation.

5. Results

The evaluation process for our optimization procedure was the following:

- preparation of the initial structure;
- evolutionary optimization of the structure, with the use of evaluation tool;
- the implementation of the optimal solution in HDL (hardware description language), and its synthesizing.

Table 1 presents the results of the evaluation (area estimation) of our optimization process with the following ISCAS test-benchmark combinatorial circuits:

- c432 ... 27-channel interrupt controller;
- c499 ... 32-bit SEC circuit;
- c880 ... 8-bit ALU;
- c6288 ... 16×16 multiplier.

The used test sets were transformed by compatible input reduction procedure [3], so test pattern width and number of test patterns are presented in second and third column, respectively, for each benchmark. The other columns of the table presents the number of gates needed for the implementation of the TPG structure (numbers are produced by the Synopsys synthesis tool using $0.35\mu m$

technology). The fourth column (Initial TPG) presents the results of the initial structures, while the fifth column (TPG) presents the results of the optimized structures.

We decided to use deterministic TPG having a form of ROM memory as a reference solution to compare with, since ROM based deterministic TPG is a classic type of deterministic TPG [4]. The memory sizes are estimated on the basis of data provided on the web for CMOS 0.35 μm technology [22]. Again results are presented in the form of equivalent gates (sixth column - ROM).

Table 1. Results (in equivalent gates) of TPG structure optimization for the ISCAS benchmarks.

	Test pattern width	*Number of test patterns*	*Initial TPG*	*TPG*	*ROM*
c432	36	27	340.6	335.6	517.8
c499	41	52	649.8	620.6	1135.7
c880	60	16	503.6	477.6	511.4
c6288	11	12	84.4	71.4	70.3

As presented in Table 1, our evolutionary TPG structure optimization approach mostly makes better results as it is in the case of ROM memory based TPG.

6. Conclusions

A new type of a deterministic TPG called DTI-MFSR is presented in the paper. It is based on a feedback shift register composed of D- and T-type flip-flops and inverters. It is also equipped with a non-linear combinational function that can invert, using XOR gates connected to the inputs of the flip-flops, any bit in any pattern generated by the register. Thus, any arbitrary test sequence can be produced at the outputs of the new TPG. The optimization algorithm, which for the given deterministic test set minimizes the area overhead of the DTI-MFSR generating this set of vectors, is also proposed in the paper. The optimization is based on genetic algorithm. The initial structure of the TPG is encoded and multiplied with some variations to form the initial population. The search for the optimal structure of the TPG is performed by selection, crossover, and mutation operators, while each solution is evaluated by the evaluation tool. The results show that our evolutionary TPG structure optimization approach makes better results as we get if we consider the initial structures of TPGs.

Acknowledgment

The work presented in the paper was supported by the Bilateral Polish-Slovenian project *Development of on-line built-in self-test structures, BI-PL/03-04-012*.

References

[1] M. Bellos, D. Kagaris, and D. Nikolos. Test Set Embedding Based on Phase Shifters. In *Proceedings of The Fourth European Dependable Computing Conference, EDCC-4*, Toulouse, France, 2002, pp. 90–101.

[2] K. Chakrabarty, B.T. Muray, and V. Iyengar. Deterministic Built in Test Pattern Generation for High Performance Circuits Using Twisted Ring Counters. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 8:633–636, 2000.

[3] C-A. Chen and K. Gupta. Efficient BIST TPG Design and Test Set Compaction via Input Reduction, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 17:692–705, 1998.

[4] G. Edirisooriya and J.P. Robinson. Design of Low Cost ROM Based Test Generators. In *Proceedings of IEEE VLSI Test Symposium*, Atlantic City, NJ, 1992, pp. 61–66.

[5] Espresso, Version # 2.3, Release date 01/31/88, UC Berkeley, http://www-cad.eecs.berkeley.edu:80/Software/software.html

[6] P. Fišer and J. Hlavička. Column Matching Based BIST Design Method. In *Proceedings of IEEE European Test Workshop*, Athens, Greece, 2000, pp. 15–16.

[7] T. Garbolino and A. Hlawiczka. Design of Test Pattern Generator Using a New LFSR with D and T Flip Flops. In *Handouts of IEEE European Test Workshop*, Constance, Germany, 1999.

[8] T. Garbolino and A. Hlawiczka. A New LFSR with D and T Flip Flops as an Effective Test Pattern Generator for VLSI Circuits. *Lecture Notes in Computer Science*, 1667:321–338, 1999.

[9] T. Garbolino, A. Hlawiczka, and A. Kristof. Fast and Low Area TPG based on T type Flip flops can be Easily Integrated to the Scan Path. In *Proceedings of the IEEE European Test Workshop*, Cascais, Portugal, 2000, pp. 161–166.

[10] S. Hellebrand, H.G. Liang, and H.J. Wunderlich. A Mixed Mode BIST Scheme Based on Reseeding of Folding Counters. In *Proceedings of International Test Conference 2000*, Atlantic City, NJ, 2000, pp. 778–784.

[11] S. Hellebrand, J. Rajski, S. Tarnick, S. Venkataraman, and B. Courtois. Built-in Test for Circuits with Scan Based on Reseeding of Multiple-Polynomial Linear Feedback Shift Registers. *IEEE Transaction on Computers*, 44:223–233, 1995.

[12] A. Hlawiczka. D or T Flip Flop Based Linear Registers. *Archives of Control Sciences*, 1:249–268, 1992.

[13] A. Hlawiczka. *Linear Registers - Analysis, Synthesis and Applications in Digital Circuits Testing*. Silesian Technical University Press, Gliwice, 1997.

[14] D. Kagaris and S. Tragoudas. Generating Deterministic Unordered Test Patterns with Counters. In *Proceedings of the 14th VLSI Test Symposium*, Princeton, NJ, 1996, pp. 374–379.

[15] C.L. Karr, I. Yakushin, and K. Nicolosi. Solving inverse initial-value, boundary-value problems via genetic algorithm. *Engineering Applications of Artificial Intelligence*, 13:625–633, 2000.

[16] B. Koroušić-Seljak. Heuristic methods for a combinatorial optimization problem - real-time task scheduling problem. In *Proceedings of the Artificial Networks in Engineering Conference (ANNIE'99)*, St. Louis, USA, 1999, pp. 1041–1046.

[17] O. Novak. Pseudorandom, Weighted Random and Pseudoexhaustive Test Patterns Generated in Universal Cellular Automata. *Lecture Notes in Computer Science*, 1667:303–327, 1999.

[18] G. Papa, B. Koroušić-Seljak, B. Benedičič, and T. Kmecl. Universal motor efficiency improvement using evolutionary optimization. *IEEE Transactions on Industrial Electronic*, 50:602–611, 2003.

[19] G. Papa and J. Šilc. Automatic Large-Scale Integrated Circuit Synthesis Using Allocation-Based Scheduling Algorithm. *Microprocessors and Microsystems*, 26:139–147, 2002.

[20] G. Papa and J. Šilc. Using Simulated Annealing and Genetic Algorithm in the Automated Synthesis of Digital Systems. In *Recent advances in Circuits and Systems*, World Scientific, Singapore, 1998, pp. 377–381.

[21] N.A. Touba and E.J. McCluskey. Bit-Fixing in Pseudorandom Sequences for Scan BIST. *IEEE Transactions on Computer-Aided Design of Integrated Circuits And Systems*, 20:545–555, 2001.

[22] Memory Compiler for diffusion programmable ROM in 0.35µm CMOS. http://asic.austriamicrosystems.com/databooks/digital/mc_rom_c35.html

EVOLUTIONARY PRODUCTION SCHEDULING IN CAR MANUFACTURE

Bogdan Filipič
Department of Intelligent Systems
Jožef Stefan Institute, Ljubljana, Slovenia
bogdan.filipic@ijs.si

Abstract The paper describes an evolutionary computation approach to solving real-world scheduling problems on a group of production lines in a car factory. The cost of schedules is determined by the quality of energy consumption management over peak demand periods, and the task is to minimize the energy costs by appropriate scheduling of process interruptions. Systematic tests of the evolutionary scheduling system on problems faced daily at the plant confirm the system is capable of finding near-optimal schedules in a reasonable period of time.

Keywords: Evolutionary algorithm, Production scheduling, Energy consumption management, Automobile industry

1. Introduction

Scheduling is concerned with allocating activities to resources over time in such a way that given objectives are optimized, while temporal constraints and resource limitations are satisfied. Problems of this type appear in manufacturing, timetabling, vehicle routing, design of computer operating systems and other domains. Because of its great practical importance, scheduling has permanently attracted research interests. Following the attempts in the fields of Operations Research and Artificial Intelligence with limited success in practice, Evolutionary Computation [1] has recently offered means of generating near-optimal schedules for complex problems at reasonable computational costs [3]. A number of applications of evolutionary algorithms in scheduling have been reported [2, 6, 10]. However, there are still challenging issues to be considered in the development of evolutionary scheduling systems. Above all, real-world problems should be dealt with and realistic criteria for schedule optimization taken into account [5].

Using evolutionary computation techniques, we deal with a class of real-world problems with schedule cost related to resource management. Our previ-

ous application oriented studies include scheduling of operations in a production unit of a textile factory, where the objective was to ensure optimal energy consumption [7], and scheduling activities in ship repair in order to balance the work load for workers of various trades [8]. In this paper we describe production scheduling on a group of production lines of an automobile factory. The objective is to schedule interruptions of the running processes in such a manner that energy consumption over the peak demand periods is minimized. In addition, the schedules are subject to time and resource constraints that have to be strictly satisfied.

Design of an evolutionary scheduling system for this problem and the initial practical results were presented in [9], while here the focus is on an improved version of the system and its evaluation. The paper explains the scheduling problem and the schedule cost that is related to energy consumption, describes the employed scheduling system, provides the results of its evaluation on real problem instances, and discusses them in view of regular exploitation at the plant.

2. The Scheduling Problem

Production systems relying on intense energy consumption, such as steel plants and other heavy industries, are faced with peak demand periods. These are the time periods over which their power demand exceeds a given limitation and the excess has to be paid at a higher rate. This measure is imposed by the energy supplier to minimize the total energy consumption over critical periods. There are several ways of reducing the peak power demand: activation of internal energy sources, interruption of energy-intensive processes, and appropriate production scheduling.

The focus of energy consumption management in the considered factory is in the car-body production unit. The unit consists of six lines of hydraulic presses that perform cutting and shaping. A line in operation is regarded as an individual work process. Power demands of the processes vary from 20kW to 370kW. The unit operates according to a daily production plan that specifies which of the lines are in operation and what is their work time. Power demand of the unit equals to the sum of power demands of the running processes. Other energy consumers at the plant contribute to the so called background power demand, P_b. The total power demand of the plant, P, consists of the demand of the pressing unit and the background demand. To asses the energy costs, the total demand is related to the prescribed limitation $P_{\max}$, also called the target load. Figure 1 shows an example of daily profiles for background demand, total power demand and target load.

The efforts to reduce the target load excess are concentrated on line production in the pressing unit, since it is an intense energy consumer and also more

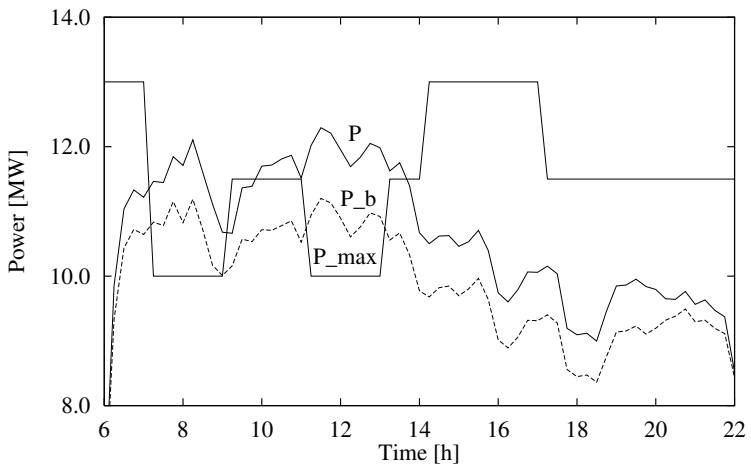


Figure 1. Power profiles: background demand P_b, total power demand P, and target load $P_{\max}$

suitable for scheduling than background processes. Two approaches are combined in the unit: process interrupting and scheduling. Process interruptions are either intended as breaks for the staff or can be spent to change machine tools and perform maintenance on the lines. The idea is to schedule these activities in such a way that the daily production plan is realized, while the contribution of the unit to the target load excess is minimized.

To balance between the conflicting requirements of plan fulfilment and reduction of the target load excess, the following constraints have been imposed on schedules:

- duration of process interruptions, T_0,
- minimum period of time between two interruptions of a process, T,
- maximum number of processes that can be interrupted simultaneously, M.

Taking into account these constraints, process interruptions have to be scheduled so as to minimize the target load excess contributed by the production lines.

2.1 Schedule Cost

The schedule cost to be minimized is formally defined as follows. Let $P_i(t)$, $i = 1, \ldots, N$, denote the power demands of the operating production

lines in the pressing unit. The total power demand of the plant is

$$P(t) = \sum_{i=1}^{N} P_i(t) + P_b(t) \tag{1}$$

where $P_b(t)$ represents the background demand. Then the contribution of the considered processes to the target load excess at time t is

$$P_{\text{exc}}(t) = \begin{cases} \sum_{i=1}^{N} P_i(t); & P_b(t) \geq P_{\max}(t) \\ P(t) - P_{\max}(t); & P_b(t) < P_{\max}(t) \;\&\; \\ & P(t) > P_{\max}(t) \\ 0; & \text{otherwise} \end{cases} \tag{2}$$

and the energy consumption resulting from the target load excess equals to

$$W_{\text{exc}} = \int_t P_{\text{exc}}(t)\, dt. \tag{3}$$

W_{exc} represents the cost of interruption schedules which is to be minimized. It is to be noted, however, that power demands are in practice sampled using certain time interval Δt and integral (3) is approximated by

$$\sum_t P_{\text{exc}}(t)\, \Delta t. \tag{4}$$

3. The Evolutionary Scheduling System

The scheduling system generates daily schedules of process interruptions and calculates the expected reduction of the target load excess. It accepts the following input information:

- estimates of power demand profiles for the processes to be executed,
- an estimate of the background demand profile,
- the target load profile, and
- constraints to be considered in schedule construction.

The power demand estimates are based on data recorded over previous days and on production plan for the current day. As the production does not change rapidly, the estimates are rather accurate and make it possible to generate realistic production schedules.

The scheduling algorithm is designed to solve problem instances with arbitrary power demand profiles and can operate at various time discretizations.

The core of the algorithm is a $(\mu + \lambda)$ evolution strategy [13]. It iteratively improves the schedules through the following sequence of steps:

Step 0: Generate an initial population of μ schedules by randomly assigning starting times to process interruptions.

Step 1: Generate λ descendants from μ parents by applying local transformations to schedules.

Step 2: Select μ best solutions out of $\mu + \lambda$ available, and make them parents for the next generation.

Step 3: If maximum number of generations is reached, exit, otherwise go to Step 1.

The best schedule found during this search process is returned as a suboptimal solution to the problem. Both, inserting the interruptions into a schedule at the initialization step and local schedule transformations are performed in such a way that constraints imposed on schedules remain satisfied. This is achieved by maintaining a direct representation of schedules within the algorithm and checking the constraints. Schedules are represented as two-dimensional arrays with the number of rows equal to the number of running processes, and columns to time intervals considered during scheduling. Each element of the array holds a value denoting the process status at the corresponding time interval. The status can be: *interrupted*, which means the process is interrupted, *interruption possible*, which means the process is running and it is possible to interrupt it, or *interruption not possible*, which means the process is running but cannot be interrupted due to the constraints.

Schedule transformations are carried out on random basis and include:

- inserting an interruption into a schedule,
- deleting an interruption from a schedule,
- shifting an interruption within a schedule.

Insertion of an interruption consists of finding a random time slot in the schedule with *interruption possible*, changing its status to *interrupted* and updating the status of the slots affected through constraint values T and M to *interruption not possible*. Deletion of an interruption includes random selection of an *interrupted* slot, changing its status to *interrupt possible*, and updating the status of the slots that are no more effected through constraint values T and M to *interruption possible*. Shifting of an interruption consists of its deletion at the current time slot and insertion at another time slot.

4. Evaluation of the Scheduling System

4.1 Initial Tests on Real Problem Instances

The scheduling system was initially tested on a set of problem instances based on real data recorded at the plant. The data were used as input to optimize daily schedules for the production lines. The constraints for schedule construction were set as follows. Duration of process interruptions, T_0, was 30 minutes. Each process had to run continuously for at least four hours between two interruptions ($T = 240$ min), and at most three process interruptions were permitted to take place simultaneously ($M = 3$). Time step used during search for assigning starting times to interruptions was 5 minutes.

The scheduling algorithm was run for 200 generations. The population size and the number of offspring generated in each generation were $\mu = \lambda = 20$. For each problem instance, the algorithm was executed 10 times, and both the best and average results were recorded. The optimized schedules of process interruptions were produced in the form shown in Table 1.

Table 1. An example of the optimized interruption schedule for the production lines.

Line number	*Number of interruptions*	*Interruption times*	
1	2	8:00–8:30	12:15–12:45
2	2	7:25–7:55	11:55–12:25
3	2	7:00–7:30	12:20–12:50
4	2	7:15–7:45	11:45–12:15
5	1		11:00–11:30
6	1		11:30–12:00

The evaluation confirmed that schedule optimization can substantially contribute to the decrease of energy costs in the production unit. Energy consumption resulting from the target load excess on the lines was reduced by at least 25% on workdays, but in most cases by about 30%. Table 2 shows the achieved reduction averaged over 10 runs of the optimization algorithm for each day in a two-week period. The reproducibility of the reduction in kWh obtained in 10 runs for each problem instance was within 2%.

4.2 Testing the Optimality of Schedules

Additional numerical experiments were carried out to check how close the optimized schedules are to the true optimal ones. For this purpose a selected scheduling problem representing a typical situation at the plant was used. All six production lines were required to operate and estimates of power demand profiles shown in Fig. 1 were used. If case of no process interruptions, the target load excess would amount to 3218.3 kWh. For this power demand situation,

Table 2. Average reduction of the target load excess on the production lines obtained by the evolutionary scheduling system.

	Target load excess	*Reduction*	
Day	[kWh]	[kWh]	[%]
Mon	2616.5	1000.4	38.2
Tue	2569.6	970.5	37.8
Wed	3218.3	1012.6	31.5
Thu	2892.2	926.2	32.0
Fri	3055.1	931.6	30.5
Sat	655.0	413.0	63.0
Sun	0.0	0.0	0.0
Mon	2461.2	810.1	32.9
Tue	2117.7	636.6	30.1
Wed	2910.3	836.8	28.7
Thu	2752.8	803.3	29.2
Fri	2523.5	869.8	34.5
Sat	0.0	0.0	0.0
Sun	0.0	0.0	0.0

test problem instances of various complexity were defined. Their complexity was varied through constraint values T_0, T, and M.

To denote a problem instance with particular constraint values, will use the notation (T_0, T, M), where times T_0 and T are given in minutes, and $M \in [1..N]$. The test set of problem instances consisted of $(30, 240, 3)$, $(30, 240, 1)$, $(30, 120, 3)$, $(30, 120, 1)$. Note that $(30, 240, 3)$ is the default setting of constraint values used at the plant, while additional settings resulting in more demanding problems were chosen to further check the performance of the developed scheduling system.

For the evaluation purposes, optimal schedules for the selected problem instances were produced by the Constraint Logic Programming approach. Constraint Logic Programming (CLP, [4, 11]) is a generalization of logic programming [12] where unification is replaced by a more general mechanism of constraint satisfaction over a specific computation domain, such as Boolean, finite or real. It is capable of finding optimal solutions to the problems of manageable size. We used the ECL^{i}PS^e CLP environment and its finite domain solver CLP($\mathcal{F}$). Unfortunately, the scheduling problem introduced in Section 2 is too complex to be treated generally. However, particular problem instances can be handled individually by considering their specificities during problem solving.

Schedule costs found by this tool and by the evolutionary scheduling system are compared in Table 3. More clear picture of the evolutionary algorithm performance can be obtained from Table 4 which shows the deviation of the schedule improvement from the optimum gained with CLP.

Table 3. Optimal schedule costs found with CLP and suboptimal costs obtained with the evolutionary algorithm (EA); the results are in kWh.

Problem instance (T_0, T, M)	*CLP*	*EA*	
		best	*average*
(30, 240, 3)	2209.1	2211.5	2213.1
(30, 240, 1)	2374.3	2385.1	2419.7
(30, 120, 3)	2185.8	2187.1	2187.4
(30, 120, 1)	2345.8	2365.8	2390.4

Table 4. Deviation of schedule cost improvement by the EA from the optimal improvement obtained with CLP.

Problem instance (T_0, T, M)	*EA*	
	best	*average*
(30, 240, 3)	0.2%	0.4%
(30, 240, 1)	1.3%	5.4%
(30, 120, 3)	0.1%	0.2%
(30, 120, 1)	2.3%	5.1%

These results are very informative for practical assessment of the evolutionary scheduling algorithm. While the initial tests on real problems showed that potential decrease of energy costs is at the expected level, we now have an absolute measure of the scheduling algorithm performance. It is particularly encouraging, that under constraint setting $(30, 240, 3)$, which is usually used at the plant, the result is very close to the optimum. For hypothetical problems with more complex spaces the gap to the optimum increases, but we believe that initial results given in this paper can still be improved.

Certainly, one may ask whether it is possible to apply the CLP system for regular scheduling at the plant. It turns out that its advantage of guaranteed optimal solutions comes at some other costs. Solving problems of this type with CLP is only efficient on individual basis, where additional constraints for schedules are derived from input data (e.g. feasible time intervals for process interruptions) and implemented to prune the search space. Further increase of problem complexity would sooner or later exceed the capabilities of the system. The CPU time spent to obtain optimal solutions with the CLP system depends very much on the problem instance, and ranges from a few minutes to several hours on an ordinary Pentium computer. On the other hand, the execution of the evolutionary algorithm on the same computer requires about half a minute for each problem instance, and only slightly increases with problem complexity.

5. Conclusions

An evolutionary algorithm was developed to schedule process interruptions on car-body production lines in an automobile factory where the objective is to decrease power demand over critical periods. In a comparative study the results of the evolutionary algorithm were assessed with regards to optimal results found by a CLP system. The comparison was beneficial in that it confirmed the evolutionary algorithm is capable of finding near-optimal results for a typical scheduling task appearing on the lines.

The approach was implemented as a process scheduling module within a system for energy consumption management at the plant. The role of the scheduling module is to assist the process supervisor in preparing daily schedules for the pressing unit. Through its application it is possible to analyse the impact of various production plans and constraint settings on energy consumption. Most important, its regular exploitation significantly contributes to the decrease of production costs at the plant.

Acknowledgment

This work was supported by the Slovenian Ministry of Education, Science and Sport, and by the companies Revoz and INEA.

References

[1] T. Bäck, D. B. Fogel, and Z. Michalewicz (eds.). *Handbook of Evolutionary Computing*. Oxford University Press, New York, and Institute of Physics Publishing, Bristol, 1997.

[2] J. Biethahn and V. Nissen (eds.). *Evolutionary Algorithms in Management Applications*. Springer-Verlag, Berlin, 1995.

[3] R. Bruns. Scheduling. In T. Bäck, D. B. Fogel, and Z. Michalewicz (eds.): *Handbook of Evolutionary Computing*. Oxford University Press, New York, and Institute of Physics Publishing, Bristol, 1997, Chapter F1.5.

[4] J. Cohen. Constraint logic programming languages. *Communications of the ACM*, 33(7):52–68, 1990.

[5] D. Corne and P. Ross. Practical issues and recent advances in job- and open-shop scheduling. In D. Dasgupta and Z. Michalewicz (eds.): *Evolutionary Algorithms in Engineering Applications*, Springer-Verlag, Berlin, 1997, pp. 531–546.

[6] D. Dasgupta and Z. Michalewicz (eds.). *Evolutionary Algorithms in Engineering Applications*. Springer-Verlag, Berlin, 1997.

[7] B. Filipič. Enhancing genetic search to schedule a production unit. In B. Neumann (ed.): *Proceedings of the 10th European Conference on Artificial Intelligence ECAI '92*, Vienna, Austria, 1992, pp. 603–607. Also published in J. Dorn and K.A. Froeschl (eds.): *Scheduling of Production Processes*, Ellis Horwood, Chichester, 1993, pp. 61–69.

[8] B. Filipič. A genetic algorithm applied to resource management in production systems. In J. Biethahn and V. Nissen (eds.): *Evolutionary Algorithms in Management Applications*, Springer-Verlag, Berlin, 1995, pp. 101–111.

[9] B. Filipič. A hybrid optimization algorithm for energy consumption management at a motor plant. In *Proceedings of the 5th European Congress on Intelligent Techniques and Soft Computing EUFIT'97*, Aachen, Germany, 1997, vol. 1, pp. 717–721.

[10] K.S. Hindi, H. Yang, and K. Fleszar. An Evolutionary Algorithm for Resource-Constrained Project Scheduling. *IEEE Transactions on Evolutionary Computation*, 6(5):512–518, 2002.

[11] J. Jaffar and J.-L. Lassez. Constraint logic programming. In *Proceedings of the 14th ACM Symposium on Principles of Programming Languages*, Munich, Germany, 1987, pp. 111–119.

[12] J.W. Lloyd. *Foundations of Logic Programming*. Springer-Verlag, Berlin, 1987.

[13] H.-P. Schwefel. *Evolution and Optimum Seeking*. John Wiley, New York, 1995.

EVOLUTIONARY OPTIMIZATION OF A ROBUST CONTROLLER FOR FLIGHT MANEUVERS

Salvatore D'Angelo
Department of Aeronautical and Space Engineering
Politecnico di Torino, Turin, Italy
salvatore.dangelo@polito.it

Edmondo Minisci
Department of Aeronautical and Space Engineering
Politecnico di Torino, Turin, Italy
edmondo.minisci@polito.it

Marco Dutto
Department of Aeronautical and Space Engineering
Politecnico di Torino, Turin, Italy
dutto_marco@liberto.it

Abstract The determination of optimal and robust controllers for the longitudinal maneuvers of a modern fighter is the aim of this work. For the constrained multi-objective optimization process both a classical evolutionary algorithm and a newer estimation of distribution one have been used. Obtained results have been commented and discussed. In short, evolutionary approach (both algorithms) gives the advantage of handling the non-linearities due to the H_∞ norm computation and allows for a true multi-objective optimization.

Keywords: Evolutionary optimization, Multi-objective optimization, Estimation of distribution algorithm, Robust control, Flight maneuvers

1. Introduction

Within the aerospace industry there is a large amount of interest in flight control system design. The controller design is a complex problem and, for a general model, it is characterized by a number of fundamental trade-offs between conflicting design specifications. For a typical aircraft systems we can recognize several classes of criteria, for instance, among the others, *performance criteria*,

which reflect tracking error and disturbance rejection characteristics of certain signals, and *robustness criteria*, which reflect the stability bounds with respect to parameter variations.

Over the past decades H_∞ control theories have been well developed and widely applied to the problems of control system design, the focus of which is often on tracking and disturbance rejection in the low-frequency region and robustness to unmodelled dynamics in the high-frequency region, although the problem specifications may in some cases determine otherwise. The former is usually captured by minimizing the closed-loop sensitivity functions at low-frequencies. The latter is usually captured minimizing the closed-loop complementary sensitivity functions at high-frequency. But the controlled system should have good responses to reference signals on the time domain as well.

Finding control gains that minimize or maximize a designated cost function in a mixed time and/or frequency domain subject to multiple constraints specified by mixed time and/or frequency domain specifications is a complex, constrained optimization problem that cannot be solved analytically. Moreover, this kind of problem is characterized by a number of non commensurate design objectives that must all be obtained for the solution to be satisfactory and, due to the nature of trade-offs involved, the problem rarely has a unique solution. Instead, a set of equally admissible solutions are sought from an appropriately formulated optimization problem.

In this paper, we consider the application of two different evolutionary algorithms (EAs) to the design of a flight controller for a simplified fighter model. After a description of the used algorithms (detailed for the estimation of distribution one), few words are spent to introduce the requirements to obtain in order to have robustness characteristics for a generic system. After, the optimization setting is detailed and achieved results are shown and commented. A conclusion section summarizes the work and introduces future applications and extensions.

2. Estimation of Distribution Algorithms

The extensive use of evolutionary algorithms in the last decade demonstrated that an optimization process can be obtained by combining effects of interactive operators such as selection - whose task is mainly to identify the best individuals in the current population - and crossover and mutation, which try to generate new and better solutions starting from the selected ones. But, if the mimicking of natural evolution in living species has been a source of inspiration of new strategies, the attempt to copy natural techniques as they are sometimes introduces a great complexity without a corresponding improvement of algorithms performance. Moreover standard evolutionary algorithms can be ineffective when problems exhibit a high level of interaction among variables.

This is mainly due to the fact that recombination operators are likely to disrupt promising sub-structures of optimal solutions.

In order to make a rational use of the evolutionary metaphor and/or to create optimization tools that are able to handle very hard problems (with several parameters, with difficulties in linkage learning, deceptive), some algorithms have been proposed that automatically learn the structure of the search space. Instead of implicit reproduction of important building blocks and their mixing by selection and two-parent recombination operators, new solutions are generated by using the information extracted from the entire set of promising solutions.

Generally, these methods, starting from results of current populations, try to identify a probabilistic model of the search space, and crossover and mutation operators are replaced with sampling. These methods have been named Estimation of Distribution Algorithms (EDAs).

Most EDAs have been developed to manage optimization processes for single-objective, combinatorial problems, but several works regarding problems in continuous domains have been proposed as well.

We can distinguish three types of EDAs depending on the way the probabilistic model is built: a) without dependencies among variables, with bivariate dependencies among variables, and c) with multivariate dependencies (more information about EDAs can be found in [6]).

In this work we used both a conventional and well known NonDominated Sorting Genetic Algorithm-II (NSGA-II) and a home made EDA. After a brief introduction to the NSGA-II, as detailed in [3], few words will be spent to describe the used multi-objective EDA named MOPED.

2.1 NSGA-II

In NSGA-II the dominance depth is used to classify the population and a *crowding parameter* is determined in order to rank the individuals inside each class of dominance.

For each element of a class, the crowding parameter is obtained as the sum of the difference of the cost functions of the nearest elements in the cost function space, divided by the range spanned by the population with respect to each objective function. Inside each class, the individuals with the higher value of the crowding parameter obtain a better rank than those with a lower one, forcing to explore the Pareto front.

The whole algorithm (the unconstrained version) can be detailed as follows. Initially, a random parent population P_0 is created. The population is sorted based on the non-domination. Each solution is assigned a fitness equal to its non-domination level (1 is the best level). Thus, minimization of fitness is assumed. Binary tournament selection, recombination, and mutation operators are used to create a child population Q_0 of size n_{ind}. From the first generation

onward, the procedure is different. First, a combined population $R_t = P_t \cup Q_t$ is formed. The population R_t will be of size $2n_{ind}$. Then, the population R_t is sorted according to non-domination. The new parent population P_{t+1} is formed by adding solutions from the first front till the size exceeds n_{ind}. Thereafter, the solutions of the last accepted front are sorted according to the crowded comparison operator and the first n_{ind} points are picked. This is how we construct the population P_{t+1} of size n_{ind}. This population of size n_{ind} is now used for selection, crossover and mutation to create a new population Q_{t+1} of size n_{ind}.

In this work the real-coded, constrained version (simulated binary crossover, SBX, and polynomial mutation) has been used. The adopted constraint handling technique follows the dictates of the *constraint-domination principle* [3], which discriminates between unfeasible and feasible solution during the non-dominated sorting procedure. The definition of constrained-domination is: a solution i is said to constrained-dominate a solution j, if (a) solution i feasible and solution j is unfeasible; (b) or solution i and j are both unfeasible, but solution i is nearer to the constraint boundary; (c) or both solutions are feasible and i dominates j.

2.2 MOPED Algorithm

The MOPED (Multi-Objective Parzen based Estimation of Distribution) is a multi-objective optimization algorithm for continuous problems that uses the Parzen method to build a probabilistic representation of Pareto solutions, with multivariate dependencies among variables.

Similarly to what was done in [5] for multi-objective Bayesian Optimization Algorithm (BOA), some techniques of NSGA-II are used to classify promising solutions in the objective space, while new individuals are obtained by sampling from the Parzen model. NSGA-II is identified as a promising base for the algorithm mainly because of its intuitive simplicity coupled with brilliant results on many problems.

The Parzen method [4] pursues a non-parametric approach to kernel density estimation and it gives rise to an estimator that converges everywhere to the true Probability Density Function (PDF) in the mean square sense. Should the true PDF be uniformly continuous, the Parzen estimator can also be made uniformly consistent. In short, the method allocates exactly n_{ind} identical kernels, each one "centered" on a different element of the sample.

Main differences between MOPED and NSGA-II are due to the classification and search techniques. For this reason only these parts of MOPED are detailed here ([1, 2]).

2.2.1 Classification & Fitness Evaluation. The individuals of the population are classified in a way that favors the most isolated individuals in the

objective function space, in the first sub-class (highest dominance) of the first class (best suited with respect to problem constraints).

The first step in the evaluation of the fitness parameter is given by the determination of the degree of compatibility of each individual with the problem constraints, the overall number of which is given by m. This compatibility is measured by a constraint parameter, cp, which is a weighted sum of every unsatisfied constraint. Once the value of cp is evaluated for all the individuals, the population is divided in a predetermined number of classes, $1 + N_{cl}$. The n_{best} individuals that satisfy all the constraints ($cp = 0$) are in the first class. The remainder of the population is divided in the other groups, each one containing an approximately equal number of individuals, given by $(n_{ind} - n_{best})/N_{cl}$. The second class is formed by those individuals with the lower values of the constraint parameter and the last one by those with the highest values. For each class, individuals are ranked in terms of dominance criterion and crowding distance in the objective function space, using the NSGA-II techniques.

After that all the individuals of the population have been ranked, from the best to the worst one, as a consequence of their belonging to a given class and subclass, and the value of their crowding parameter, a fitness value, linearly varying from $2 - \alpha$ (best individual of the entire population) to α (worst individual), with $\alpha \in [0, 1)$, is assigned to each individual. This fitness value determines the weighting of the kernel for the sampling of the next generation individuals.

2.2.2 Building the Model and Sampling. On the basis of the information given by n_{ind} individuals of the current population, by means of the Parzen method, a probabilistic model of the promising search space portion is built (see previous section). Then, τn_{ind} new individuals, with $\tau \geq 1$ are sampled, using the probabilistic model just determined. The variance associated to each kernel depends on (i) the distribution of the individuals in the search space and (ii) on the fitness value associated to the pertinent individual, so as to favor the sampling in the neighborhood of the most promising solutions. For generic processes it can be useful to alternatively adopt different kernels from a generation to the other, in order to obtain an effective exploration of the search space.

3. Robust Control

Consider a control system as shown in Fig. 1, where $\boldsymbol{P}_0(s)$ is an n_i-inputs and n_o-outputs nominal plant, $\Delta\boldsymbol{P}(s)$ is the plant perturbation, $\boldsymbol{C}(s)$ is the controller, $\boldsymbol{r}(s)$ is the reference input, $\boldsymbol{u}_p(s)$ is the control input, $\boldsymbol{e}(s)$ is the tracking error, $\boldsymbol{d}(s)$ is the external disturbance, and $\boldsymbol{y}(s)$ is the output of the system. Without loss of a generality, the plant perturbation $\Delta\boldsymbol{P}(s)$ is assumed to be bounded by a known stable function matrix $\boldsymbol{W}_1(s)$, i.e. $\bar{\sigma}\left(\Delta\boldsymbol{P}(j\omega)\right) \leq$

$\bar{\sigma}\left(\boldsymbol{W}_1(j\omega)\right), \ \forall\omega \in [0,\infty)$, where $\bar{\sigma}(\boldsymbol{A})$ denotes the maximum singular value of a matrix $\boldsymbol{A}$.

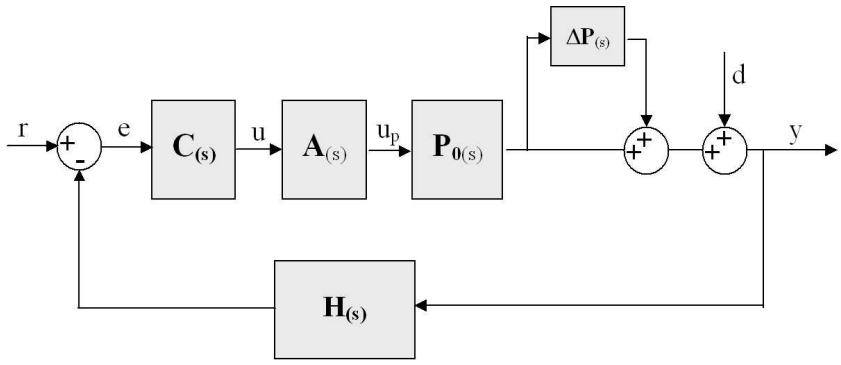


Figure 1. General feedback configuration.

The robust H_∞ optimal disturbance attenuation problem for the system in Fig. 1 lies in how to find a controller $\boldsymbol{C}(s)$ to stabilize the closed-loop system and attenuate the external disturbance. The principle criterion for this problem is simultaneously minimizing the robust stability performance

$$\|\boldsymbol{W}_1(s)\boldsymbol{T}(s)\|_\infty < 1 \tag{1}$$

and satisfying the following disturbance attenuation performance

$$\|\boldsymbol{W}_2(s)\boldsymbol{S}(s)\|_\infty \leq \gamma < 1 \tag{2}$$

where $\|\boldsymbol{A}\|_\infty = \max_\omega[\bar{\sigma}(\boldsymbol{A}(j\omega))]$, $\boldsymbol{W}_2(s)$ is a stable weighting function matrices specified by designer, and $\boldsymbol{S}(s)$ and $\boldsymbol{T}(s)$ are the sensitivity function and complementary sensitivity function, respectively, of the nominal system with the following representations:

$$\begin{aligned} \boldsymbol{S}(s) &= \left(\boldsymbol{I} + \boldsymbol{P}_0(s)\boldsymbol{A}(s)\boldsymbol{C}(s)\boldsymbol{H}(s)\right)^{-1} \\ \boldsymbol{T}(s) &= \left(\boldsymbol{I} + \boldsymbol{P}_0(s)\boldsymbol{A}(s)\boldsymbol{C}(s)\boldsymbol{H}(s)\right)^{-1}\left(\boldsymbol{P}_0(s)\boldsymbol{A}(s)\boldsymbol{C}(s)\boldsymbol{H}(s)\right) \end{aligned} \tag{3}$$

In short, the weighting matrices allow imposing what kind of *"unexpected"* uncertainty and what kind of *"unexpected"* disturbances the controlled system can deal with without problems [7].

4. Optimization Process

4.1 System Model

The aim is designing a longitudinal controller for the simplified F-14 fighter as modeled in MATLAB$^{TM}/Simulink$. The input into the linear model $\boldsymbol{P}_0$

is the elevator deflection angle δ_e, the outputs are the angle of attack α and the pitch rate q, and the states are q and the vertical velocity w, the feed-back is only on the q channel. The linearized plant model $\boldsymbol{P}_0$ is as follows:

$$\frac{\alpha}{\delta_e} = \frac{-0.09283s - 6.946}{s^2 + 1.296s + 4.501} \tag{4}$$

$$\frac{q}{\delta_e} = \frac{-6.885s - 4.017}{s^2 + 1.296s + 4.501} \tag{5}$$

4.2 Optimization Statement

The performance of the controlled system are specified in terms of command response characteristics to a normalized reference signal (step response), disturbance rejection features and robustness despite uncertainties of the plant.

The step command response characteristics are defined in terms of rise time t_r, settling time t_s and overshoot M_p. Rise time is defined here as the time the unit step response $y(t)$ takes from $y = 0.10$ to $y = 0.90$, i.e., $t_r = t(y_{90\%}) - t(y_{10\%})$, settling time is here defined as the time for $y(t)$ to stabilize within 5 percent of its final value and the overshoot is defined as the relative peak of $y(t)$, i.e., $Mp = (y_{peak} - y_\infty)/(y_\infty)$. Whereas, the disturbance rejection features are accounted by means of the ∞-norm of the sensitivity S and the robustness is accounted by the ∞-norm of the complementary sensitivity T.

The optimization processes were aimed at finding the six coefficients $(k_i,\ \ i = 1, 6)$ of the single-input/single-output (SISO) controller C, the general structure of which is

$$C(s) = \frac{k_1 s^2 + k_2 s + k_3}{k_4 s^2 + k_5 s + k_6}, \tag{6}$$

which minimize the elements of the vector function $\boldsymbol{f} = [f_{over,q}, f_{rise,q}]^T$, where $f_{over,q}$ and $f_{rise,q}$ are the overshoot and the rise time function of the q response, respectively.

Optimal solutions have to satisfy the constraint:

$$\left\| \begin{matrix} W_s S \\ W_t T \end{matrix} \right\|_\infty < 1 \tag{7}$$

and additional constraints have been imposed to the characteristics of the actuator signal, in order to avoid actuator saturation, and to the time responses in order to concentrate searching in the more interesting region of the Pareto front, avoiding useless solutions.

In this case we have:

$$W_s = \tfrac{s+10}{10s+1} \quad W_t = \tfrac{1000s+100}{s+10000} \tag{8}$$

which are scalar functions (the system is a SISO one) and $O_{max,q} = 2$, $Tr_{max,q} = 3$, $Ts_{max,q} = 5$, and $O_{max,\alpha} = 2$, $Tr_{max,\alpha} = 3$, $Ts_{max,\alpha} = 5$

are constraints of the overshoot, the rise time and the settling time for q and α, respectively. $\delta_{e,max} = 27$ deg and $\dot{\delta}_{e,max} = 60$ deg/s are the limits of the deflection angle and of the deflection rate of the elevator, respectively.

The used algorithms have been set with the parameters in Tables 1 and 2. The starting point approach is that of maximum ignorance, that is the initial population is uniformly sampled within the allowed range of variables ($-30 \leq k_i \leq 30$, $i = 1, 6$)

Table 1. NSGA-II parameters.

	Values
Population	*150*
Generations	*150*
Crossover prob.	*0.99*
Crossover index	*5*
Mutation prob.	*1/6*
Mutation index	*5*

Table 2. MOPED parameters.

	Values
Population	*150*
Generations	*150*
Fitness param.	*1.1*
Generator index	*1*
Constraint classes	*150/2*

4.3 Results

For each code 5 independent runs have been performed (with a maximum generation stopping criterion, 150 generations). Figure 2 shows two of the fronts (NSGA-II and MOPED) randomly chosen within the obtained set. A comparison between the two fronts points out that if the approximation of the NSGA-II is better in the internal side of the front, MOPED is able to find more solutions at the extremes.

This is surely due to the different search operators used by the algorithms. The real coded SBX has a rapid convergence to the central part of the front, but this means also an impoverishment of the information of the population, which makes difficult the exploration of distant regions. The operator of the MOPED, on the contrary, has a slower convergence and, as a consequence, can better explore a wider region.

Both algorithms are successful, when a deterministic sequential quadratic programming (SQP) algorithm with a random starting point (with the same bounds) is not able at finding any satisfying solution. For this problem the SQP algorithm demonstrated to be useless even if the starting point is a solution of the Pareto set.

In order to show how the robustness requirements influence the goodness of the solutions, the q output of two individuals are depicted in Fig. 3. One solution, which satisfies every constraint, is picked up from the Pareto set and the other one, which satisfies every constraints but that for the $W_2(s)S$, is picked up from an intermediate population. Notwithstanding the nominal response is

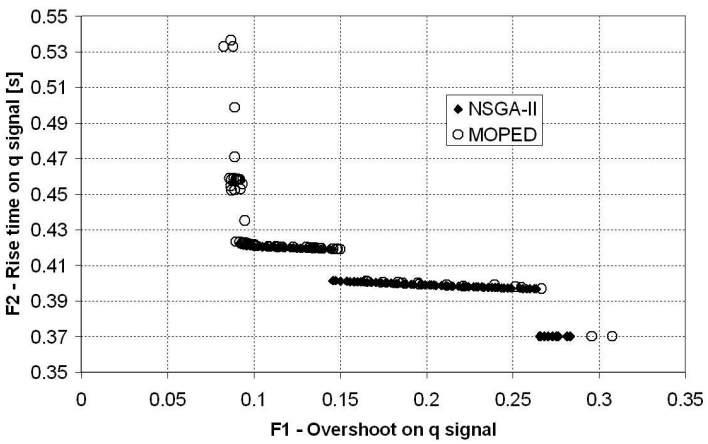


Figure 2. Obtained Pareto fronts.

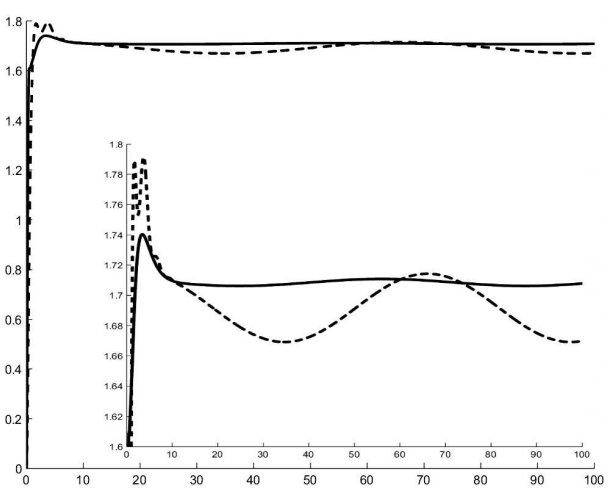


Figure 3. Time domain solutions (q channel) for a solution satisfying the constraint on the S function (solid line) and another one which does not (dashed line). On the right side a zoom allows a better view.

almost equal, a low frequency disturbance has minor effect on the solution with $\|W_2(s)S\| < 1$.

5. Conclusions

In this work we applied two different evolutionary algorithms to the optimization of a controller for longitudinal maneuvers. The problem is just a test case, where more traditional, deterministic techniques fail because of high nonlinearity of the problem. Both algorithms, a traditional NSGA-II and a newer EDA, demonstrate being able to handle such a kind of problem. Substantially, the two algorithms are similar and differences are only due to the used search operators. The results point out that if both algorithms allow finding Pareto solutions, the EDA one is able at maintaining diversity and, consequently, a better spread of individuals on the Pareto front.

This work, however, is only the first step towards a tool, which, by means of evolutionary techniques with or without the cooperation of other soft computing techniques, could carry out a fully automatic controller design for different kinds of flying vehicles.

References

[1] G. Avanzini, D. Biamonti, and E.A. Minisci. Minimum-Fuel/Minimum-Time Maneuvers of Formation Flying Satellites. In *Proceedings of the Astrodynamics Specialist Conference*, Big Sky, Montana, August 2003, pp. 1–20. AAS 03-654

[2] M. Costa and E. Minisci. MOPED: a Multi-Objective Parzen-based Estimation of Distribution algorithm. In *Proceedings of the Second International Conference on Evolutionary Multi-Criterion Optimization (EMO 2003)*, Faro, Portugal, 2003. *Lecture Notes in Computer Science*, 2632:282–294.

[3] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II. *IEEE Transaction on Evolutionary Computation*, 6:182–197, 2002.

[4] K. Fukunaga. *Introduction to statistical pattern recognition*. Academic Press, 1972.

[5] N. Khan, D.E. Golberg, and M. Pelikan. Multi-objective Bayesian Optimization Algorithm. Technical Report IlliGAL 2002009, University of Illinois at Urbana-Champain - IlliGAL, 2002.

[6] P. Larrañaga, J.A. Lozano. *Estimation of Distribution Algorithms. A New Tool for Evolutionary Computation*. Kluwer Academic Publishers, 2002.

[7] K. Zhou and J.C. Doyle. *Essentials of Robust Control*. Prentice-Hall, 1998.

EVOLUTIONARY BALANCING OF HEALTHY MEALS

Barbara Koroušić Seljak
Computer Systems Department
Jožef Stefan Institute, Ljubljana, Slovenia
barbara.korousic@ijs.si

Abstract In this paper we present an evolutionary heuristic based upon genetic algorithms for the problem of balancing healthy meals. We formulate the problem as a multiconstrained fractional knapsack problem that is easy to formulate, yet, its decision problem is in the class of NP-complete problems. Experimental results show that the genetic algorithm heuristic is capable of obtaining high-quality solutions for the problem of balancing healthy meals, requiring a polynomial computational time. Using this method, software can generate balanced healthy meals that consider multiple weakly-correlated dietary recommendations and guidelines and include as much as possible functional food and drinks that are available in a given season for a good price.

Keywords: Optimization, Multiconstrained fractional knapsack problem, Genetic algorithms, Penalty functions, Repair algorithms

1. Introduction

Appropriate nutrition is one of the most important protective factors against chronic diseases such as cardiovascular diseases, diabetes, cancer, and osteoporosis. Many dietary recommendations designed to prevent and control chronic diseases have been integrated with guidelines designed to prevent nutritional deficiencies and infectious diseases [3]. However, transfer of these useful but complex recommendations and guidelines into practice is not an easy task. We can use nutrition software as an assistant who handles numerous research quality data and tracks various nutrient information.

In this paper we introduce a meta-heuristic evolutionary optimization method for solving the problem of balancing healthy meals that is applied by high-end nutrition software. In Sect. 2 we provide a formulation of the problem of balancing meals as a multiconstrained fractional knapsack problem; in Sect. 3 we describe a genetic algorithm for the multiconstrained knapsack problem;

in Sect. 4 we evaluate the method; and in Sect. 5 we list our conclusions and suggest possible future work.

2. The Problem of Balancing Healthy Meals

Foods and drinks are defined by many research quality data, such as basic nutrients, vitamins, minerals, amino acids, fatty acids, etc. Finding a healthy and a balanced meal is a complex problem because of two reasons: there are many dietary recommendations and guidelines, and the search space is wide and complex. Moreover, the quality of food and drinks change dynamically with a season. Solutions of the problem are trade-off solutions. For such solutions no improvement in any constraint is possible without violating at least one of other constraints.

We formulate the problem of balancing healthy meals as a multiconstrained (multidimensional) fractional knapsack problem (MFKP) that is easy to formulate, yet, it can be solved efficiently by using some heuristic method. The MFKP is defined as follows:

Given food and drink items of different weights (nutrient data) and values (qualities) that are weakly-correlated, find the most valuable (healthy-and-balanced) mix of pieces (fractions) of items which fit in a knapsack (meal) of fixed volumes. Values are defined subjectively with respect to food functionality, seasonal availability and price. Knapsack volumes represent dietary recommendations and guidelines, such as:

- *Energy intake and calories;*
- *Intake of nutrients (carbohydrates, lipids, protein, vitamins and minerals);*
- *Ratio of essential fatty acids;*
- *Consumption of water, fruits and vegetables;*
- *Fiber requirement;*
- *Number of meals;*
- *Etc.*

Formal definition: There is a knapsack of M capacities C_j and N items. Each item i has a value $v_i \in R$, $v_i > 0$, and a set of weights $\omega_{i,j} \in R$, $\omega_{i,j} > 0$, one for each capacity. Find the selection of items $(x_i \in R, x_i > 0)$ that fit $\sum_{i=1}^{N} \omega_{i,j} x_i \; \Theta \; C_j$, Θ can be $\leq$, $=$, or $\geq$, $j = 1, \ldots, M$, and the total value, $\sum_{i=1}^{N} v_i x_i$, is maximized.

The MFKP is in the class of NP-complete problems [2].

3. Heuristic Method for the MFKP

There have been presented numerous methods for solving the knapsack problems. A comprehensive review of the multiconstrained 0-1 knapsack problem and its associated exact and heuristic algorithms is given by Chu and Beasley [4].

We decided to apply an evolutionary approach for balancing healthy meals in a heuristic way by using a genetic algorithm (GA) [1]. GAs simulate nature on a very abstract level to get solutions for sophisticated problems. They are searching through an arbitrary search space both for exploration and exploitation purpose.

With evolutionary GAs the MFKP can be solved by checking whether one of the trial solutions represented as an abstract chromosome is a good solution of the problem, i.e., whether it fulfills all the criteria. To create such a solution, selected trial chromosomes are recombined and/or perturbed genetically to form new chromosomes that are new members of the abstract population. It has been proved that each finite GA will always converge to its global optimum region [5], meaning that one good solution will always be found.

3.1 Direct Encoding

People consume several thousands of food and drink items, which intensifies difficulties in creation of a trial chromosome. To relieve this difficulty, we decided to separate N items into G groups, where $G \leq N$ and G is few tens. Creating a composite meal, we select at most one item from each group. Hence, in our representation, a trial chromosome contains G pairs $S[g]$ of item code i_g and its quantity (Fig. 1). A value $i_g = 0$ implies that no item is

g	1		2		...	G	
$S[g]$	01060	31	02033	3	...	13523	85

Figure 1. Representation scheme.

selected for a group g. The number of possible solutions is approx. $(\frac{N}{G})^{G|P|}$, $P = \{0.25, 0.5, 0.75, 1, 1.5, 2, 3, \ldots, 10\}$, assuming each item i_g has assigned a quantity x_{i_g} that is a multiple p_{i_g} of some predefined value (portion size). Item qualities are given by discrete, integer values from $\{1, 2, 3\}$. A higher value denotes a higher item functionality and availability in season.

In our implementation, the GA can start either with a random population of trial solutions or a population of solutions known by experience. The population size is few tens.

3.2 Fitness Evaluation

Each solution of the trial population is evaluated using the following fitness function:

$$Fitness(chromosome) = \sum_{g=1}^{G} v_{i_g} p_{i_g}. \tag{1}$$

Chromosomes having higher fitness values are more likely to be good solutions.

3.3 Infeasible Solutions

A chromosome might represent an infeasible solution. An infeasible solution is one for which at least one knapsack constraint is violated, i.e., $\neg \sum_{i=1}^{N} \omega_{i,j} x_i \; \Theta \; C_j$, Θ being $\leq$, $=$, or $\geq$, for some $j = 1, \ldots, M$. There are several ways of dealing with infeasible solutions in GAs [6]. However, we applied:

1 a *penalty function* to penalise the fitness of any infeasible solution without distorting the fitness landscape, and in addition

2 a *repair operator* which transforms any infeasible solution into a feasible one.

The penalty function is defined in a static way by adding a metric based on a number of constraints violated, such as:

$$Penalty(chromosome) = \sum_{j=1}^{m} X_j \delta_j, \tag{2}$$

where

$$\delta_j = \begin{cases} 1, & \text{if constraint } j \text{ is violated} \\ \\ 0, & \text{if constraint } j \text{ is satisfied} \end{cases},$$

and X_j is the weight of a certain constraint j.

An infeasible solution is left unrepaired in the population with some probability to prevent premature convergence of the algorithm. Namely, optimal solutions frequently lie on the boundary of the feasible region [7]. A repair operator consists of the following phases:

1 Rank the problem constraints (i.e., define constraint weights $X_j, j = 1, \ldots, M$);

2 Sort food and drink items in the chromosome according to the increasing order of their values (qualities) and the decreasing/increasing order of

their weights (nutrient data) for the exceeded/underestimated constraints, considering the constraint weights X_j;

3 For each item in the sorted chromosome, starting with the least quality one, find an alternative item with more appropriate nutrient profile, again considering the constraint weights X_j. A more appropriate nutrient profile is searched in the neighborhood of the item or randomly, depending on the repairing probability.

4 Repeat these local-optimization steps for several times (repairing rate) or until the chromosome does not satisfy all the constraints, respectively.

3.4 Parent Selection

To create (reproduce) new trial solutions two strings have to be selected from the current population as parents. In our implementation of the GA, solutions with the best fitness are more likely to be selected for reproduction as we apply the *elitism* strategy, where a number of least-fit members of the current trial population are interchanged with an equal number of the best-ranked chromosomes. Such an approach increases performance of the GA, because it prevents losing the best-found solutions.

The parent selection is realized via the *tournament* approach. This is based upon an idea of forming two pools of trial solutions, each consisting of the same number of chromosomes. Two solutions with the best fitness, each taken from one of the tournament pools, are chosen to be parents. Using a larger size of the tournament pools has the effect of increasing selection pressure on the more fit solutions. The problem of getting stuck in a locally optimum solution can happen. To avoid this problem, we adopt the standard (binary) tournament selection technique and realize the elitism through the interchange ratio of least-fit to best-ranked solutions. This ratio is in the order of 4 or 6 down to 1 chromosome per population, depending on the population size.

3.5 Crossover and Mutation

In crossover, two "fit" parents are mated to produce a child that replaces the least-fit solution in a given population if its fitness ranks above. This *steady-state approach* may perform better than generational GAs because it better retains feasible solutions found in the populations and may have higher selection pressure. We apply a *uniform crossover* operator to produce a solution that preserves the "genetic material" from both parents. Each element of the child chromosome (a pair of the item code and its quantity) is created by copying the corresponding element from one of the parents, chosen according to a binary random number generator $[0, 1]$. In our implementation of the GA, using a *two-point crossover* operator can also perform crossover. In this approach, copying

the corresponding elements from one parent, and all the others by copying the corresponding elements from other parent, creates elements of the child chromosome between two points, selected by using a crossover probability.

Once a child solution has been generated through parent selection and crossover, a mutation procedure is performed that mutates some randomly selected item codes and their quantities in the child solution. Each selected item is mutated in one of the following ways, selected randomly,

- by applying local optimization, i.e., the item code and its quantity are replaced by a close item from the item group;
- by replacing the item code and its quantity with a randomly selected item from the same group;
- by multiplying the item quantity by a random factor from $P = \{0.25, 0.5, 0.75, 1, 1.5, 2, 3, \ldots, 10\}$.

Mutation has to be used to prevent convergence to local optima. The rate of mutation is set to be a small value (in the order of 1 or 2 bits per chromosome).

3.6 Termination Criteria

We use two possibilities to stop the evolution:

- A *time-out approach*: when a certain number of populations have been generated and evaluated, the system is assumed to be in a stable state;
- A *wanted-solution approach*: once a solution with a predefined value of the fitness function is found, the optimization is terminated.

4. Evaluation of the GA for the MFKP

In order to evaluate the proposed evolutionary method for balancing healthy meals, we optimized a set of meals using the GA. We used the USDA nutrition database, Release 16 [8], which is the major source of food composition data in the United States. It includes nutrient profiles for more than 6500 food and drink items that are grouped into 23 groups. The items are ordered so that similar foods and drinks are grouped together. Each nutrient profile contains more than 30 values, such as macronutrients, elements, vitamins, etc. These data were used as weights. Qualities of items were defined considering their functionality and availability in a given season. Seasonal foods and drinks belonging to the list of functional foods and drinks (including broccoli, blueberries, red wine, avocado, etc.) were assigned higher values (qualities). We considered the following dietary recommendations and guidelines:

- Calories (kcal) per meal (based on user's individual criteria, such as weight, height, age, gender and activity level);

- Ratio of carbohydrates, proteins and lipids: 55:15:30 (% of total energy);
- Maximum level of cholesterol in milligrams;
- Recommended intake of vitamin E in milligrams;
- Ratio of linoleic (omega-6) and alpha-linoleic (omega-3) fatty acids: 2:1.

These constraints are equality constraints, except the "cholesterol" one, which is an inequality (less-than-equal) constraint.

We developed software that implements the GA that balances healthy meals using the Borland Delphi programming tool. It runs under the Microsoft Windows operating systems on a Pentium PC. The USDA database is available in a Microsoft Access format.

4.1 Experiments and Results

After several runs of the program, the most advantageous settings for the GA were defined (Table 1) and a set of good solutions was collected.

Table 1. Settings for the GA parameters.

Parameter	*Value*
Population size	20
Repairing probability	0.5
Repairing rate	3
Elitism ratio	0.2–0.05
Tournament pool size	2
Crossover probability	0.7
Mutation rate	0.05
Termination criteria	100 populations

It has proved that a repair operator has to be used in addition to the static penalty function. Otherwise, the search gets stuck in a local minimum and violates the termination criteria. We estimated that approximately each third trial solution was infeasible and required repairing. Although the worst-case time complexity of the repairing algorithm is $\sum_{g=1}^{G} O(N_g M)$, in practice $O(M)$ steps were needed to find a close feasible solution. However, the most difficult task in repairing was to derive the proper constraint weights $X_j, j = 1, \ldots, M$. We defined the highest weight to the "cholesterol" constraint and the lowest to the "energy intake" constraint. In between were the second and the fourth problem constraints (the ratios of macronutrients and essential fatty acids, respectively).

In Fig. 2 and 3 a high-quality daily-meal solution generated from an initial population of random trial solutions that satisfies the selected problem constraints is presented. The quantities were selected as multipliers of the portion

sizes. Larger portions of more than 100 grams were multiplied by a factor from $\{0.25, 0.5, 0.75, 1, 1.5, 2\}$ and smaller portions of few grams by a factor from $\{1, \ldots, 10\}$, respectively.

Grams	*Food or drink item*
72	ORANGE DRK, BRKFST TYPE, W/JUC & PULP, FRZ CONC
38	CEREALS, MALTEX, DRY
141	WEIGHT WATCHERS ON-THE-GO CHICK, BROCLI & CHDR POCKT SNDWCH, FRZ
496	SOUP, TOMATO, LO NA, W/H2O
80	FAST FOODS, POTATO, MASHED
31	TURKEY, YOUNG HEN, SKN ONLY, CKD, RSTD
1	GINGER, GROUND
123	BEETS, CND, REG PK, SOL & LIQUIDS
28	CAKE, CHERRY FUDGE W/CHOC FRSTNG
31	ENSURE PLUS, LIQ NUTR
150	BANANAS, RAW
21,5	SOYBEANS, MATURE SEEDS, RSTD, SALTED
195	RICE, BROWN, LONG-GRAIN, CKD
85	MACKEREL, ATLANTIC, RAW
42	ALMOND PASTE

Figure 2. A good daily-meal solution.

RESULTS	*Recommended*	*Achieved*
Quality	3	3
Energy (Kcal)	1800 ± 100	1875.7
Protein (% of energy)	10–15 (preferable 15)	14.5
Total lipids (% of energy)	20–30 (preferable 30)	29.8
Carbohydrates (% of energy)	50–60 (preferable 55)	55.7
Vitamin E (mg)	15 ± 2	14.1
Saturated fatty acids (% of energy)	≤ 10	7.5
Omega-6 FA + Omega-3 FA (g)	$(11 + 5.5) \pm 2$	17.4
Cholesterol (mg)	≤ 300	120.4

Figure 3. Nutritional profile for the daily-meal solution.

In Fig. 4 performance of the proposed balancing method, based on measurements of the number of times the trial solutions and constraints were evaluated to come within a certain fraction of the optimum, are presented.

5. Conclusions

In the paper, we have presented a heuristic method for balancing healthy meals that considers several constraints and the quality of food and drinks. The method is based on a steady-state genetic algorithm because the search space is complex and wide, yet, it has very small feasible regions. The GA uses the direct coding of problem solutions, the elitism and the tournament approach for selection of parents, uniform and two-point crossover, and "local-optimization" mutation. Infeasible solutions are penalized by decreasing the "fitness" of the

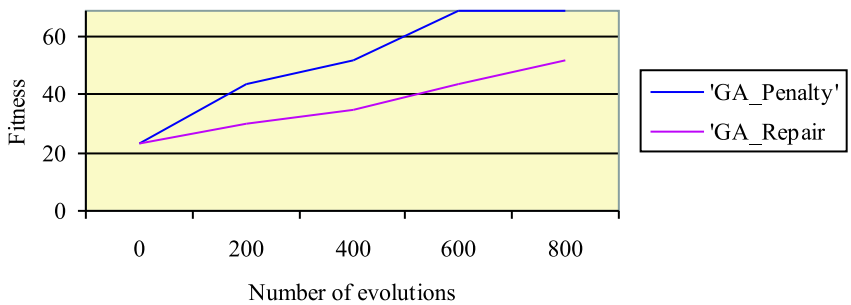


Figure 4. Performance of the balancing method.

evaluation function. Some infeasible solutions are further repaired so that the least-quality food and drink items are changed with more appropriate items or their quantities are modified by a multiple of the predefined portion size. We collected the experimental results by running the GA from an initial population of random trial solutions. Including some "real" meals, the method could perform better. We could also experiment with other (dynamic) penalty functions to reduce the cost of repairing. Last but not least the USDA database need to be replaced with the Slovene national nutritional database.

Acknowledgment

The work presented in the paper has been supported by the Slovenian Ministry of Health (project: *Development of a public domain server application for food analysis and optimization that considers modern dietary recommendations*).

References

[1] D.E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, Reading, 1989.

[2] M.R. Garey and D.S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman, 1979.

[3] D. Pokorn. *Dietetika*. DZS, Ljubljana, 1999.

[4] P.C. Chu and J.E. Beasley. A Genetic Algorithm for the Multidimensional Knapsack Problem. *Journal of Heuristics*, 4:63–86, 1998.

[5] C.L. Karr, I. Yakushin, and K. Nicolosi. Solving inverse initial-value, boundary-value problems via genetic algorithm. *Engineering Applications of Artificial Intelligence*, 13:625–633, 2000.

[6] Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer-Verlag, 1996.

[7] W. Siedlecki and J. Sklansky. Constrained genetic optimization via dynamic reward-penalty balancing and its use in pattern recognition. In *Proceedings of the Third International Conference on Genetic Algorithms*, Los Altos, USA, 1989, pp.141–150.

[8] United States Department of Agriculture (USDA), Nutrient Data Laboratory. http://www.nal.usda.gov/fnic/foodcomp.

INFORMATION SOCIETY 2004

In its 7th year, the Information Society Conference (http://is.ijs.si) is for the first time organized in the European Union. It is one of the leading conferences in the Central Europe, and an important European scientific meeting on computer and information sciences.

In 2004, there are 11 independent conferences constituting the multi-conference, which is yet another important achievement. Very wide spectrum of interests and topics makes this conference unique among similar conferences. The Conference is running in 2 to 3 parallel sessions for over one week with over 200 presentations of scientific papers, round-tables, and discussions. Many events are video recorded and can be viewed through the Internet (http://solomon.ijs.si/). Typically, there are 150 to 200 papers published in several separate proceedings with 500 to 700 pages, and in 2 special journal issues. One of them is Informatica with its 25 years of tradition in excellent research publications.

The Information Society 2004 Conference consists of the following conferences:

- BIOMA - Bioinspired Optimization Methods and their Applications
- Cognitive Sciences
- Collaboration and Information Society
- CSeB - Complex Systems in e-Business
- Data Mining and Warehouses
- Development and Reengineering of Information Systems
- Education in Information Society
- Intelligent Systems
- Language Technologies
- Management in Information Society
- Theoretical Computer Science

The motto of the Conference is a synergy of different interdisciplinary approaches dealing with the challenges of the information society. The major driving forces of the Conference are search and demand for new knowledge related to information, communication, and computer services. We present, analyze, and verify new discoveries in scientific community first, and prepare a ground for their enrichment and development in practice. The main objective of the Conference is presentation and promotion of research results, to encourage their practical application in new ICT products and information services in Slovenia and also wider region.

The Conference is co-organized and supported by several major research institutions and societies, among them ACM Slovenia, i.e. the Slovenian chapter of ACM. We would like to express our appreciation to the Slovenian Government for cooperation and support, in particular through the Ministry of Education, Science and Sport, and the Ministry of Information Society.

On behalf of the conference organizers we would particularly like to thank all participants for their valuable contribution and their interest in this event.

Cene Bavec, Programme Committee Chair
Matjaž Gams, Organizing Committee Chair

CONFERENCE COMMITTEES

International Programme Committee
Vladimir Bajic, South Africa
Heiner Benking, Germany
Se Woo Cheon, Korea
Howie Firth, UK
Vladimir Fomichov, Russia
Vesna Hljuz Dobric, Croatia
Alfred Inselberg, Israel
Jay Liebowitz, USA
Huan Liu, Singapore
Henz Martin, Germany
Marcin Paprzycki, USA
Karl Pribram, USA
Claude Sammut, Australia
Jiri Wiedermann, Czech Republic
Xindong Wu, USA
Yiming Ye, USA
Ning Zhong, USA
Wray Buntine, Finland

Programme Committee
dr. Cene Bavec, chair
dr. Tomaž Kalin, co-chair
dr. Jozsef Györkös, co-chair
prof. dr. Tadej Bajd
mag. Jaroslav Berce
prof. dr. Marko Bohanec
prof. dr. Ivan Bratko
dr. Andrej Brodnik
dr. Dušan Caf
prof. dr. Saša Divjak
dr. Tomaž Erjavec
doc. dr. Bogdan Filipič
prof. dr. Matjaž Gams
Marko Grobelnik
prof. dr. Nikola Guid
dr. Marjan Heričko
prof. dr. Borka Jerman Blažič Džonova
prof. dr. Gorazd Kandus
prof. dr. Marjan Krisper
mag. Andrej Kuščer
prof. dr. Jadran Lenarčič
dr. Borut Likar
dr. Dunja Mladenič
dr. Franc Novak
prof. dr. Marjan Pivka
prof. dr. Vladislav Rajkovič
asist. dr. Grega Repovš
prof. dr. Ivan Rozman
Niko Schlamberger, dipl. ing.
prof. dr. Franc Solina
prof. dr. Stanko Strmčnik
dr. Tomaž Šef
dr. Jurij Šilc
prof. dr. Jurij Tasič
prof. dr. Denis Trček
prof. dr. Andrej Ule
doc. dr. Tanja Urbančič
prof. dr. Boštjan Vilfan
prof. dr. David B. Vodušek
prof. dr. Baldomir Zajc
prof. dr. Blaž Zupan

Organizing Committee
prof. dr. Matjaž Gams, chair
mag. Aleksander Pivk
dr. Damjan Demšar
dr. Gregor Papa
Lili Lasič
Mili Bauer, dipl. ing
Mitja Lasič
Mitja Luštrek, dipl. ing
Tea Robič, dipl. ing

There are 11 independent conferences and 5 separate proceedings:

- Proceedings A: Cognitive Science
- Proceedings B: Language Technologies
- Proceedings C: Complex Systems in e-Business, Data Mining and Warehouses, Intelligent Systems
- Proceedings D: Collaboration and Information Society, Development and Reengineering of Information Systems, Education in Information Society, Management in Information Society, Theoretical Computer Science
- Proceedings E: BIOMA - Bioinspired Optimization Methods and their Applications