BIOINSPIRED OPTIMIZATION METHODS AND THEIR APPLICATIONS

BIOINSPIRED OPTIMIZATION METHODS AND THEIR APPLICATIONS

Proceedings of the Fourth International Conference on Bioinspired Optimization Methods and their Applications, BIOMA 2010

20–21 May 2010, 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
Printed by Tiskarna Artelj, Ljubljna
Published by Jožef Stefan Institute, Ljubljana, Slovenia

Typesetting in `kapproc`-based LaTeX style with kind permission from Kluwer Academic Publishers

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

004.02(082)
005.519.1(082)
510.5(082)

INTERNATIONAL Conference on Bioinspired Optimization Methods and their Applications (4 ; 2010 ; Ljubljana)
Bioinspired optimization methods and their applications : proceedings of the Fourth International Conference on Bioinspired Optimization Methods and their Applications - BIOMA 2010, 20-21 May 2010, Ljubljana, Slovenia / edited by Bogdan Filipič, Jurij Šilc. - Ljubljana : Jožef Stefan Institute, 2010

ISBN 978-961-264-017-0
1. Gl. stv. nasl. 2. Filipič, Bogdan

250776320

BIOMA 2010 is partially financed by the Slovenian Research Agency.

Contents

Preface

Competing among themselves and adapting to the environment, members of biological populations accumulate experience and improve their performance. Their genetic material is recombined and propagated from generation to generation according to the laws of genetics. Relying on elementary activities of individuals, societies of insects and bird flocks exhibit complex emergent behavior. These and many other fascinating natural phenomena have been a rich source of inspiration in computer algorithms design for decades. As a result, the family of bioinspired algorithms has become quite large and includes evolutionary algorithms, ant colony optimization, particle swarm optimization, and artificial immune systems, to name just a few. They were designed to overcome the drawbacks of traditional algorithms in demanding application scenarios where little, if any, information is available to assist problem solving. Optimization is an area where these techniques are studied and exercised with particular practical success.

This volume contains recent theoretical and practical contributions to the field of bioinspired optimization presented at the Fourth International Conference on Bioinspired Optimization Methods and their Applications (BIOMA 2010), held at the Jožef Stefan Institute, Ljubljana, Slovenia, on 20 and 21 May 2010.

Encouraged by the success of the previous BIOMA conferences organized in 2004, 2006 and 2008 as part of the Information Society Multiconference, BIOMA starts its own way this year. It continues its mission of bringing together theoreticians and practitioners to present their recent achievements and exchange the ideas, and promoting the bioinspired optimization methods to wider audience.

Each paper submitted to the conference was reviewed by three members of the international program committee. In the reviewing procedure, 18 papers were selected for presentation at the conference and publication in the proceedings. They were contributed by 40 (co)authors coming from 9 countries.

The BIOMA 2010 invited speaker is Christian Blum, a research fellow at the Technical University of Catalonia, Barcelona, Spain. He has contributed to the fields of swarm intelligence and hybridization of metaheuristics. His talk on hybrid mataheuristics reviews recent developments in combining metaheuristics, such as ant colony optimization, evolutionary algorithms and variable neighborhood search, with techniques from operations research and artificial intelligence.

Theoretical and algorithmic studies presented at the conference address a variety of issues in bioinspired optimization: parallelization of the differential evolution algorithm, analysis of exploration and exploitation in evolutionary algorithms, component decomposition in parallel differential evolution, a distinct candidate approach to multiobjective optimization, parameter estimation in a cell regulatory system model with bioinspired optimization algorithms, a slow feature analysis approach to the analysis of nonstationary time series, and a self-organizing cognitive architecture. The applied work reports come from a number of interesting domains: multi-processor system-on-chip design, design optimization in mechanical, electrical and chemical engineering, optimization of a metallurgical production process, optimization of broadcasting in sensor networks, production planning, scheduling of bus transportation staff, online mathematics learning, woody plants model recognition, and variable selection in econometric modeling with stochastic algorithms.

BIOMA 2010 is sponsored by the Slovenian Research Agency. Technical sponsors of the conference are the World Federation on Soft Computing, the Slovenian Artificial Intelligence Society (SLAIS), and the Jožef Stefan Institute.

Our thanks go to the conference sponsors, members of the program and organizing committees, the invited speaker, paper presenters and other participants for contributing their parts to the conference. We wish you an inspiring scientific meeting and a pleasant stay in Ljubljana.

Ljubljana, 6 May 2010

Bogdan Filipič and Jurij Šilc

Program Committee

Bogdan Filipič, *Chair*, Jožef Stefan Institute, Ljubljana, Slovenia
Jurij Šilc, *Chair*, Jožef Stefan Institute, Ljubljana, Slovenia
Christian Blum, Technical University of Catalonia, Barcelona, Spain
Janez Brest, University of Maribor, Slovenia
Dirk Büche, MAN Turbo AG, Zürich, Switzerland
Carlos A. Coello Coello, CINVESTAV-IPN, Mexico
Kalyanmoy Deb, Indian Institute of Technology Kanpur, India
Rolf Drechsler, University of Bremen, Germany
Jürgen Jakumeit, ACCESS e.V., Aachen, Germany
Peter Korošec, Jožef Stefan Institute, Ljubljana, Slovenia
Barbara Koroušić Seljak, Jožef Stefan Institute, Ljubljana, Slovenia
Marjan Mernik, University of Maribor, Slovenia
Zbigniew Michalewicz, University of Adelaide, Australia
Edmondo Minisci, University of Glasgow, UK
Nalini N, Siddaganga Institute of Technology, Karnataka, India
Gregor Papa, Jožef Stefan Institute, Ljubljana, Slovenia
Jim Tørresen, University of Oslo, Norway
Xin-She Yang, National Physical Laboratory, Teddington, UK

Organizing Committee

Gregor Papa, *Chair*, Jožef Stefan Institute, Ljubljana, Slovenia
Erik Dovgan, Jožef Stefan Institute, Ljubljana, Slovenia
Peter Korošec, Jožef Stefan Institute, Ljubljana, Slovenia
Barbara Koroušić Seljak, Jožef Stefan Institute, Ljubljana, Slovenia
Katerina Tashkova, Jožef Stefan Institute, Ljubljana, Slovenia
Tea Tušar, Jožef Stefan Institute, Ljubljana, Slovenia
Vida Vukašinović, Jožef Stefan Institute, Ljubljana, Slovenia

Contributing Authors

Christian Blum received his Ph.D. degree in Natural Sciences from the Free University of Brussels, Belgium, in 2004. He is currently a research fellow at the Technical University of Catalonia, Barcelona, Spain. His research interests include swarm intelligence methods for optimization and the management of large-scale wireless networks. Moreover, he has contributed to the field of hybrid metaheuristics, which deals with the combination of metaheuristics with more classical techniques for optimization. He is currently an associate editor for the Swarm Intelligence journal (Springer), among others, and a member of the PC of around 10-15 conferences per year.

Borko Bošković received his Ph.D. degree in Computer Science from the University of Maribor, Slovenia, in 2010. He is currently a teaching assistant at the University of Maribor, Faculty of Electrical Engineering and Computer Science. His research interests include evolutionary algorithms and chess program development.

Janez Brest received his Ph.D. degree in Computer Science from the University of Maribor in 2000. He is currently an associate professor at the University of Maribor, Faculty of Electrical Engineering and Computer Science. His research interests include evolutionary computing, artificial intelligence, and optimization. He is a member of IEEE and ACM.

Matej Črepinšek received his Ph.D. degree in Computer Science from the University of Maribor, Slovenia, in 2005. He is currently a teaching assistant at the University of Maribor, Faculty of Electrical Engineering and Computer Science. His research interests include programming languages, compilers, grammar-based systems, grammatical inference, and evolutionary computations.

Matjaž Depolli received his B.S. degree in Computer Science from the University of Ljubljana, Slovenia, in 2005. He is a research assistant at the Department of Communication Systems of the Jožef Stefan Institute, Ljubljana, and a Ph.D. student of New media and e-science at the Jožef Stefan International Postgraduate School where he is completing his dissertation on parallelization of evolutionary algorithms for multiobjective optimization. His research interests are in parallel computing, evolutionary algorithms, stochastic multiobjective optimization, and application of these techniques in engineering and medicine.

Sašo Džeroski received his Ph.D. degree in Computer Science from the University of Ljubljana, Slovenia, in 1995. He is currently a scientific councilor at the Department of Knowledge Technologies, Jožef Stefan Institute, Ljubljana, Slovenia. He is also an associate professor at the Jožef Stefan International Postgraduate School in Ljubljana. His research interests include data mining and machine learning and their applications in environmental sciences (ecology) and life sciences (systems biology). He is an ECCAI fellow, member of the executive board of SLAIS, member of ACM SIGKDD and IEEE.

Bogdan Filipič received his Ph.D. in Computer Science from the University of Ljubljana, Slovenia, in 1993. He is currently a senior researcher at the Department of Intelligent Systems of the Jožef Stefan Institute, Ljubljana, and an associate professor of Computer and Information Science at the University of Ljubljana. His research interests include stochastic optimization, evolutionary computation and intelligent data analysis. He is also active in promoting these techniques in practical problem solving in engineering and industry. He has been a principle investigator in several projects dealing with optimization of continuous casting of steel, scheduling of automobile production, optimal cargo transportation, and ICT for energy efficiency. Bogdan Filipič serves as a member of editorial boards of several scientific journals and a programme committee member for the Genetic and Evolutionary Computation Conference (GECCO) and Congress on Evolutionary Computation (CEC).

Antonio Gargano is currently a Ph.D. candidate at the Bocconi University, Department of Finance, Milan, Italy.

Hugo Hernández received his M.Sc. degree in Computer Science from the Technical University of Catalonia, Spain, in 2008. He is currently a Ph.D. student at the Technical University of Catalonia, Department of Languages and Computer Systems. His research interests include optimization, swarm intelligence and sensor networks.

Dejan Hrnčič received his B.Sc. degree from University of Maribor, Slovenia, Faculty of Electrical Engineering and Computer Science in 2007. He is currently working on his Ph.D. thesis in Computer Science at the same faculty. His research interests include evolutionary computations, grammatical inference and optimization techniques.

Peter Dueholm Justesen received his M.Sc. degree from the Department of Computer Science, Aarhus University, Denmark, in 2007. He is currently a Ph.D. student co-funded by Department of Computer Science, Aarhus University and Grundfos R&T.

Carlos Kavka received his Ph.D. degree in Computer Science from the University of Paris-Sud, France in 2006. He has been professor at the National University of San Luis, Argentina, Director of Workshops at the International Center for Theoretical Physics, Trieste, Italy, and is currently a researcher at ESTECO SRL, Trieste, Italy. His research interests include evolutionary algorithms, fuzzy logic, real time software design, and service oriented architecture.

Željko Knez received his Ph.D. degree in Chemical Engineering from the University of Maribor, Slovenia, in 1984. He is currently a professor at the University of Maribor, Faculty of Chemistry and Chemical Engineering. His research interests include high pressure processes. He is a member of executive board of EFCE European Federation of Chemical Engineering, head of Working party "High pressure technologies" and member of editorial board of The Journal of Supercritical Fluids.

Maša Knez Hrnčič received her B.Sc. degree from University of Maribor, Slovenia, Faculty of Chemistry and Chemical Engineering in 2008. She is currently working on her Ph.D. thesis in Chemical Engineering at the same faculty. Her research interests include high pressure technologies.

Patrick Koch received his diploma degree in Computer Science from the University of Paderborn, Germany, in 2008. He is currently researcher at the Cologne University of Applied Sciences, Faculty of Computer Science. His research interests include computational intelligence, machine learning and optimization.

Wolfgang Konen received his Ph.D. degree in Physics from the University of Mainz, Germany, in 1990. He is currently a professor at the Cologne University of Applied Sciences, Faculty of Computer Science and Engineering. His research interests include computational intelligence, data mining, machine learning, reinforcement learning, pattern recognition and computer vision. He is a member of the ACM SIGEVO Special Interest Group on Genetic and Evolutionary Computation and of the GMA Special Interest Group on Computational Intelligence.

Peter Korošec received his Ph.D. from the Jožef Stefan International Postgraduate School, Ljubljana, Slovenia, in 2006. Since winter 2002 he has been a researcher at the Jožef Stefan Institute, Ljubljana, Slovenia. He is presently a researcher at the Computer Systems Department and an assistant professor at the University of Primorska, Faculty of Mathematics, Natural Sciences and Informational Technologies Koper, Koper, Slovenia. His current areas of research include combinatorial and numerical optimization with ant-based metaheuristics, and distributed computing. More information about his work can be found at `http://csd.ijs.si/korosec/`.

Barbara Koroušić Seljak received her Ph.D. degree in Computer Science and Informatics from the University of Ljubljana, Slovenia, in 1997. She is currently a researcher at the Computer Systems Department of the Jožef Stefan Institute, Ljubljana. Her research focuses on design of real-time and embedded systems. At the Jožef Stefan International Postgraduate School she runs a course on Real-Time Embedded Systems as a senior lecturer. Her research interests include modern heuristic methods, which may be applied to practical optimization problems.

Jari Kyngäs received his Ph.Lic. degree in Information Technology from the University of Joensuu, Finland, in 1997. He is currently a principal lecturer and a researcher at the Satakunta University of Applied Sciences. His research interests include sports scheduling, staff scheduling, production line scheduling, automation and robotics.

Erkki Laitinen received his Ph.D. in Applied Mathematics from the University of Jyväskylä, Finland, in 1989. He is a now a lecturer in Applied Mathematics at the Department of Mathematical Sciences of the Faculty of Science at the University of Oulu, Finland, and a docent in Computer Science at the Department of Mathematical Information Sciences at the University of Jyväskylä. His current research interests are in nonlinear optimization techniques, numerical analysis, modeling and simulation. He has published several papers in journals and conference proceedings, and participated in many industrial projects dealing, among others, with optimal water spray control in the steel continuous casting process.

Shih-Hsi "Alex" Liu received his Ph.D. degree in Computer Science from the University of Alabama at Birmingham, USA, in 2007. He is currently an assistant professor in the Department of Computer Science at the California State University, Fresno. Dr. Liu's primary research interests are in evolutionary computations, domain-specific languages, service-oriented computing, software product line engineering, and model-driven engineering. He is a member of ACM, IEEE, and Upsilon Pi Epsilon.

Marjan Mernik received his Ph.D. degree in Computer Science from the University of Maribor, Slovenia, in 1998. He is currently a professor at the University of Maribor, Faculty of Electrical Engineering and Computer Science. He is also a visiting professor at the University of Alabama at Birmingham. His research interests include programming languages, compilers, grammar-based systems, grammatical inference, and evolutionary computations. He is a member of the IEEE, ACM and EAPLS.

Ferrante Neri received his Ph.D. degree in Electrotechnical Engineering from the Technical University of Bari, Italy, and his Ph.D. degree in Computer Science in 2007 from the University of Jyväskylä, Jyväskylä, Finland. He is currently an assistant professor in Simulation and Optimization at the University of Jyväskylä and a research fellow with the Academy of Finland, Helsinki, Finland. His research interests include memetic computing, differential evolution, noisy and large-scale optimization.

Kimmo Nurmi received his Ph.D. degree in Applied Mathematics from the University of Turku, Finland, in 1999. He is currently a research director at the Satakunta University of Applied Sciences. His research interests include sports scheduling, staff scheduling, production line scheduling, automation and robotics.

Gregor Papa is a research assistant at the Computer Systems Department, Jožef Stefan Institute, Ljubljana, Slovenia, and an assistant professor at the Jožef Stefan International Postgraduate School, Ljubljana, Slovenia. He received his M.Sc. and Ph.D. degrees in Electrical Engineering from the University of Ljubljana, Slovenia, in 2000 and 2002, respectively. His research interests include optimization techniques, meta-heuristic algorithms, high-level synthesis of integrated circuits, hardware implementations of high-complexity algorithms, and industrial product improvements. His work is published in several international journals. More information about his work can be found at `http://csd.ijs.si/papa/`.

Jakob Puchinger received his Ph.D. degree in Computer Science from the Vienna University of Technology, Austria, in 2006. He is currently a research scientist at the Austrian Institute of Technology in Vienna in the Dynamic Transportation Systems team of the Mobility department. He is also external lecturer at the Vienna University of Technology. His research interests include transport optimization and logistics and hybrid combinatorial optimization. He is a member of of various conference programme committees as well as reviewer for several international journals.

Günther R. Raidl received his Ph.D. degree in Computer Science from the Vienna University of Technology, Austria, in 1994. He is currently a professor at the Vienna University of Technology, Faculty of Informatics. His research interests include algorithms and data structures in general, combinatorial optimization, mathematical programming techniques, metaheuristics, and hybrid optimization approaches.

Andrea Roli received his Ph.D. degree in Computer Science from "Alma Mater Studiorum" Universita di Bologna, Italy, in 2003. He is currently assistant professor at the same University, II Faculty of Engineering. He is also a visiting professor at IRIDIA, Universit Libre de Bruxelles. His research interests are in the intersection between artificial

intelligence and complex systems and, in particular, they include hybrid metaheuristics, bioinformatics and genetic network models. He is member of the steering committee of the Italian Association for Artificial Intelligence.

Oscar Javier Romero López received his Ph.D. degree in Computer Science from the Politécnica de Madrid University, Spain, in 2010. He is currently a professor at the Konrad Lorenz University, Faculty of Intelligent Systems. He is also a visiting researcher at the Imperial College London. His research interests include cognitive science, intelligent agents, bio-inspired artificial intelligence, evolutionary computation, and artificial immune systems.

Jurij Šilc received his Ph.D. in Electrical Engineering from the University of Ljubljana, Slovenia, in 1992. In 1980 he joined the Jožef Stefan Institute, where he is now a senior 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 and an assistant professor at the Jožef Stefan International Postgraduate School, Ljubljana. He has published over 150 research papers on subjects including magnetic bubble memories, dataflow computing, algorithm mapping, parallel processing, high-level synthesis, combinatorial and numerical optimization, and processor architecture. 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. He is a member of the IEEE.

Katerina Tashkova is a Ph.D. student at the Jožef Stefan International Postgraduate School, Ljubljana, Slovenia. Since 2007 is young researcher at Computer System Department, Jožef Stefan Institute, Ljubljana, Slovenia. Current areas of research include numerical optimization, mathematical modeling, equation discovery. She has received B.S. in electrical engineering from "St.Cyril and Methodius" University, Skopje, Macedonia, in 2005.

Ville Tirronen received the Ph.D degree in computer science in 2008 from the University of Jyväskylä, Jyväskylä, Finland. He is currently a post-doctoral fellow at the Department of Mathematical Information Technology, University of Jyväskylä. His research interests include

image processing, functional programming, differential evolution, and memetic algorithms

Ljupčo Todorovski received his Ph.D. degree in Computer Science from the University of Ljubljana, Slovenia, in 2003. He is currently a professor at the University of Ljubljana, Faculty of Public Administration. He has been a post-doc fellow at the Stanford University (2004-2005) and Osaka University (2003). His research interests include artificial intelligence, machine learning, equation discovery, and applications of methods from these three areas to real-life problems in life sciences, ecology, and public administration.

Alessandro Turco received his Ph.D. degree in Applied Mathematics from the International School for Advanced Studies (SISSA), Trieste, Italy, in 2008. He is currently a researcher at ESTECO SRL, Trieste, Italy. His research interests include multi-objective optimization, designs of experiments, and metaheuristics.

Rasmus Kjær Ursem received his Ph.D. degree from the Department of Computer Science, Aarhus University, Denmark, in 2003. He has since then been employed as optimization expert at Grundfos R&T, which is the research and technology organization of the Danish pump manufacturer Grundfos. Currently, Grundfos R&T employs more than 200 researchers world-wide covering multiple disciplines such as fluid mechanical engineering, structural engineering, electrical engineering, motor engineering, computer science, physics, chemistry, and biology.

Vida Vukašinović is a young researcher at the Computer Systems Department at the Jožef Stefan Institute, Ljubljana, Slovenia. She graduated from Faculty of Mathematics and Physics at University of Ljubljana. Currently she is a Ph.D. student at Jožef Stefan International Postgraduate School. Her research interests include optimization techniques, meta-heuristic algorithms and graph theory.

Matthieu Weber is an assistant professor in Mobile Technology and Business and a Ph.D. student at the University of Jyväskylä, Finland. His research interests include differential evolution and large-scale optimization.

Barbara Zadobovšek received her B.Sc. degree in Computer Science from the University of Maribor, Slovenia, in 2009. She is currently an engineer at software company Nova vizija d.d., Slovenia. Her research interests include programming languages, and evolutionary computations.

Aleš Zamuda received his M.Sc. degree in Computer Science from the University of Maribor, Slovenia, in 2008. He is currently a teaching assistant at the University of Maribor, Faculty of Electrical Engineering and Computer Science. His research interests include evolutionary algorithms, multicriteria optimization, artificial life, and computer animation. He is a member of IEEE.

Viljem Žumer is a full professor in the Faculty of Electrical Engineering and Computer Science, University of Maribor, Slovenia. He is the Head of Laboratory for Computer Architecture and Programming Languages. His research interests include programming languages and computer architecture. He is a member of IEEE.

I

INVITED CONTRIBUTION

A BRIEF SURVEY ON HYBRID METAHEURISTICS

Christian Blum
ALBCOM research group
Universitat Politècnica de Catalunya, Barcelona, Spain
cblum@lsi.upc.edu

Jakob Puchinger
Mobility Department
Austrian Institute of Technology, Vienna, Austria
jakob.puchinger@ait.ac.at

Günther R. Raidl
Institute of Computer Graphics and Algorithms
Vienna University of Technology, Vienna, Austria
raidl@ads.tuwien.ac.at

Andrea Roli
Dipartimento di Elettronica, Informatica e Sistemistica (DEIS)
Alma Mater Studiorum, Università di Bologna, Italy
andrea.roli@unibo.it

Abstract The combination of components from different algorithms is currently one of the most successful trends in optimization. The hybridization of metaheuristics, such as ant colony optimization, evolutionary algorithms, and variable neighborhood search, with techniques from operations research and artificial intelligence plays hereby an important role. The resulting hybrid algorithms are generally labelled hybrid metaheuristics. The rising of this new research field was due to the fact that the focus of research in optimization has shifted from an algorithm-oriented point of view to a problem-oriented point of view. In this brief survey on hybrid metaheuristics we provide an overview on some of the most interesting and representative developments.

Keywords: Exact techniques, Hybridization, Metaheuristics, Optimization

1. Introduction

The term *metaheuristic* was introduced to define heuristic methods that can be applied to a wide set of different problems. In other words, a metaheuristic can be seen as a general algorithmic framework which can be applied to different optimization problems with relatively few modifications to make them adapted to a specific problem. Genetic and evolutionary algorithms, tabu search, simulated annealing, iterated local search, and ant colony optimization, just to name a few, are typical representatives falling under this generic term. Each of them has an individual historical background, follows certain paradigms and philosophies, and puts one or more particular strategic concepts in the foreground. For a detailed introduction to metaheuristics we refer the interested reader to [13, 26].

In contrast to the early days of metaheuristic research, the last 5-10 years have produced a large number of algorithms that simply do not fit into a single metaheuristic category. This is because these untraditional approaches combine various algorithmic ideas, often originating from several branches of artificial intelligence, operations research and computer science in general. Such approaches are commonly referred to as *hybrid metaheuristics* [11]. The lack of a precise definition of this term is sometimes subject to criticism. In our opinion, however, the relatively open nature of this term is rather helpful, as strict borderlines between related fields of research are often a hindrance for creative thinking and the exploration of new research directions.

The main motivation for the hybridization of different algorithmic concepts has been to obtain better performing systems that exploit and combine advantages of the individual pure strategies, that is, hybrids are believed to benefit from *synergy*. In fact, choosing an adequate combination of multiple algorithmic concepts is often the key for achieving top performance in solving many hard optimization problems. However, the task of developing a highly effective hybrid approach is not easy at all. Nevertheless, there are several hybridization types that have proven successful on many occasions, and they can provide some guidance.

The growing popularity of this line of research is documented by rather recent conferences and workshops such as *CPAIOR* [63], *Hybrid Metaheuristics* [6, 7], and *Matheuristics* [39]. Moreover, the first book specifically devoted to hybrid metaheuristics has recently been published in 2008 [11]. In this brief survey, we provide an overview of hybrid metaheuristics by illustrating prominent and paradigmatic examples, which range from the integration of metaheuristic techniques among them-

selves, to the hybridization of metaheuristics with constraint and mathematical programming. The interested reader can find other reviews on hybrid metaheuristics in [11, 17, 20, 51].

2. Examples and Literature Overview

In our opinion, the current body of research on hybrid metaheuristics can be subdivided into five different categories, namely, the hybridization of metaheuristics with (meta-)heuristics, constraint programming, tree search methods, problem relaxations, and dynamic programming. Each of these five categories is treated below in its own subsection. For each category we will list representative works.

2.1 Hybridization of Metaheuristics with (Meta-)Heuristics

The hybridization of metaheuristics with (meta-)heuristics is quite popular, especially for what concerns the use of local search methods inside population-based methods. Indeed, most of the successful applications of evolutionary computation and ant colony optimization make use of local search procedures for refining the generated solutions. This is because the major strength of population-based methods is their exploration capability. At the start of the search they generally try to capture a global picture of the search space, and typically, rather simple and problem-dependent operations are then iteratively applied to derive diverse new solutions successively, focusing the search on promising regions of the search space. Conversely, the strength of local search methods is their rather fast intensification capability, that is, the capability of quickly finding better solutions in the vicinity of given starting solutions. In summary, population-based methods are good in identifying promising areas of the search space in which local search methods can then quickly determine the best solutions. Therefore, this type of hybridization is often very successful. In the field of evolutionary algorithms these hybrids even carry their own name, *memetic algorithms* [35].

Apart from the usual above-mentioned hybridization, this category contains, for example, so-called *multi-level* techniques [65, 66]. They are heuristic frameworks with the potential of making the search process of a metaheuristic more effective and efficient. The basic idea of a multilevel technique is the one of coarsening a given problem instance. Then, the problem is solved on the coarsened instance and the obtained solution is transformed in order to obtain a solution to the original instance. *Variable fixing* strategies [46, 50] are related techniques.

Another hybrid in this first category are *hyper-heuristics* [14]. They work on a higher level than classical metaheuristics, in the sense that they do not directly operate on the search space of the problem under consideration. Instead they operate on a search space consisting of lower-level heuristics—or even metaheuristics—for the tackled problem. Hyper-heuristics are broadly concerned with selecting the right metaheuristic at any situation.

Interestingly, in recent years a few examples have appeared for the use of components from population-based methods within metaheuristics based on local search. One of these examples concerns *population-based iterated local search* [59] where iterated local search is extended from working on a single solution to working on a population that is managed in the style of evolution strategies. In a different example, Lozano and García-Martínez [37] advocate the use of an evolutionary algorithm as a perturbation technique within iterated local search, while Resende et al. [54] devise several versions of a hybrid algorithm based on GRASP and path relinking methodologies.

An important branch of hybridization is the enhancement of metaheuristics with additional techniques for improving run-time, results, or both. Montemanni and Smith [41] propose an algorithm to solve the frequency assignment problem that is based on tabu search. Hereby, tabu search is enhanced by *heuristic manipulation*, a mechanism based on the idea that adding constraints to a problem results in a search space reduction, which, in turn, may facilitate the solution of the problem.

2.2 Hybridization of Metaheuristics with Constraint Programming

Constraint programming (CP) is a programming paradigm that is build upon constraints and constraint solving [38]. CP is generally said to be particularly effective in finding feasible solutions to highly constrained problems. On the other side, metaheuristics are generally very effective in finding good-quality solutions to mildly constrained optimization problems while requiring a limited amount of computational resources. In turn, metaheuristics are generally not very effective in tackling highly constrained problems, while CP alone usually does not achieve a particularily high performance in solving loosely constrained optimization problems. Given these considerations, the combination of metaheuristics with CP seems applicable to problems with a fairly high number of constraints and, at the same time, a sufficiently large number of feasible solutions [40].

The integration of (meta)heuristics and CP dates back to the late 1990s; see, for example, the works by Pesant and Gendreau [43, 44] and subsequent works [19, 58]. A survey on possible ways of integrating metaheuristics and CP is provided by Focacci et al. in [24]. With a bit of oversimplification, four main approaches for the integration of metaheuristics and CP can be identified:

1 Metaheuristics are applied before CP, providing a valuable input, or vice versa.

2 Metaheuristics, mainly local search methods, use CP to efficiently explore the neighborhood of the current solution.

3 Construction-based metaheuristics use CP in order to prune the search space.

4 CP applies a metaheuristic in order to improve a solution (i.e., a leaf of the search tree) or a partial solution (i.e., an inner node). Metaheuristic concepts can also be used to obtain incomplete but efficient tree exploration strategies.

The first one of these approaches represents a rather loose hybridization and can be seen as an instance of *cooperative search* [22]. The second approach combines the advantages of a fast search space exploration by means of a metaheuristic with the efficient neighborhood exploration performed by a systematic method. A prominent example of such a kind of integration is large neighborhood search and related approaches [15, 57]. The third approach has found applications especially in ant colony optimization [34, 40, 61]. Hereby, CP is used at each solution construction step to filter the available options for the extension of the current partial solution. The fourth approach preserves the search space exploration based on systematic search (such as tree search), but sacrifices the exhaustive nature of the search [28, 29]. The hybridization is usually achieved by integrating concepts developed in the context of metaheuristics (e.g., probabilistic choices, aspiration criteria, heuristic construction) into tree search methods. For example, instead of a chronological backtracking, a backjumping based on search history or information retrieved from local search samples can be performed. Other examples of this approach can be found in [48, 56].

2.3 Hybridizing Metaheuristics with Tree Search

Optimization techniques can be characterized by their way of exploring the search space. Some algorithms consider the search space of an

optimization problem in form a of a tree, the so-called *search tree*, which is generally defined by an underlying solution construction mechanism. Each path from the root node of the search tree to one of the leaves corresponds to a step-by-step construction of a candidate solution. Inner nodes of the tree are partial solutions to the given problem. The process of moving from an inner node to one of its child nodes is called a solution construction step, or an extension of a partial solution.

The class of tree search algorithms comprises approximate methods such as ant colony optimization and GRASP, but also complete techniques such as branch & bound and heuristic variants of complete techniques such as beam search. Therefore, the hybridization of metaheuristics with other tree search techniques is probably one of the most popular hybridization approaches. One of the first works on a combination of branch & bound with an evolutionary algorithm is the one by Nagar et al [42]. Hereby, an incomplete execution of branch & bound is used to guide the working of an evolutionary algorithm.

Exact tree search methods have been used quite a few times in *solution merging*, which is based on the idea of deriving new and hopefully better solutions from the attributes originating from two ore more input solutions. Applegate et al. [2, 3] were among the first to apply tree search methods in the context of merging. In an application for the travelling salesman problem they merge solutions and produce a (potentially) new solution by solving the resulting reduced graph to optimality.

Similarly as constraint programming is sometimes used for searching large neighborhoods (see Section 2.2), other tree search methods are also utilized for this purpose. Especially branch & bound techniques based on linear programming, including branch-and-cut, are often a promising option when the problem at hand can be expressed by a mixed integer programming (MIP) model. The availability of highly effective general purpose MIP solvers, which are typically based on sophisticated branch-and-cut frameworks but nevertheless can be relatively easily applied, makes this approach particularly interesting in practice. In the literature, numerous successful examples exist for such approaches. Among the more generally applicable ones is *local branching* [23]. A successful problem-specific example for large neighborhood search by means of solving sub-MIPs via branch-and-cut has been described by Prandtstetter and Raidl for the car sequencing problem [47].

Instead of using an exact method within a metaheuristics, the literature also offers examples where incomplete versions of exact methods are used to enhance metaheuristics. One of these examples is Beam-ACO [8, 9], which is a hybrid algorithm that combines ant colony optimization with beam search. This algorithm employs parallel and non-

independent solution constructions at each iteration, in the style of beam search. Another example is the hybridization of metaheuristics with backtracking. In [30] the authors describe applications of various hybrid metaheuristics to problems ranging from car sequencing and graph coloring to scheduling. For example, a tabu search algorithm for the job shop scheduling problem is presented, combining local search with complete enumeration as well as limited backtracking search.

The examples mentioned above are characterized by a subordinate use of an exact method within the metaheuristic. However, the literature also offers examples where metaheuristics are used for guiding the search process of an exact technique, or a heuristic derivate. For example, Rothberg [55] suggests a tight integration of an evolutionary algorithm in a MIP solver based on branch & cut. The evolutionary algorithm is applied at regular intervals as a branch & bound tree node heuristic. Another example concerns the works presented in [12, 25], where the applications of beam search and a memetic algorithm are intertwined. More specifically, phases of beam search and the memetic algorithm alternate. Beam search purges its queue of open partial solutions by excluding those ones whose upper bounds are worse than the value of the best solution found by the memetic algorithm. On the other side, beam search guides the search of the memetic algorithm by injecting information about promising regions of the search space into the population. Another example where metaheuristics may be used as a subordinate technique is *diving* [18], which is a mechanism for focusing the search process of branch & bound in an initial phase to neighborhoods of promising incumbents in order to quickly identify high-quality solutions.

2.4 Hybridization of Metaheuristics with Problem Relaxation

Guiding metaheuristics by *problem relaxation* has become quite popular in recent years. A so-called relaxed problem is obtained by simplifying or omitting constraints from the original problem formulation. The hope is, first, that the relaxed problem can be efficiently solved, and second, that the structure of an optimal solution to the relaxed problem together with its objective function value can be used in some way for solving the original problem. For example, the optimal solution value of a relaxed problem can be seen as a bound for the optimal solution value of the original problem. Therefore, it can be used in a branch & bound algorithm for discarding parts of the search tree. An important type of relaxation in combinatorial optimization concerns dropping the

integrality constraints of the involved variables from a MIP formulation. The resulting linear programming (LP) relaxation can then be solved to optimality by efficient methods like the well-known *Simplex algorithm*.

One of the most obvious ways to utilize an optimal solution to the LP relaxation of a problem at hand is to directly derive a heuristic integer solution which is feasible for the original problem. Depending on the problem, this can be achieved by simple rounding or more sophisticated repairing strategies. For example, Raidl and Feltl [52] present a hybrid genetic algorithm for the generalized assignment problem. The LP relaxation of the problem is solved and its solution is exploited by a randomized rounding procedure to create an initial population of promising integral solutions.

In [64], Vasquez and Hao present a two-phase approach for the multidimensional 0-1 knapsack problem (MKP). This algorithm is a prime example for a hybrid metaheuristic guided by problem relaxation. The main idea consists in solving a number of relaxed problems obtained by dropping the integrality constraints to optimality. This is done in a first phase. Afterwards, in a second phase, tabu search is used to search around the optimal solutions to the relaxed problems. Another examples concerns the work by Puchinger and Raidl [49]. They introduced *relaxation guided variable neighborhood search* (RGVNS). The main algorithmic framework of RGVNS is variable neighborhood search. However, neighborhoods are dynamically ordered according to so-called *improvement-potentials*. These estimates are determined by computing bounds on the objective function values of the optimal solutions within each neighborhood. Such bounds are obtained by solving a relaxation of the original problem.

A successful example for using other relaxation techniques is the hybrid Lagrangian GA for the prize collecting Steiner tree problem by Haouari and Siala [27]. Hereby, the GA uses results from the previously solved Lagrangian relaxation. In particular, the original graph is reduced by discarding edges, meaningful initial solutions are generated, and the objective function is modified by considering reduced costs.

For the knapsack constrained maximum spanning tree problem, a similar combination of Lagrangian decomposition and a genetic algorithm is described in Pirkwieser et al. [45]. A combination of a Lagrangian relaxation approach and a variable neighborhood descent metaheuristic, which is also based on similar principles, has recently been developed for a real-world fiber optic network design problem by Leitner and Raidl [36].

Tamura et al. [60] propose an algorithm that works by first executing a GA for identifying a promising region of the search space. The fitness of solutions is hereby related to Lagrangian relaxations. Afterwards, an

exhaustive search is used to find the best solution within the identified region.

Finally, Reimann [53] introduces an ant colony optimization algorithm for the symmetric travelling salesman problem where an optimal solution to the minimum spanning tree relaxation is used for biasing the search of the artificial ants towards edges that form part of the minimum spanning tree.

3. Hybridization of Metaheuristics with Dynamic Programming

Dynamic programming (DP) is another example of an optimization method from operations research and control theory that can be successfully integrated with metaheuristics, both in the case of constructive and local search techniques. DP provides a method for defining an optimal strategy that leads from an initial state to the final goal and it has been successfully applied to many optimization and control problems [5].

Iterated dynasearch is a hybrid metaheuristic that uses DP as a neighborhood exploration strategy inside iterated local search [31]. The rationale behind this integration is that in neighborhood search, the larger the neighborhood size, the better the quality of the local optimum returned (on average). Suitable neighborhoods are often of exponential size, making it impractical to perform an explicit exhaustive lexicographical enumeration. Therefore, more computationally efficient neighborhood exploration techniques are required. In some cases, DP can make it possible to completely explore an exponential size neighborhood in polynomial time and space [1, 16].

In [10], Blum and Blesa present the use of a DP algorithm within two different metaheuristics for the k-cardinality tree (KCT) problem. The general idea of their approaches is not limited to the KCT problem and can, potentially, be used for other subset problems. Basically, the idea is to let the metaheuristic generate objects that are bigger than solutions, containing in general an exponential number of solutions to the problem under consideration. DP is then used to efficiently find for each object the best solution that it contains.

Hu and Raidl [32] use DP within an evolutionary algorithm as a mechanism for generating the best solution that can be obtained from an incomplete solution. A somewhat related approach is presented in [21]. In [33], DP is used purely as a decoder for tackling the rectangle packing problem with general spatial costs, which consists in packing given rectangles without overlap in the plane so that the maximum cost of the rectangles is minimized.

The following examples deal with hybridizations based on problem decompositions. In [67], the authors propose a hybrid method that combines adaptive memory, sparse DP, and reduction techniques to reduce and explore the search space. The first step consists in the generation of a bi-partition of the variables. The resulting small problem is solved using the forward-phase of DP. The space defined by the remaining variables is explored using tabu search. Hereby, each partial solution is completed with the information stored during the forward phase of DP. The application of DP to subproblems is also proposed in [62]. The presented local search technique called *iterative dynamic programming* works by subdividing the problem into subproblems, and optimizing the subproblems separately by DP.

Finally, we would like to point out an interesting heuristic version of DP known as *bounded dynamic programming* in which at each level the number of states is heuristically reduced. In this way, the authors of [4] were able to find most optimal solutions to benchmark instances of the simple assembly line balancing problem in a reduced amount of computation time.

4. Conclusions

Research on hybrid metaheuristics is still in its early stages. However, we are convinced that, in the years to come, most publications on metaheuristic applications will be concerned with hybrids. Nevertheless, the process of designing and implementing hybrid metaheuristics is rather complicated and involves knowledge about a broad spectrum of algorithmic techniques, programming and data structures, as well as algorithm engineering and statistics. In fact, it is hardly possible to provide guidelines for the successful development of hybrid metaheuristics. However, in the process of developing a hybrid metaheuristic it is indispensable (1) to carefully search the literature for the most successful optimization approaches for the problem at hand or for similar problems, and (2) the study of different ways of combining the most promising features of the identified approaches. We hope that this paper may serve as a starting point for this purpose.

Acknowledgements

This work was supported by grant TIN2007-66523 (FORMALISM) of the Spanish government. Moreover, Christian Blum acknowledges support from the *Ramón y Cajal* program of the Spanish Ministry of Science and Innovation.

References

[1] E. Angel and E. Bampis. A multi-start dynasearch algorithm for the time dependent single-machine total weighted tardiness scheduling problem. *Eur. J. Oper. Res.*, 162(1):281–289, 2005.

[2] D. L. Applegate, R. E. Bixby, V. Chvátal, and W. J. Cook. On the solution of the traveling salesman problem. *Doc. Math.*, Extra Vol. ICM III:645–656, 1998.

[3] D. L. Applegate, R. E. Bixby, V. Chvátal, and W. J. Cook. *The Traveling Salesman Problem: A Computational Study.* Princeton Series in Applied Mathematics. Princeton University Press, 2007.

[4] J. Bautista and J. Pereira. A dynamic programming based heuristic for the assembly line balancing problem. *Eur. J. Oper. Res.*, 194(3):787–794, 2009.

[5] D. P. Bertsekas. *Dynamic Programming and Optimal Control.* Athena Scientific, Nashua, NH, 3rd edition, 2007.

[6] M. J. Blesa, C. Blum, L. Di Gaspero, A. Roli, M. Sampels, and A. Schaerf (Eds). *Proceedings of HM 2009 – Sixth International Workshop on Hybrid Metaheuristics. Lect. Notes Comput. Sc.*, 5818, 2009.

[7] M. J. Blesa Aguilera, C. Blum, C. Cotta, A. J. Fernández, J. E. Gallardo, A. Roli, and M. Sampels (Eds). *Proceedings of HM 2008 – Fifth International Workshop on Hybrid Metaheuristics. Lect. Notes Comput. Sc.*, 5296, 2008.

[8] C. Blum. Beam-ACO–hybridizing ant colony optimization with beam search: an application to open shop scheduling. *Comput. Oper. Res.*, 32(6):1565–1591, 2005.

[9] C. Blum. Beam-ACO for simple assembly line balancing. *INFORMS J. Comput.*, 20(4):618–627, 2008.

[10] C. Blum and M. J. Blesa. Solving the KCT problem: Large-scale neighborhood search and solution merging. In E. Alba, C. Blum, P. Isasi, C. León, and J. A. Gómez (Eds). *Optimization Techniques for Solving Complex Problems*, pages 407–421. Wiley & Sons, Hoboken, NJ, 2009.

[11] C. Blum, M. J. Blesa Aguilera, A. Roli, and M. Sampels, editors. *Hybrid Metaheuristics – An Emerging Approach to Optimization*, volume 114 of *Studies in Computational Intelligence*. Springer, 2008.

[12] C. Blum, C. Cotta, A. J. Fernández, J. E. Gallardo, and M. Mastrolilli. Hybridization of metaheuristics with branch & bound derivatives. In C. Blum, M. J. Blesa Aguilera, A. Roli, and M. Sampels (Eds). *Hybrid Metaheuristics – An Emerging Approach to Optimization. Studies in Computational Intelligence*, 114:85–116. Springer, 2008.

[13] C. Blum and A. Roli. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Comput. Surv.*, 35(3):268–308, 2003.

[14] E. K. Burke, G. Kendall, J. Newall, E. Hart, P. Ross, and S. Schulenburg. Hyper-heuristics: An emerging direction in modern search technology. In F. Glover and G. Kochenberger (Eds). *Handbook of Metaheuristics. International Series in Operations Research & Management Science*, 57:457–474. Kluwer Academic Publishers, 2003.

[15] Y. Caseau and F. Laburthe. Effective Forget-and-Extend heuristics for scheduling problems. In *Proc. Fourth International Workshop on Integration of AI*

and OR techniques in Constraint Programming for Combinatorial Optimization Problems, 1999.

[16] R. K. Congram, C. N. Potts, and S. L. van de Velde. An iterated dynasearch algorithm for the single-machine total weighted tardiness scheduling problem. *INFORMS J. Comput.*, 14(1):52–67, 2002.

[17] C. Cotta. A study of hybridisation techniques and their application to the design of evolutionary algorithms. *AI Commun.*, 11(3–4):223–224, 1998.

[18] E. Danna, E. Rothberg, and C. Le Pape. Exploring relaxation induced neighborhoods to improve MIP solutions. *Math. Program., Ser. A*, 102(1):71–90, 2005.

[19] B. De Backer, V. Furnon, and P. Shaw. Solving vehicle routing problems using constraint programming and metaheuristics. *J. Heuristics*, 6(4):501–523, 2000.

[20] I. Dumitrescu and T. Stuetzle. Combinations of local search and exact algorithms. *Lect. Notes Comput. Sc.*, 2611:211–223, 2003.

[21] T. Dunker, G. Radons, and E. Westkämper. Combining evolutionary computation and dynamic programming for solving a dynamic facility layout problem. *Eur. J. Oper. Res.*, 165(1):55–69, 2005.

[22] Mohammed El-Abd and Mohamed Kamel. A taxonomy of cooperative search algorithms. *Lect. Notes Comput. Sc.*, 3636:32–41, 2005.

[23] M. Fischetti and A. Lodi. Local Branching. *Math. Program., Ser. B*, 98(1–3):23–47, 2003.

[24] F. Focacci, F. Laburthe, and A. Lodi. Local search and constraint programming. In F. Glover and G. Kochenberger (Eds). *Handbook of Metaheuristics. International Series in Operations Research & Management Science*, 57:369–403. Kluwer Academic Publishers, 2003.

[25] J. E. Gallardo, C. Cotta, and A. J. Fernández. On the hybridization of memetic algorithms with branch-and-bound techniques. *IEEE T. Syst. Man Cy. B*, 37(1):77–83, 2007.

[26] F. Glover and G. Kochenberger (Eds). *Handbook of Metaheuristics. International Series in Operations Research & Management Science*, 57. Kluwer Academic Publishers, 2003.

[27] M. Haouari and J. C. Siala. A hybrid Lagrangian genetic algorithm for the prize collecting Steiner tree problem. *Comput. Oper. Res.*, 33(5):1274–1288, 2006.

[28] W. D. Harvey. *Nonsystematic Backtracking Search*. PhD thesis, CIRL, University of Oregon, Eugene, Orgeon, 1995.

[29] W. D. Harvey and M. L. Ginsberg. Limited discrepancy search. In *Proc.14th International Joint Conference on Artificial Intelligence (IJCAI)*, volume 1, pages 607–615. Morgan Kaufmann Publishers, San Mateo, CA, 1995.

[30] P. Van Hentenryck and L. Michel. *Constraint-Based Local Search*. MIT Press, Cambridge, MA, 2005.

[31] H. Hoos and T. Stützle. *Stochastic Local Search – Foundations and Applications*. Morgan Kaufmann Publishers, 2005.

[32] B. Hu and G. R. Raidl. Effective neighborhood structures for the generalized traveling salesman problem. *Lect. Notes Comput. Sc.*, 4972:36–47, 2008.

[33] S. Imahori, M. Yagiura, and T. Ibaraki. Improved local search algorithms for the rectangle packing problem with general spatial costs. *Eur. J. Oper. Res.*, 167(1):48–67, 2005.

[34] M. Khichane, P. Albert, and C. Solnon. Integration of ACO in a constraint programming language. *Lect. Notes Comput. Sc.*, 5217:84–95, 2008.

[35] N. Krasnogor and J. Smith. A tutorial for competent memetic algorithms: model, taxonomy, and design issues. *IEEE T. Evol. Comput.*, 9(5):474–488, 2005.

[36] M. Leitner and G. R. Raidl. Lagrangian decomposition, metaheuristics, and hybrid approaches for the design of the last mile in fiber optic networks. *Lect. Notes Comput. Sc.*, 5296:158–174, 2008.

[37] M. Lozano and C. García-Martínez. Hybrid metaheuristics with evolutionary algorithms specializing in intensification and diversification: Overview and progress report. *Comput. Oper. Res.*, 37(3):481–497, 2010.

[38] K. Marriott and P. J. Stuckey. *Introduction to Constraint Logic Programming.* MIT Press, Cambridge, MA, 1998.

[39] Matheuristics 2008. http://astarte.csr.unibo.it/matheuristics2008/. Visited in April 2009.

[40] B. Meyer. Hybrids of constructive meta-heuristics and constraint programming: A case study with ACO. In C. Blum, M. J. Blesa Aguilera, A. Roli, and M. Sampels (Eds). *Hybrid Metaheuristics – An Emerging Approach to Optimization. Studies in Computational Intelligence*, 114:151–183. Springer, 2008.

[41] R. Montemanni and D. H. Smith. Heuristic manipulation, tabu search and frequency assignment. *Comput. Oper. Res.*, 37(3):543–551, 2010.

[42] A. Nagar, S. S. Heragu, and J. Haddock. A meta-heuristic algorithm for a bi-criteria scheduling problem. *Ann. Oper. Res.*, 63:397–414, 1995.

[43] G. Pesant and M. Gendreau. A view of local search in Constraint Programming. *Lect. Notes Comput. Sc.*, 1118:353–366, 1996.

[44] G. Pesant and M. Gendreau. A Constraint Programming Framework for Local Search Methods. *J. Heuristics*, 5(3):255–279, 1999.

[45] S. Pirkwieser, G. R. Raidl, and J. Puchinger. Combining Lagrangian decomposition with an evolutionary algorithm for the knapsack constrained maximum spanning tree problem. *Lect. Notes Comput. Sc.*, 4446:176–187, 2007.

[46] D. Pisinger. Core problems in knapsack algorithms. *Oper. Res.*, 47(4):570–575, 1999.

[47] M. Prandtstetter and G. R. Raidl. An integer linear programming approach and a hybrid variable neighborhood search for the car sequencing problem. *Eur. J. Oper. Res.*, 191(3):1004–1022, 2008.

[48] S. Prestwich. Combining the scalability of local search with the pruning techniques of systematic search. *Ann. Oper. Res.*, 115(1–4):51–72, 2002.

[49] J. Puchinger and G. R. Raidl. Bringing order into the neighborhoods: Relaxation guided variable neighborhood search. *J. Heuristics*, 14(5):457–472, 2008.

[50] J. Puchinger, G. R. Raidl, and U. Pferschy. The core concept for the multidimensional knapsack problem. *Lect. Notes Comput. Sc.*, 3906:195–208, 2006.

[51] G. R. Raidl. A unified view on hybrid metaheuristics. *Lect. Notes Comput. Sc.*, pages 4030:1–12, 2006.

[52] G. R. Raidl and H. Feltl. An improved hybrid genetic algorithm for the generalized assignment problem. In *Proc. 2003 ACM Symposium on Applied Computing*, pages 990–995. ACM Press, 2004.

[53] M. Reimann. Guiding ACO by problem relaxation: A case study on the symmetric TSP. *Lec. Notes Comput. Sc.*, 4771:45–55, 2007.

[54] M. G. C. Resende, R. Martí, M. Gallego, and A. Duarte. GRASP and path relinking for the max–min diversity problem. *Comput. Oper. Res.*, 37(3):498–508, 2010.

[55] E. Rothberg. An evolutionary algorithm for polishing mixed integer programming solutions. *INFORMS J. Comput.*, 19(4):534–541, 2007.

[56] A. Schaerf. Combining local search and look-ahead for scheduling and constraint satisfaction problems. In *Proc. 15th International Joint Conference on Artificial Intelligence*, pages 1254–1259, 1997.

[57] P. Shaw. Using constraint programming and local search methods to solve vehicle routing problems. *Lect. Notes Comput. Sc.*, 1520:417–431, 1998.

[58] P. Shaw, B. De Backer, and V. Furnon. Improved local search for CP toolkits. *Ann. Oper. Res.*, 115(1–4):31–50, 2002.

[59] T. Stützle. Iterated local search for the quadratic assignment problem. *Eur. J. Oper. Res.*, 174(3):1519–1539, 2006.

[60] H. Tamura, A. Hirahara, I. Hatono, and M. Umano. An approximate solution method for combinatorial optimisation. *T. Soc. Instr. Contr. Eng.*, 130:329–336, 1994.

[61] D. R. Thiruvady, C. Blum, B. Meyer, and A. T. Ernst. Hybridizing Beam-ACO with constraint programming for single machine job scheduling. *Lect. Notes Comput. Sc.*, 5818:30–44, 2009.

[62] S.-M. Tse, Y. Liang, K.-S. Leung, K.-H. Lee, and T. S.-K. Mok. A memetic algorithm for multiple-drug cancer chemotherapy schedule optimization. *IEEE T. Syst. Man Cy. B*, 37(1):84–91, 2007.

[63] W. J. van Hoeve and J. N. Hooker, editors. *Proc. 6th International Conference on the Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems (CPAIOR). Lect. Notes Comput. Sc.*, 5547, 2009.

[64] M. Vasquez and Y. Vimont. Improved results on the 0–1 multidimensional knapsack problem. *Eur. J. Oper. Res.*, 165(1):70–81, 2005.

[65] C. Walshaw. Multilevel refinement for combinatorial optimization problems. *Ann. Oper. Res.*, 131(1–4):325–372, 2004.

[66] C. Walshaw. Multilevel refinement for combinatorial optimisation: Boosting metaheuristic performance. In C. Blum, M. J. Blesa Aguilera, A. Roli, and M. Sampels (Eds). *Hybrid Metaheuristics – An Emerging Approach to Optimization. Studies in Computational Intelligence*, 114:261–289. Springer, 2008.

[67] C. Wilbaut, S. Hanafi, A. Fréville, and S. Balev. Tabu search: global intensification using dynamic programming. *Control Cybern.*, 35(3):579–598, 2009.

II

THEORY AND ALGORITHMS

PARALLEL DIFFERENTIAL EVOLUTION WITH ENDEMIC RANDOMIZED CONTROL PARAMETERS

Matthieu Weber, Ferrante Neri*, Ville Tirronen
Department of Mathematical Information Technology
University of Jyväskylä, Finland
{matthieu.weber; neferran; ville}.tirronen@jyu.fi

Abstract This paper proposes the use of endemic control parameters within a Parallel Differential Evolution algorithm. The Differential Evolution running at each subpopulation is associated with randomly initialized scale factor and crossover rate, which are then repeatedly updated during the optimization process. Numerical results show that the Endemic Randomized Control Parameter Parallel Differential Evolution seems to be a simple, robust, and efficient algorithm suited for various applications. An important finding of this study is that randomized initial values of both control parameters and repeated updates of the scale factor are beneficial to the optimization process, whereas repeated updates of the crossover rate are detrimental.

Keywords: Differential Evolution, Parallel Evolutionary Algorithm, Self-adaptation

1. Introduction

Differential Evolution (DE), see [8], is an optimization algorithm which has shown high performance in various types of applications particularly when applied to continuous problems, for example [11]. DE is implicitly self-adaptive (see [3]), which allows it to extensively explore the problem space during the early stages of the evolution and progressively narrow the search within the most promising areas of the decision space. Although this mechanism is effective, there is a hidden drawback: the DE contains a limited amount of search moves and the population could fail at enhancing upon the available genotypes, thus resulting in a stagnation

*This work is supported by Academy of Finland, Akatemiatutkija 130600, Algorithmic Design Issues in Memetic Computing.

condition. In order to overcome this drawback, computer scientists in recent years have attempted to improve the DE performance by modifying the basic DE. Some popular examples of this scientific trend can be found in [9] where multiple search strategies are employed, in [2] where the offspring are generated by combining two mating pools (one global and one local, respectively), and in [1] where a randomization of the parameters increases the variability of the potential search moves. Another popular method of enhancing the DE performance is through employment of structured populations. In [6], a distributed DE scheme employing a ring topology (the cores are interconnected in a circle and the migrations occur following the ring) has been proposed for the training of a neural network. [10] proposes a distributed DE characterized by a ring topology and the migration of individuals with the best performance to replace random individuals of the neighbor subpopulation; an application of this algorithm for training of a neural network has been presented in [7].

This paper focuses on distributed DE and in particular on the ring topology scheme presented in [10]. The main novelty in Parallel Differential Evolution (PDE) described in [10] consists of the migration scheme and the related probability: the DE is independently performed on each subpopulation composing the ring and, at the end of each generation, with a certain probability the individual with the best performance is copied into the neighbor subpopulation and replaces a randomly selected individual from the target subpopulation. In [10], a compromise value of migration probability is proposed. In this paper we propose a novel algorithm based on PDE, namely Endemic Randomized Control Parameters Parallel Differential Evolution. Instead of the fixed values of the control parameters used in PDE, ERCPDE uses pseudo-randomly generated values, inspired by the Self-Adapting Control Parameters method described in [1]. Scale factors and crossover rates are endemic (local) to each subpopulation and they are updated over time according to a probabilistic scheme.

The remainder of this article is organized in the following way. Section 2 describes the working principles of DE, PDE and ERCPDE. Section 3 shows the experimental setup and numerical results of the present study. Section 4 gives the conclusions of this paper.

2. Endemic Randomized Control Parameters Parallel Differential Evolution

In order to clarify the notation used throughout this paper we refer to the minimization problem of an objective function $f(x)$, where x is a vector of n design variables in a decision space D.

At the beginning of the optimization process S_{pop} individuals are pseudo-randomly sampled with a uniform distribution function within the decision space D (for simplicity, the term random will be used instead of pseudo-random in the reminder of this paper). The S_{pop} individuals constituting the populations are distributed over m subpopulations P_k, $k = 1, \ldots, m$, organized into a unidirectional ring. Each subpopulation is therefore composed of S_{pop}/m individuals.

Within each subpopulation a original DE, following its original definition, is performed. At each generation, for each individual x_i of the S_{pop}, three individuals x_r, x_s and x_t are randomly extracted from the population. According to the DE logic, a provisional offspring x'_{off} is generated by mutation as:

$$x'_{off} = x_t + F(x_r - x_s) \tag{1}$$

where $F \in [0, 1+]$ is a scale factor which controls the length of the exploration vector $(x_r - x_s)$ and thus determines how far from point x_i the offspring should be generated. With $F \in [0, 1+]$, it is meant here that the scale factor should be a positive value which cannot be much greater than 1, see [8]. While there is no theoretical upper limit for F, effective values are rarely greater than 1.0. The mutation scheme shown in Equation (1) is also known as DE/rand/1. It is worthwhile mentioning that there exist many other mutation variants, see [9].

When the provisional offspring has been generated by mutation, each gene of the individual x'_{off} is exchanged with the corresponding gene of x_i with a uniform probability and the final offspring x_{off} is generated:

$$x_{off,j} = \begin{cases} x'_{off,j} & \text{if} \quad rand\,(0,1) \leq CR \\ x_{i,j} & \text{otherwise} \end{cases} \tag{2}$$

where $rand\,(0, 1)$ is a random number between 0 and 1; j is the index of the gene under examination.

The resulting offspring x_{off} is evaluated and, according to a one-to-one spawning strategy, it replaces x_i if and only if $f(x_{off}) \leq f(x_i)$; otherwise no replacement occurs. It must be remarked that although the replacement indexes are saved one by one during generation, actual replacements occur all at once at the end of the generation.

In PDE, each subpopulation P_k runs the DE algorithm described above. On each generation, the subpopulation has a given probability ϕ to send a copy of its best individual to its next neighbor subpopulation in the ring. When a migration occurs, the migrating individual replaces a randomly selected individual belonging to the target subpopulation, with an exception being made of the subpopulation's best performing

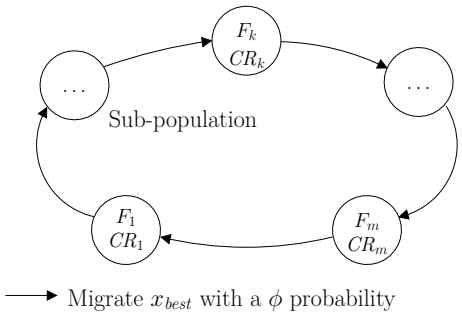


Figure 1. Working principle of the ERCPDE

individual, which can never be replaced. For the sake of clarity a scheme highlighting the working principles of ERCPDE is shown in Fig. 1.

In PDE, the control parameters of DE (namely the scale factor F and the crossover rate CR) are global for all the subpopulations, meaning that whichever subpopulation an individual x belongs to at a given time the same values of F and CR are used when the DE algorithm is applied to it. In ERCPDE however, each subpopulation P_k has its own scale factor F_k and crossover rate CR_k. Thus, when an individual x resides within population P_k, DE is applied to it using the local F_k and CR_k control parameters. During the initialization phase of the algorithm, the values of F_k and CR_k are randomly set in each subpopulation, such that $F_k \in [F_l, F_l + F_u]$ and $CR_k \in [0, 1]$. Additionally, the values of F_k and CR_k of each subpopulation vary in time; more precisely, on each generation F_k and CR_k have a probability to be updated of τ_1 and τ_2, respectively, see [1]:

$$F_k = \begin{cases} F_l + F_u \times rand_1, & \text{if} \quad rand_2 < \tau_1 \\ F_k, & \text{otherwise} \end{cases} \tag{3}$$

$$CR_k = \begin{cases} rand_3, & \text{if} \quad rand_4 < \tau_2 \\ CR_k, & \text{otherwise} \end{cases} \tag{4}$$

where $rand_j$, $j \in \{1, 2, 3, 4\}$, are uniform pseudo-random values between 0 and 1; τ_1 and τ_2 are constant values which represent the probabilities that parameters are updated, F_l and F_u are constant values which represent the minimum value that F could take and the maximum variable contribution to F, respectively.

To understand the rationale behind the proposed mechanism, it is important to analyze the concept of parallelism and migration in a PDE scheme. In the classical DE, for each stage of the optimization process the algorithm can generate only a limited amount of exploratory moves.

If these moves are not enough for generating new promising solutions, stagnation occurs as the algorithm does not manage to improve upon any solution of its population for a prolonged number of generations.

The use of multiple populations in distributed DE algorithms allows an observation of the decision space from various perspectives and, most importantly, decreases the risk of stagnation. In addition, the migration mechanism ensures that solutions with a high performance are introduced into the subpopulations during their evolution, which modifies the set of search moves and promote detection of new promising search directions. Thus, the migration is supposed to mitigate the risk of stagnation of the DE subpopulations and to enhance the global algorithmic performance.

Since the number of search moves allowed to an individual depends not only on other individuals in the population, but also on the values of the scale factor and crossover rate, allowing for different values of these two control parameters within the framework of a structured population such as the one used by PDE leads to a higher number of possible search moves, which increases the explorative capacity of the algorithm and leads to a quicker improvement. This paper proposes to achieve such a setup by assigning multiple values of the scale factor and the crossover rate to each subpopulation in PDE and by updating them over time so as to yet increase the number of search moves a given individual is allowed.

Table 1. Test Problems

Test Problem	**Function**	**Decision Space**	**Optimum**
Ackley	$-20 + e + 20 \exp\left(-\frac{0.2}{n}\sqrt{\sum_{i=1}^{n} x_i^2}\right)$ $- \exp\left(\frac{1}{n}\sum_{i=1}^{n} \cos(2\pi \cdot x_i) x_i\right)$	$[-1, 1]^n$	0
Alpine	$\sum_{i=1}^{n} \lvert x_i \sin x_i + 0.1 x_i \rvert$	$[-10, 10]^n$	0
Sphere	$\lVert x \rVert^2$	$[-5.12, 5.12]^n$	0
Michalewicz	$-\sum_{i=1}^{n} \sin x_i \left(\sin\left(\frac{i \cdot x_i^2}{\pi}\right)\right)^{20}$	$[0, \pi]^n$	unknown
Rastrigin	$10n + \sum_{i=0}^{n} \left(x_i^2 - 10\cos(2\pi x_i)\right)$	$[-5.12, 5.12]^n$	0
Schwefel	$-\sum_{i=1}^{n} x_i \sin\left(\sqrt{\lvert x_i \rvert}\right)$	$[-500, 500]^n$	$-418.9829n$

3. Experimental Results

The test problems listed in Table 1 have been considered in this study.

The rotated version of some of the test problems listed in Table 1 have been included into the benchmark set. These rotated problems have been generated through the multiplication of the vector of variables by a randomly generated sparse orthogonal rotation matrix, created by composing n rotations of random angles (uniformly sampled in $[-\pi, \pi]$), one around each of the n axes of the search space. In total, ten test problems have been considered in this study with $n = 500$.

Table 2. Average Fitness $\pm$ standard deviation at the end of the optimization

	PDE	ERCPDE CR+F	ERCPDE-F
Ackley	$1.62e-01 \pm 1.67e-02$	$1.52e-02 \pm 7.50e-03$	$\mathbf{6.47e-03 \pm 4.88e-03}$
Alpine	$8.88e+01 \pm 1.26e+01$	$7.05e+00 \pm 3.95e+00$	$\mathbf{1.98e+00 \pm 2.36e+00}$
DeJong	$1.92e+01 \pm 3.57e+00$	$3.37e-01 \pm 5.80e-01$	$\mathbf{8.82e-02 \pm 1.95e-01}$
Michalewicz	$-3.06e+02 \pm 5.68e+00$	$-3.11e+02 \pm 1.49e+01$	$\mathbf{-3.51e+02 \pm 3.58e+01}$
Rastrigin	$1.91e+03 \pm 9.94e+01$	$1.08e+03 \pm 1.42e+02$	$\mathbf{8.64e+02 \pm 1.38e+02}$
Schwefel	$-1.30e+05 \pm 3.17e+03$	$-1.33e+05 \pm 3.27e+03$	$\mathbf{-1.50e+05 \pm 1.14e+04}$
Rt. Ackley	$2.15e-01 \pm 2.50e-02$	$4.45e-02 \pm 9.44e-03$	$\mathbf{3.36e-02 \pm 1.03e-02}$
Rt. Michalewicz	$-1.76e+02 \pm 7.76e+00$	$-1.56e+02 \pm 7.13e+00$	$\mathbf{-1.83e+02 \pm 2.58e+01}$
Rt. Rastrigin	$1.95e+03 \pm 1.51e+02$	$1.16e+03 \pm 1.55e+02$	$\mathbf{9.61e+02 \pm 1.65e+02}$
Rt. Schwefel	$-1.65e+05 \pm 4.74e+03$	$-1.54e+05 \pm 6.07e+03$	$\mathbf{-1.66e+05 \pm 8.11e+03}$

In order to prove the effectiveness of varying control parameters, PDE has been used as a reference algorithm and run with a value of ϕ set to 0.2 (suggested value in [10]). For the sake of comparison, the jDE algorithm described in [1] has been run on the same function, but preliminary experiments show it to be clearly inferior to PDE (as is illustrated in Fig. 2 on page 49) and is therefore not included in the results presented below. ERCPDE has been run with the same value of ϕ; F_l and F_u were set to 0.1 and 0.9, respectively. Regarding the values of τ_1 and τ_2, two versions are presented here. The first version, indicated with ERCPDE-CR+F, has $\tau_1 = 0.1$ and $\tau_2 = 0.1$, the second indicated with ERCPDE-F has $\tau_1 = 0.1$ and $\tau_2 = 0$. In the last case, crossover rates are randomly sampled values and remain unchanged for the duration of the optimization process.

All the algorithms in this study have been run with populations of 200 individuals divided into 5 subpopulations of 40 individuals each, a setup which, according to our preliminary study, leads to the best average performance over the test functions. Each algorithm has undergone 50 independent runs for each test problem. Each single run has been performed for $500,000$ fitness evaluations. Table 2 shows the average of the final results detected by each algorithm $\pm$ the standard deviations,

Table 3. Results of the Wilcoxon Rank-Sum test (Comparison with ERCPDE-F)

Ackley	+	+
Alpine	+	+
DeJong	+	+
Michalewicz	+	+
Rastrigin	+	+
Schwefel	+	+
Rt. Ackley	+	+
Rt. Michalewicz	=	+
Rt. Rastrigin	+	+
Rt. Schwefel	=	+

with the 500 dimension case. The algorithm achieving the best result for each test problem is highlighted in bold face.

In order to prove the statistical significance of the results, the Wilcoxon Rank-sum test has been applied according to the description given in [12] for a confidence level of 0.95. Table 3 shows results of the test. A "+" indicates the case in which ERCPDE-F statistically outperforms, for the corresponding test problem, the algorithm mentioned in that column and a "=" indicates that a pairwise comparison leads to success of the Wilcoxon test, i.e., the two algorithms have the same performance. The results presented in Table 3 show that ERCPDE-F outperforms PDE in eight out of the ten test problems, and has comparable results in two cases. ERCPDE-F also outperforms ERCPDE-CR+F on all ten test problems.

Table 4. Results of the Holm procedure (Comparison with ERCPDE-F)

i	Optimizer	z	p	α/i	Hypothesis
2	PDE	-4.02e+00	2.85e-05	2.50e-02	Rejected
1	ERCPDE CR+F	-2.68e+00	3.65e-03	5.00e-02	Rejected

In order to strengthen the statistical significance of the results, the Holm procedure [5] has been applied by following the description in [4]. Considering the results in Table 2, the three algorithms under analysis have been ranked based on their average performance over the ten test problems, assigning to each algorithm a score R_i for $i = 0, \ldots, N_A - 1$ (where N_A is the number of algorithms under analysis, $N_A = 3$ in our case). With the calculated R_i values, the ERCPDE-F has been taken as

the reference algorithm. The values z_i have been calculated as

$$z_i = (R_0 - R_i)/\sqrt{N_A(N_A+1)/(6N_{TP})}$$

where R_0 is the rank of ERCPDE-F and N_{TP} is the number of test problems in consideration ($N_{TP} = 10$ in our case). The corresponding cumulative normal distribution values p_i corresponding to the z_i values have been calculated and compared with the corresponding α/i where α is the confidence threshold, set to 0.05 in our case. Table 4 displays z_i values, p_i values, and corresponding α/i. Moreover, it is indicated whether the null-hypothesis (when the two algorithms have indistinguishable performances) is "Rejected" i.e., the ERCPDE-F statistically outperforms the algorithm under consideration, or "Accepted" if the distribution of values can be considered the same (no algorithm is outperformed). The Holm procedure confirms that the ERCPDE-F displays a significantly better performance with respect to the other algorithms in this study.

Table 5. Results of the Q-test

	PDE	ERCPDE-CR+F	ERCPDE-F
Ackley	4.91e+03	2.24e+03	**1.96e+03**
Alpine	8.14e+04	2.18e+03	**2.04e+03**
DeJong	2.49e+03	**9.27e+02**	9.67e+02
Michalewicz	∞	7.00e+04	**5.69e+03**
Rastrigin	∞	3.64e+03	**2.71e+03**
Schwefel	∞	∞	**4.86e+03**
Rt. Ackley	8.39e+03	2.35e+03	**2.05e+03**
Rt. Michalewicz	7.44e+03	∞	**5.21e+03**
Rt. Rastrigin	∞	3.88e+03	**2.75e+03**
Rt. Schwefel	4.35e+03	1.23e+04	**3.48e+03**

In order to carry out a numerical comparison of the convergence speed performance for each test problem, the average fitness values J and G returned by the best performing algorithm respectively at the beginning and at the end of the optimization process have been computed. The threshold value $THR = J - 0.95(J - G)$ has then been calculated and represents 95% of the decay in the fitness value of the best performing algorithm. If during a certain run an algorithm succeeds in reaching the value THR, the run is said to be successful. For each test problem, the average amount of fitness evaluations $\bar{ne}$ required for each algorithm to reach THR has been computed. Subsequently, the Q-test (Q stands for Quality) described in [3] has been applied. For each test problem and each algorithm, the Q measure is computed as: $Q = \bar{ne}/R$ where

the robustness R is the percentage of successful runs. It is clear that, for each test problem, the smallest value equals the best performance in terms of convergence speed. The value "∞" means that $R = 0$, i.e., the algorithm never reached the *THR*. Table 5 shows the Q values for the ten problems; the best results are highlighted in bold face.

They show that the ERCPDE-F variant of the proposed ERCPDE algorithm has the best performance in terms of convergence speed in nine cases out of the ten test problems considered. Most importantly, the ERCPDE-F algorithm, throughout all considered test problems, is never characterized by an ∞ value of Q-measure. This fact demonstrates that the proposed algorithm is always competitive with the other algorithms in the benchmark and is never outperformed. In summary, the algorithmic behavior of ERCPDE-F is extremely promising in terms of algorithmic robustness.

Results show that endemic control parameters are an improvement over the original PDE algorithm. In addition, the fact that ERCPDE-F is superior to ERCPDE-CR+F indicates that repeated updates of the scale factor is beneficial to the performance of the algorithm, whereas the same kind of update applied to crossover rate is detrimental. In this sense, our study partly confirms and extends the self-adaptive control parameters strategy presented in [1] to PDE systems. The fact that in our case a crossover rate update is detrimental to the algorithmic performance is, according to our conjectures, explained by the fact that it appears to lead to an excessively frequent variation in the search strategy of each subpopulation, thus not allowing for an efficient exploitation of the available genotypes. Instead, as results show, it makes the algorithm too explorative, which seems to lead to stagnation (see Fig. 2).

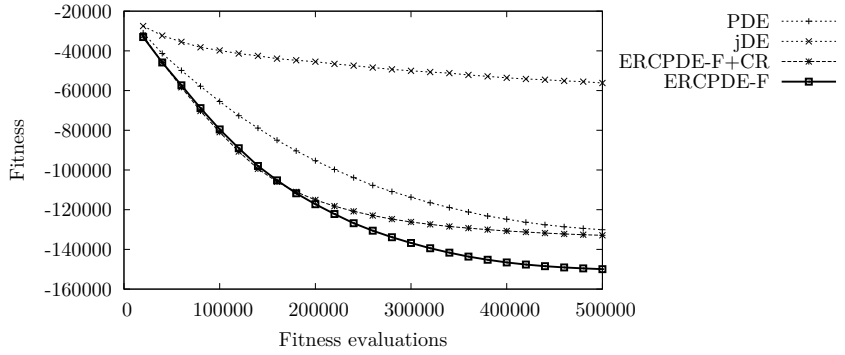


Figure 2. Performance trend (Schwefel)

4. Conclusion

This paper proposes a novel distributed algorithm based on a parallel differential evolution scheme previously proposed in literature, namely Endemic Randomized Control parameters Parallel Differential Evolution. Each population is characterized by its own control parameter values. The individuals displaying the best performance migrate across the subpopulations, thus conforming to various search strategies. The endemic (belonging to the subpopulation) control parameters are updated over time according to probabilistic criteria. Numerical results show that the proposed algorithmic strategy leads to significant improvements in terms of algorithmic performance with respect to original Parallel Differential Evolution. In addition, the scheme employing only the update of the scale factor seems more promising with respect to the scheme that updates both control parameters, which seems to indicate that while Parallel Differential Evolution structure requires a certain degree of randomization in order to highly enhance its performance, excessive randomization may lead to a too explorative algorithmic behavior and therefore stagnation.

References

[1] J. Brest, S. Greiner, B. Bošković, M. Mernik, and V. Žumer. Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems. *IEEE T. Evolut. Comput.*, 10(6):646–657, 2006.

[2] S. Das, A. Abraham, U. K. Chakraborty, and A. Konar. Differential evolution with a neighborhood-based mutation operator. *IEEE T. Evolut. Comput.*, 13(3):526–553, 2009.

[3] V. Feoktistov. *Differential Evolution: In Search of Solutions*, pages 83–86. Springer, 2006.

[4] S. Garca, A. Fernndez, J. Luengo, and F. Herrera. A study of statistical techniques and performance measures for genetics-based machine learning: accuracy and interpretability. *Soft Comput.*, 13(10):959–977, 2008.

[5] S. Holm. A simple sequentially rejective multiple test procedure. *Scand. J. Stat.*, 6(2):65–70, 1979.

[6] W. Kwedlo and K. Bandurski. A parallel differential evolution algorithm. In *Proc. IEEE International Symposium on Parallel Computing in Electrical Engineering*, pages 319–324, 2006.

[7] N. G. Pavlidis, D. K. Tasoulis, V. P. Plagianakos, G. Nikiforidis, and M. N. Vrahatis. Spiking neural network training using evolutionary algorithms. In *Proc. IEEE International Joint Conference on Neural Networks*, pages 2190–2194, 2005.

[8] K. V. Price, R. Storn, and J. Lampinen. *Differential Evolution: A Practical Approach to Global Optimization*. Springer, 2005.

[9] A. K. Qin, V. L. Huang, and P. N. Suganthan. Differential evolution algorithm with strategy adaptation for global numerical optimization. *IEEE T. Evolut. Comput.*, 13(2):398–417, 2009.

[10] D. K. Tasoulis, N. G. Pavlidis, V. P. Plagianakos, and M. N. Vrahatis. Parallel differential evolution. In *Proc. IEEE Congress on Evolutionary Computation (CEC)*, pages 2023–2029, 2004.

[11] V. Tirronen, F. Neri, T. Kärkkäinen, K. Majava, and T. Rossi. An enhanced memetic differential evolution in filter design for defect detection in paper production. *Evol. Comput.*, 16(4):529–555, 2008.

[12] F. Wilcoxon. Individual comparisons by ranking methods. *Biom. Bull.*, 1(6):80–83, 1945.

ANCESTRY TREE AS BASE FOR ANALYSIS OF EXPLORATION AND EXPLOITATION IN EVOLUTIONARY ALGORITHMS

Matej Črepinšek, Marjan Mernik
Faculty of Electrical Engineering and Computer Science
University of Maribor, Slovenia
{matej.crepinsek; marjan.mernik}@uni-mb.si

Barbara Zadobovšek
Nova vizija d.d.
Žalec, Slovenia
barbara.zadobovsek@vizija.si

Shih-Hsi Liu
Department of Computer Science
California State University, Fresno, USA
shliu@csufresno.edu

Abstract

This paper introduces an ancestry tree-based approach for exploration and exploitation analysis. The approach introduces a data structure to record the evolution history of a population and a number of exploration and exploitation metrics. Such an approach not only provides insight of how and when the two "e" influence an evolution process but also how the genetic structure of an individual is affected. The approach is applied to the multi-objective 0/1 knapsack problem.

Keywords: Exploration, Exploitation, Multi-objective 0/1 Knapsack Problem

1. Introduction

Any search algorithm is leveraged by two important aspects: exploration discovers potential offspring in new search regions; and exploitation utilizes promising solutions already identified. Exploration and exploitation (i.e., EE) of search space of an evolution process is one of utmost importance in evolutionary computation community, too [2]: High ratio of exploration tends to search entirely new regions of search space for potential offspring and to prevent local optimum convergence; and high ratio of exploitation tends to carry over identified potential individuals until the end of an evolution process. Yet, how and when to blend exploration with exploitation and balance them towards optimization and/or optimum convergence is still a challenging topic.

Existing approaches achieve the *how* objective by controlling crossover rate, mutation rate, selection pressure, and population size during an evolution process [7]. For example, mutation and crossover operators answer EE demands respectively by modifying individuals to increase the structural diversity of a population and by maintaining most of original or parents' genetic materials. Similarly, an evolution process may be adapted toward more exploitation/exploration by increasing/decreasing selection pressure or decreasing/increasing population size. As for the *when* objective, there have been plenty of metrics introduced as guidelines or insight to determine when to adjust operators toward more exploration or exploitation. The simplest metrics are fitness. A number of approaches utilize fitness to guide the EE explicitly or implicitly. For example, 1/5 success rule uses fitness to determine whether an individual is mutated successfully and then decide if mutation rate needs to be changed. Similarly, Parameter-Less GA [4] compares the average fitness of two populations with different sizes and then determines if either one requests population resizing. Other approaches within this category include [1, 3], among others. Diversity-derived metrics are another common one for guiding EE. The simplest diversity metrics are Hamming distance, Euclidean distance, and standard deviation. Various kinds of entropy (e.g., [6, 8]) are used to measure the diversity of a population. Other diversity-related approaches include [5, 9], to name a few. For other kinds of metrics, they are left to interested readers.

However, the aforementioned approaches/metrics are still difficult to delimit exploration from exploitation. To tackle this well-known challenge, this paper introduces an ancestry tree-based approach for analysis of EE. The approach introduces an ancestry tree data structure to represent the evolution of an individual, including its genetic materials, ancestors, descendants, types of modifications and number of

changes (i.e., distance) from previous generation. Additionally, a number of metrics are presented to offer insight of how (i.e., which operators and their ratios) and when (i.e., generation and progressiveness) EE influence an evolution process. Because of retaining the evolution record of an individual at a genotype level, EE applied to an individual's genetic structure can be also explicitly observed and measured. This paper utilizes the multi-objective 0/1 knapsack problem to testified the usage and explicitness of the approach.

The paper is organized as follows: Section 2 reviews the multi-objective 0/1 knapsack problem and the algorithm to solve the problem; Section 3 introduces the ancestry tree and metrics for EE; Section 4 presents results; and conclusions are presented in Section 5.

2. Multi-Objective 0/1 Knapsack Problem

The multi-objective 0/1 knapsack problem [10] is defined as follows: Given a set of m items and a set of n knapsacks, where c_i is capacity of knapsack i, $p_{i,j}$ is a profit and $w_{i,j}$ is weight of item j according to knapsack i, the task is to find a vector $x = (x_1, ..., x_m) \in \{0, 1\}^m$, such that $\forall i \in 1, ..., n : \sum_{j=1}^{m} w_{i,j} \cdot x_j \leq c_i$ and for which $f(x) = (f_1(x), ..., f_n(x))$ is maximum, where $f_i(x) = \sum_{j=1}^{m} p_{i,j} \cdot x_j$.

This paper applies SPEA2 [10] to solve the aforementioned problem. The algorithm starts with introducing an initial population (P_0 with $size = N$) and an empty archive (A_0 with $size = M$). It then iteratively performs four sequential steps until stopping criteria are met: (1) Fitness Assignment evaluates individuals of current population and those in the archive; (2) Environmental Selection chooses/copies non-dominated individuals to archive for next generation (A_{t+1}) from current population (P_t) and current archive (A_t); (3) Mating Selection performs binary tournament selection with replacement on A_{t+1} in order to fill the mating pool; and (4) Variation applies recombination and mutation operators on the mating pool and P_{t+1}. The output of the algorithm is a set of decision vectors represented by the non-dominated individuals in A_{t+1}. For more details, please refer to [10].

3. The Ancestry Tree

An ancestry tree is to record the evolution history of an individual within a population by the composition of a set of data collection structures (shown in Fig. 1). Each structure, representing an evolved individual at a specific generation, records individual's current generation, individual id, ancestor and its id, structure of the individual (chromosome data), how structure of the individual was obtained from ancestor

(by crossover (c), mutation (m), and/or repair (r), etc.), and number of changes between the individual and its ancestor (e.g., Hamming distance). If there is more than one parent (*crossover operator*), the ancestor that results in smaller number of changes is selected. The entire tree is then constructed accordingly based on the data collected during the process. An ancestry tree example is presented in Fig. 2. As shown in

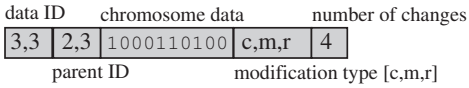


Figure 1. Data collection structure

the figure, ancestries of any individual and how an individual is obtained during the evolution can be identified. For example, individual (0, 8), individual 8 from 0^{th} generation, is randomly generated in initial population (*random*). Individual (3, 9), individual 9 from 3^{rd} generation, is evolved by ancestor (0, 8) using crossover operator and its Hamming distance to its ancestor is 2. Similarly, individual (4, 7), having an ancestor (3, 9), is generated by reproduction and hence Hamming distance is 0 (*clone*).

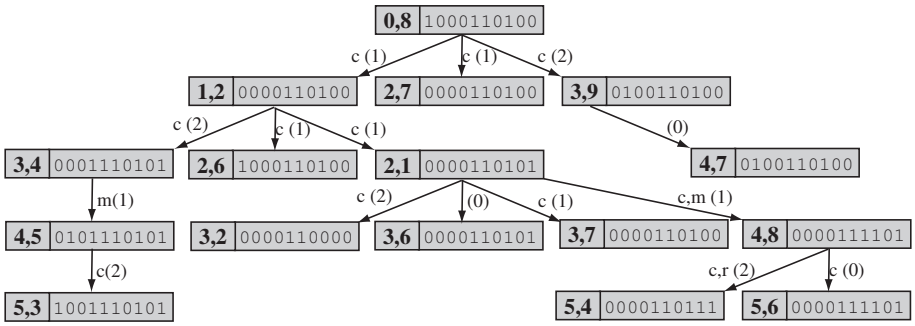


Figure 2. Example of ancestry tree

It is clear that the number of ancestry trees (τ) is equal to the initial population size (*pop_size*) and that the roots of these trees represent initial individuals. Let's define the number of tree nodes as tree size ($size(\tau)$), so the tree size of Fig. 2 is 16. As such, trees with individuals, having low fitness, can stop growing before the end of an evolution process. Conversely, individuals that were preferred in selection process may have higher probabilities evolve in trees with higher size.

From this perspective, one may claim that a root individual that has ancestry tree with higher size has more exploitation than smaller one(s), because it has more descendants evolved by modifying parent genome. But this could be true only if a small part of genome is changed. If any operator changes almost the entire parent genome, one may not claim that a new individual is produced by exploitation. To validate the above claim, an ancestry tree has to be split at a certain threshold number x, which can delimit exploration from exploitation. Various distance measures (e.g., Euclidian and Hamming) can be used for this purpose. For the multi-objective 0/1 knapsack problem, Hamming distance is applied since knapsack items are independent among each others. Every connection in τ with the number of changes higher than x results in splitting the ancestry tree ($\tau^{(x)}$). The splitting process is symbolically presented in Fig. 3.

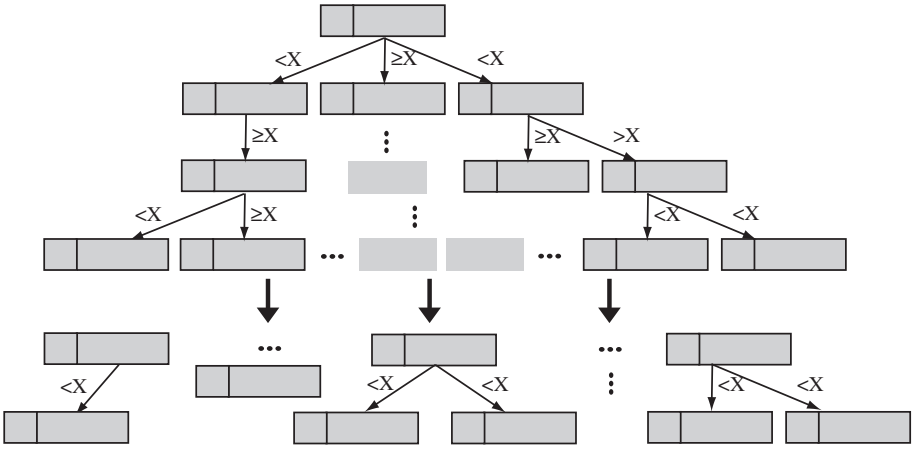


Figure 3. Splitting process of ancestry tree

Every split tree $\tau^{(x)}$ can be indexed as $\tau_{i,j}^{(x)}$, where $i \in [0 \dots I]$; $j \in [0 \dots J]$; $I = pop\_size - 1$; and $J = splits_i^{(x)} - 1$, describing number of trees after splitting process of $\tau_i^{(x)}$. In the case of tree from Fig. 2, number of $splits_8^{(2)} = 6$ (i.e., 5 + remaining one). Bigger $size(\tau_{i,j}^{(x)})$ means more exploitation of tree $\tau_{i,j}$ root and its descendants. The number of all trees with threshold x, $count(x)$ can be calculated as: $count(x) = \sum_{i=0}^{I} splits_i^{(x)}$.

3.1 Exploration Metrics

With the combination of the facts from previous subsection, ratio between EE can be defined as percentage of exploration, calculated as ratio between $count(x)$ and all individuals, where G is number of generations.

$$explorRatio(x) = \frac{count(x)}{pop\_size + pop\_size \cdot G} = \frac{count(x)}{count(0)}$$

Metric can also be calculated just for one tree, for our example of $\tau_8^{(2)}$ we get $explorRatio(2) = \frac{count(\tau_8^{(2)})}{count(\tau_8^{(0)})} = \frac{6}{16} = 0.2375$.

As splitting process with $x = 2$ is applied to Fig. 2, six new trees ($j \in [0 \dots 5]$) are obtained. Each new root individual represents a new search area, which is a starting point for exploration ($root(\tau_{i,j}^{(x)})$). With simple analysis of these individuals, more detailed information about exploration phase can be obtained. For example, exploration type ($explorType$) is metric that counts $root(\tau_{i,j}^{(x)})$ modification types ($count^{(type)}$), different operators that are applied during evolution. Root individuals (3,2), (3,4), (3,9), (5,3) and (5,4) were obtained by crossover, individual (5,4) was also affected by repair operator and individual (0,8) is from initial population and was generated randomly. After type counting, metrics is normalized by sum of all modification types ($countAllTypes$). Result can be interpreted as impact of operators on exploration.

$$explorType(x, type) = \frac{count^{(type)}(root(\tau_{i,j}^{(x)}))}{countAllTypes(root(\tau_{i,j}^{(x)}))} \text{ where } i \in [0 \dots I],\ j \in [0 \dots J].$$

Metrics can be also used for just one ancestry tree. For $\tau_8^{(2)}$ we get:

$$explorType(2, *) = \frac{count^{(type)}(root(\tau_{8,j}^{(2)}))}{countAllTypes(root(\tau_{8,j}^{(2)}))} = \begin{cases} 0 & mutation \\ 0.72 & crossover \\ 0.14 & repair \\ 0.14 & random \end{cases}$$

Metric $explorType$ answers the question of how exploration was started, while next metric tries to answer when it was started. To be able to analyze only exploration we transform an ancestry tree in such a way that we ignore all individuals made by exploitation. Individuals in such a tree are $root(\tau_{i,j}^{(x)})$ and their parents $parentRoot(\tau_{i,j}^{(x)})$. For our example, transformed tree is presented on Fig. 4.

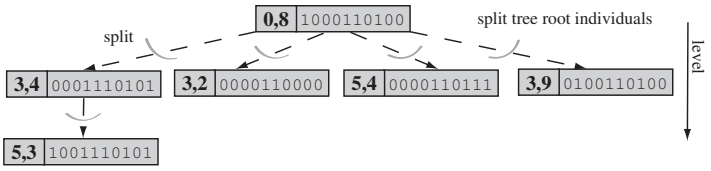


Figure 4. Transformed example of ancestry tree

Metric $explorDynamic_1$ is calculating average (avr) number of generations (gen) that was needed from parent ($parentRoot(\tau_{i,j}^{(x)})$) to the sibling ($root(\tau_{i,j}^{(x)})$). For example, from Fig. 4, function *parentRoot* for individuals (3,2), (3,4), (3,9) and (5,4)) returns individual (0,8) and for individual (5,3) it returns individual (3,4). Generational difference is 3 generations for individuals (3,2), (3,4), (3,9); 5 generations for individual (5,4) and 2 generations for individual (5,3).

$$explorDynamic_1(2) = avr(3, 3, 3, 5, 2) \approx 3.2 \pm 1.09$$

In general, the formula can be stated as:

$$explorDynamic_1(x) = avr(gen(root(\tau_{i,j}^{(x)}))) - gen(parentRoot(\tau_{i,j}^{(x)})),$$

where $i \in [0 \dots I]$, $j \in [1 \dots J]$.

Because metric $explorDynamic_1$ does not provide any information about how many splits happened before (in direct ancestry line), the following metric is defined. Metric $explorDynamic_2$ is analyzing tree split level ($treeLevel$). With this metric we can observe progressiveness of exploration. A low number indicates that most explorations were made from the same search point (same *parentRoot*)). This can also mean that a lot of unsuccessful explorations were tried from local optimum.

For $\tau_8^{(2)}$ we get for all root individuals except individual (5,3) tree level 1. Individual (5,3) has tree level 2 (path (0,8)→(3,4)→(5,3)).

$$explorDynamic_2(2) = (1, 1, 1, 1, 2) \approx 1.2 \pm 0.45$$

General formula:

$$explorDynamic_2(x) = avr(treeLevel(\tau_{i,j}^{(x)})), \text{ where } i \in [0 \dots I],\ j \in [1 \dots J].$$

3.2 Exploitation Metrics

To define exploitation ratio ($exploitRatio$), we can use metric $explorRatio$ as base: $exploitRatio(x) = 1 - explorRatio(x)$.

Unlike exploration analysis, where number of split trees and their root individuals were important, for exploitation analysis we are interested in split tree sizes, nodes, and their structures. Because we are also interested in all node individuals, we will index them with $k \in [0 \dots K]$. A tree root individual is always indexed with 0 and last index K is equal $size(\tau_{i,j}^{(x)}) - 1$.

Metric $exploitType$ is similar to $explorType$ except that we are interested in all nodes except root. It is important not to count root nodes

because they represent exploration.

$$exploitType(x, type) = \frac{count^{(type)}(\tau_{i,j}^{(x)}(k))}{countAllTypes(\tau_{i,j}^{(x)}(k))} \text{ where } \begin{array}{l} i \in [0 \dots I], \\ j \in [0 \dots J], \\ k \in [1 \dots K] \end{array}$$

In the case of example from Fig. 2 we get:

$$exploitType(2, *) = \frac{count^{(type)}(\tau_{8,j}^{(2)}(k))}{countAllTypes(\tau_{8,j}^{(2)}(k))} = \begin{cases} 0.18 & mutation \\ 0.64 & crossover \\ 0.00 & repair \\ 0.18 & clone \end{cases}$$

To measure the influence of selection on exploitation, *exploitStructure* metric is defined. If same individual was selected more than once, a tree becomes wider and as a result it has more leaves. For example, after splitting tree $\tau_8^{(2)}$: root individual (0,8) has 5 leafs ((2,6), (2,7), (3,6), (3,7) and (5,6)); root individual (3,4) has leaf (4,5); root individual (3,9) has leaf (4,7); and root individuals (5,3), (3,2), (3,9) and (5,4) have no leafs. This can be expressed as ratio between tree size and number of leafs (root is not counted as leaf). We get:

$exploitStructure(2) = \frac{\sum_{j=0}^{5} countLeafs(\tau_{8,j}^{(2)})}{size_8^{(2)}} = \frac{5+1+1+0+0+0+0}{16} = 0.43$

General formula:

$$exploitStructure(x) = \frac{\sum_{i=0}^{I} \sum_{j=0}^{J} countLeafs(\tau_{i,j}^{(x)})}{\sum_{i=0}^{I} size_i^{(x)}}$$

4. Results

In this section, due to page limitations, short analysis of EE on the multi-objective 0/1 knapsack problem with $n = 2$ and $m = 100$ is presented only. For this experiment SPEA2 algorithm with control parameters $p_m = 0.01$, $p_c = 0.8$, $tournament\_size = 2$, $pop\_size = 250$, $archive\_size = 250$, and $G = 290$ was used. Similar control parameters were used in [10], where SPEA2 found near optimum pareto front.

First, let us identify interesting threshold x (first three columns in Table 1). Threshold $x = 4$ was selected, because exploration/exploitation ratio is near 50%. Distribution of EE between different algorithm operators can be observed with *explorType* and *exploitType* metrics. For these metrics we may expose a significant impact of repair operator (25%) and low participation of cloning (8.8%). In average, new exploration was started after 73 generations ($explorDynamic_1$) and exploration was progressively continued 12 times ($explorDynamic_2$). The

ancestry tree has almost uniform distribution between ancestors and descendants (*exploitStructure*). How number of generations is influencing on EE (last six columns in Table 1) can be also observed. We can also see that exploration ratio is slightly falling and influence of repair operator is slightly increasing.

Influence of mutation rate p_m (0.5%, 1% and 2% in Table 2) is analyzed next. In this scenario we are not just analyzing all individuals (*all*), but also most optimistic scenario (*optimistic*), where only pareto optimum individuals and their ancestors were included. Because it is also interesting to know how many children that direct ancestors need to evolve in a right path, we also run metrics just on those individuals (*semioptimistic*). In Table 2 we can see that for $p_m = 1\%$ we need, in most optimistic case, only 236 individuals to obtain pareto front (*countAllNodes* for *optimistic* set of individuals). Low percentage of individuals in *semioptimistic* scenario (against all individuals) means that there were a lot of search in local optima. Ratio between *all* individuals 72750 and *semioptimistic* scenario 57112 ($p_m = 1\%$) is good. It indicates that the SPEA2 algorithm is well balanced in this experiment and it has no problems with local optima. But, to confirm this assumption we would need more statistically interpreted tests on different problems. Even more importantly, Table 2 also indicates that by increasing mutation rate balanced exploration/exploitation ratio is broken, which resulted in less optimal results (pareto front).

5. Conclusions

This paper introduces a novel ancestry tree-based approach to record the evolution history of a population. A number of metrics are derived which offer insight of how and when to perform exploration or exploitation, although some metrics due to space limitation were not presented and discussed. Additionally, the (splitting) tree structure and exploration/exploitation metrics also guided us how genetic structure was explored/exploited during an evolution process. The results of the multi-objective 0/1 knapsack problem show that such metrics explicitly present how and when exploration and exploitation dominated, which can also help us analyze the behavior of the evolution process in a better way. The presented approach can be used in various scenarios: evolution parameter control analysis, problem based success rate analysis, and the comparison of different evolution algorithms, among others. The analysis can be carried out incrementally, in selected parts or as a whole. As a result of such analysis we may develop a new algorithm, improve an algorithm or just get better understanding of an algorithm.

Table 1. Analyse of threshold x and influence of generations over time in $x = 4$

Metrics	$x = 3$	$x = 4$	$x = 5$	$G = 50$	$G = 100$	$G = 150$	$G = 200$	$G = 250$	$G = 290$
$explorRatio$	0.605	0.472	0.382	0.521	0.489	0.479	0.477	0.474	0.472
$explorType(c)$	0.414	0.420	0.419	0.439	0.429	0.425	0.422	0.421	0.420
$explorType(m)$	0.325	0.319	0.316	0.321	0.323	0.322	0.321	0.320	0.319
$explorType(r)$	0.258	0.257	0.261	0.222	0.239	0.247	0.252	0.255	0.257
$explorType(rnd)$	0.003	0.003	0.004	0.018	0.009	0.006	0.005	0.004	0.003
$explorDynamic_1$	59.445	73.652	83.890	5.942	16.937	30.864	46.126	61.515	73.652
$stdev$	$\pm$57.01	$\pm$66.84	$\pm$71.43	$\pm$5.78	$\pm$17.79	$\pm$30.56	$\pm$43.89	$\pm$56.78	$\pm$66.84
$explorDynamic_2$	13.477	12.090	11.286	8.130	10.116	11.012	11.561	11.897	12.090
$stdev$	$\pm$3.73	$\pm$3.23	$\pm$3.01	$\pm$3.89	$\pm$3.77	$\pm$3.55	$\pm$3.42	$\pm$3.31	$\pm$3.23
$exploitRatio$	0.395	0.528	0.618	0.479	0.511	0.521	0.523	0.526	0.528
$exploitType(c)$	0.415	0.407	0.411	0.393	0.405	0.406	0.408	0.408	0.407
$exploitType(m)$	0.344	0.345	0.343	0.367	0.355	0.350	0.348	0.346	0.345
$exploitType(r)$	0.110	0.160	0.174	0.137	0.147	0.153	0.157	0.159	0.160
$exploitType(cln)$	0.131	0.088	0.072	0.103	0.093	0.090	0.087	0.087	0.088
$exploitStructure$	0.381	0.509	0.593	0.417	0.470	0.490	0.498	0.504	0.509
$countAllNodes$	72750	72750	72750	12750	25250	37750	50250	62750	72750

Table 2. p_m analysis on all (a), optimistic (o) and semioptimistic (s) scenario

Metrics	$a(0.5\%)$	$o(0.5\%)$	$s(0.5\%)$	$a(1\%)$	$o(1\%)$	$s(1\%)$	$a(2\%)$	$o(2\%)$	$s(2\%)$
$explorRatio$	0.377	0.508	0.367	0.472	0.551	0.464	0.617	0.705	0.613
$explorType(c)$	0.477	0.491	0.477	0.420	0.446	0.418	0.372	0.363	0.372
$explorType(m)$	0.236	0.236	0.239	0.319	0.289	0.320	0.381	0.318	0.380
$explorType(r)$	0.283	0.267	0.284	0.257	0.261	0.262	0.245	0.310	0.248
$explorType(rnd)$	0.004	0.006	0.000	0.003	0.003	0.000	0.002	0.009	0.000
$explorDynamic_1$	73.756	35.019	73.675	73.652	43.783	79.063	72.605	38.209	69.581
$stdev$	±66.92	±52.46	±63.21	±66.84	±50.32	±65.72	±65.50	±57.68	±63.17
$explorDynamic_2$	11.683	9.565	12.261	12.090	11.682	12.764	12.946	10.491	12.951
$stdev$	±4.09	±4.27	±3.57	±3.23	±3.17	±2.28	±2.86	±3.74	±2.47
$exploitRatio$	0.623	0.492	0.633	0.528	0.449	0.536	0.383	0.295	0.387
$exploitType(c)$	0.479	0.574	0.481	0.407	0.494	0.409	0.357	0.353	0.357
$exploitType(m)$	0.238	0.277	0.239	0.345	0.333	0.344	0.447	0.451	0.447
$exploitType(r)$	0.144	0.149	0.144	0.160	0.172	0.161	0.157	0.195	0.159
$exploitType(cln)$	0.139	0.000	0.136	0.088	0.000	0.086	0.039	0.000	0.038
$exploitStructure$	0.602	0.069	0.612	0.509	0.076	0.521	0.369	0.063	0.373
$countAllNodes$	72750	321	59380	72750	236	57112	72750	237	51015

References

[1] T. Bäck, D. B. Fogel, and Z. Michalewicz. *Handbook of Evolutionary Computations*. University of Oxford Press, New York, 1996.

[2] A. Eiben and C. Schippers. On evolutionary exploration and exploitation. *Fund. Inform.*, 35(1-4):35–50, 1998.

[3] M. Gallagher and B. Yuan. A general-purpose tunable landscape generator. *IEEE T. Evolut. Comput.*, 10(5):590–602, 2006.

[4] G. Harik and F. Lobo. *A Parameter-less Genetic Algorithm.* Technical Report IlliGAL 9900, University of Illinois at Urbana-Champaign, 1999.

[5] T.-Y. Huang and Y.-Y, Chen. Diversity-based selection pooling scheme in evolution strategies. In *Proc. 16th ACM Symposium on Applied Computing*, pages 351–355, 2001.

[6] S.-H. Liu, M. Mernik, and B. R. Bryant. Entropy-driven exploration and exploitation in evolutionary algorithms. In *Proc. Second International Conference on Bioinspired Optimization Methods and their Applications (BIOMA)*, pages 15–24, 2006.

[7] F. G. Lobo, C. F. Lima, and Z. Michalewicz. *Parameter Setting in Evolutionary Algorithms*. Springer, Studies in Computational Intelligence (54), 2007.

[8] J. Rosca. Entropy-driven adaptive representation. In *Proc. Workshop on Genetic Programming: From Theory to Real-World Applications*, pages 23–32, 1995.

[9] R. K. Ursem. Diversity-guided evolutionary algorithms. *Lect. Notes Comput. Sc.*, 2439:462–471, 2002.

[10] E. Zitzler, M. Laumanns, and L. Thiele. SPEA2: Improving the strength Pareto evolutionary algorithm for multiobjective optimization. In *Evolutionary Methods for Design, Optimisation and Control with Application to Industrial Problems*, pages 95–100, 2002.

COMPONENT DECOMPOSITION IN PARALLEL DIFFERENTIAL EVOLUTION

Matthieu Weber, Ferrante Neri*, Ville Tirronen
Department of Mathematical Information Technology
University of Jyväskylä, Finland
{matthieu.weber; ferrante.neri; ville.tirronen}@jyu.fi

Abstract This paper proposes decomposing the search space of large-scale problems into lower-dimensionality subspaces, and associating each of these to one subpopulation of a Parallel Differential Evolution algorithm. Each subpopulation is running a modified Differential Evolution algorithm, where the crossover function is limited to components of the subpopulation's associated subspace. According to numerical results the Parallel Component Decomposition Differential Evolution seems to be a clear improvement over the original Parallel Distributed Evolution, making it a simple, robust, and efficient algorithm suited for various applications.

Keywords: Differential Evolution, Large-scale optimization, Parallel Evolutionary Algorithm, Search space decomposition

1. Introduction

Differential Evolution (DE), see [7], has shown high performance in various types of optimization problems, and more particularly in continuous problems, for example [10]. Like all optimization algorithms, DE suffers from the so-called "curse of dimensionality", which refers to the fact that the complexity of a multidimensional problem increases with its dimensionality in an exponential fashion. In [12] we have shown that the structured population characterizing the Parallel Differential Evolution (PDE) described in [9] is one enhancement that allows to break this curse: instead of running the DE over a single population, PDE

*This work is supported by Academy of Finland, Akatemiatutkija 130600, Algorithmic Design Issues in Memetic Computing.

runs multiple DE algorithms over multiple subpopulations which are organized in a unidirectional ring; with a given probability, the best individual of a subpopulation is migrated to the next subpopulation in the ring, where it replaces a random individual. In order to enhance PDE, [12] then proposes to use two families of subpopulations, the first one running a regular PDE while the second runs independent instances of the population-size reduction DE proposed in [2]; after an observation period, individuals from the second family are injected into the subpopulations of the first family.

The randomization of DE's control parameter during the course of the optimization process, such as in [1], is another strategy which allows to improve the performance of the algorithm. In [13], we propose to blend the randomization of the control parameters with the migration mechanism of the PDE, resulting in a variant of the latter where the scale factor used by the DE at one given subpopulation is migrated (with a small perturbation) along with the best individual.

The rationale behind these variants of PDE and DE lies in the fact that DE contains a limited amount of search moves and the population could fail at enhancing upon the available genotypes thus resulting in stagnation. The variation over time of the control parameters or the injection of independently optimized individuals increases the number of available search moves, thus allowing for a more exhaustive exploration of the problem space.

Yet another possibility of enhancing the DE is to subdivide the search space into subspaces and optimize only the components of the solution that belong to one subspace, while keeping the others constant. This approach, which effectively breaks the curse of dimensionality by locally reducing the dimensionality of the problem, was first proposed in [6]. Here each component of the solution was optimized separately by a dedicated subpopulation, while the fitness of one solution was evaluated over the whole set of components by taking random individuals from the other subpopulations in order to reconstruct a whole solution; this process was called cooperative co-evolution. This decomposition scheme is, however, reputed to be inefficient on non-separable functions (see [6, 15]) i.e., where the variables of the problem interact with each other. To overcome this problem, [17] uses randomly selected sub-components which change over time, [15] proposes using a set of weights, itself evolved using DE, to select the sub-components, [16] makes use of the DE with neighborhood search, while [11] selects components showing the highest variance across the population as the ones to be optimized at a given time. The three above cited articles made use of non-standard DE algorithms in order to yet increase their efficiency.

The algorithm proposed in this paper, the Parallel Component Decomposition Differential Evolution (PaCoDDE) borrows the structured population and best individual migration from the Parallel Differential Evolution (PDE) described in [9] and merges it with a static subcomponent decomposition: the search space is decomposed into subspaces of near-equal dimensionality, and each subspace is assigned to one subpopulation. The subpopulations focus on optimizing their solutions within their own subspaces, while keeping the rest of the components constant. On every generation, each subpopulation has a given probability of migrating its best-performing individual to the next subpopulation in the ring, allowing to the optimized sub-components to incrementally propagate the other subpopulations.

The remainder of this article is organized in the following way. Section 2 describes the working principles of DE, PDE and PaCoDDE. Section 3 shows the experimental setup and numerical results of the present study. Section 4 gives the conclusions of this paper.

2. Parallel Component Decomposition Differential Evolution

In order to clarify the notation used throughout this paper we refer to the minimization problem of an objective function $f(x)$, where x is a vector of n design variables in a decision space D.

At the beginning of the optimization process, S_{pop} individuals are pseudo-randomly sampled with a uniform distribution function within the decision space D (for simplicity, the term random will be used instead of pseudo-random in the reminder of this paper). The S_{pop} individuals constituting the populations are distributed over m subpopulations $P_k, k = 1, \ldots, m$ arranged in a unidirectional ring. Each subpopulation is therefore composed of S_{pop}/m individuals. Additionally, the set of dimensions of the search space D is decomposed into m subsets C_k of approximately equal sizes n_k, with the constraints that $\sum_{k=1}^{m} n_k = n$ and $\forall (k_1, k_2) \in \{1, \ldots, m\}^2, k_1 \neq k_2, C_{k_1} \cap C_{k_2} = \emptyset$. In other words, C_k represents the sub-component of x that is to be optimized by subpopulation P_k.

Within each subpopulation P_k, a modified DE is performed. At each generation, for each individual x_i of P_k, three individuals x_r, x_s and x_t are randomly extracted from the subpopulation. According to the DE logic, a provisional offspring x'_{off} is generated by mutation as:

$$x'_{off} = x_t + F(x_r - x_s) \tag{1}$$

where $F \in [0, 1+]$ is a scale factor which controls the length of the exploration vector $(x_r - x_s)$ and thus determines how far from point x_i the offspring should be generated. With $F \in [0, 1+]$, it is meant here that the scale factor should be a positive value which cannot be much greater than 1, see [7]. The mutation scheme shown in Eq. (1) is also known as DE/rand/1. It is worthwhile mentioning that there exist many other mutation variants, see [8].

When the provisional offspring has been generated by mutation, each gene $x_{i,j}, j \in C_k$ of x_i is exchanged with the corresponding gene of x'_{off} with a uniform probability and the final offspring x_{off} is generated:

$$x_{off,j} = \begin{cases} x'_{off,j} & \text{if} \quad j \in C_k \text{ and } rand\,(0,1) \leq CR \\ x_{i,j} & \text{otherwise} \end{cases} \tag{2}$$

where $rand\,(0,1)$ is a random number between 0 and 1; $j = 1, \ldots, n$ is the index of the gene under examination. This modified crossover function always keeps the parent's genes which are not part of the considered sub-component, and actually crosses over only the genes of the offspring and of the parent which are part of the sub-component.

The resulting offspring x_{off} is evaluated and, according to a one-to-one spawning strategy, replaces x_i if and only if $f(x_{off}) \leq f(x_i)$; otherwise no replacement occurs. It must be remarked that although the replacement indexes are saved one by one during generation, actual replacements occur all at once at the end of the generation.

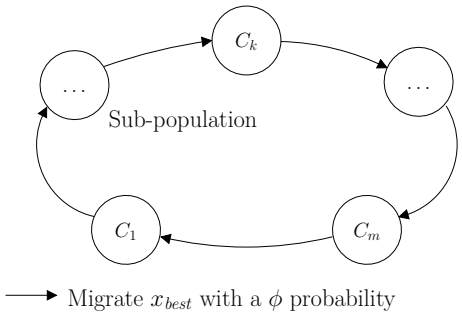


Figure 1. Working principle of the PaCoDDE

In PDE, each subpopulations P_k runs a regular DE algorithm, which is replaced in PaCoDDE by the modified DE algorithm described above. On each generation, the subpopulation has a given probability ϕ to send a copy of its best individual to its next neighbor subpopulation in the ring. When a migration occurs, the migrating individual replaces a randomly selected individual belonging to the target subpopulation, with

an exception being made of the subpopulation's best performing individual, which can never be replaced. For the sake of clarity, a schema highlighting the working principles of PaCoDDE is shown in Fig. 1.

Although PaCoDDE may seem crude due to the fact that it makes use of a static decomposition of the search space while the modern above-mentioned algorithms are based on random or dynamic decomposition schemes, its specificity lies in that the individuals showing the best performance are traveling between subpopulations during the run of the algorithm, being optimized incrementally over each of their subcomponents, and revisiting each subpopulation multiple times. The migration rate must therefore be high enough to ensure that these individuals are not allowed to become excessively specialized within one subspace while being far from optimal when considering the whole search space.

3. Experimental Results

The test problems listed in Table 1 have been considered in this study. The rotated version of some of them have been included into the benchmark set. These rotated problems have been generated through the multiplication of the vector of variables by a randomly generated sparse orthogonal rotation matrix, created by composing n rotations of random angles (uniformly sampled in $[-\pi, \pi]$), one around each of the n axes of the search space. While the six unrotated functions are separable, the rotated versions are not. In total, ten test problems have been considered in this study with $n = 500$.

Table 1. Test Problems

Test Problem	**Function**	**Decision Space**	**Optimum**
Ackley	$-20 + e + 20\exp\left(-\frac{0.2}{n}\sqrt{\sum_{i=1}^{n} x_i^2}\right)$ $-\exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi \cdot x_i)x_i\right)$	$[-1, 1]^n$	0
Alpine	$\sum_{i=1}^{n} \lvert x_i \sin x_i + 0.1x_i \rvert$	$[-10, 10]^n$	0
Sphere	$\lVert x \rVert^2$	$[-5.12, 5.12]^n$	0
Michalewicz	$-\sum_{i=1}^{n} \sin x_i \left(\sin\left(\frac{i \cdot x_i^2}{\pi}\right)\right)^{20}$	$[0, \pi]^n$	unknown
Rastrigin	$10n + \sum_{i=0}^{n}\left(x_i^2 - 10\cos(2\pi x_i)\right)$	$[-5.12, 5.12]^n$	0
Schwefel	$-\sum_{i=1}^{n} x_i \sin\left(\sqrt{\lvert x_i \rvert}\right)$	$[-500, 500]^n$	$-418.9829n$

Table 2. Average Fitness ± standard deviation at the end of the optimization process

	PDE	DEwSAcc	PaCoDDE
Ackley	$1.62e-01 \pm 1.67e-02$	$\mathbf{4.22e-02 \pm 9.85e-03}$	$5.44e-02 \pm 5.80e-03$
Alpine	$8.88e+01 \pm 1.26e+01$	$7.07e+01 \pm 1.74e+01$	$\mathbf{1.97e+01 \pm 3.05e+00}$
Sphere	$1.92e+01 \pm 3.57e+00$	$\mathbf{1.51e+00 \pm 5.68e-01}$	$2.42e+00 \pm 4.55e-01$
Michalewicz	$-3.06e+02 \pm 5.68e+00$	$-2.52e+02 \pm 1.04e+01$	$\mathbf{-4.18e+02 \pm 3.18e+00}$
Rastrigin	$1.91e+03 \pm 9.94e+01$	$1.57e+03 \pm 2.24e+02$	$\mathbf{7.84e+02 \pm 4.48e+01}$
Schwefel	$-1.30e+05 \pm 3.17e+03$	$-1.27e+05 \pm 6.91e+03$	$\mathbf{-1.59e+05 \pm 2.12e+03}$
Rt. Ackley	$2.15e-01 \pm 2.50e-02$	$\mathbf{8.36e-02 \pm 1.26e-02}$	$1.41e-01 \pm 2.10e-02$
Rt. Michalewicz	$-1.76e+02 \pm 7.76e+00$	$-1.48e+02 \pm 4.90e+00$	$\mathbf{-2.55e+02 \pm 4.93e+00}$
Rt. Rastrigin	$1.95e+03 \pm 1.51e+02$	$2.50e+03 \pm 2.32e+02$	$\mathbf{1.19e+03 \pm 5.06e+01}$
Rt. Schwefel	$-1.65e+05 \pm 4.74e+03$	$-1.25e+05 \pm 4.54e+03$	$\mathbf{-1.90e+05 \pm 2.81e+03}$

To prove the effectiveness of component decomposition within the framework of a PDE algorithm, PaCoDDE has been compared to the original PDE and to the DEwSAcc described in [17]. In both PDE and PaCoDDE the control parameters of DE, namely the scale factor F and the crossover rate CR have been set to 0.7 and 0.3 respectively, in accordance with the suggestions given in [19] and [18]. The migration rate ϕ was set to 0.2, as suggested in [9]. The τ parameter of DEwSAcc was set to $0.2\sqrt{2}/\sqrt{500}$, as recommended in [17] for 500-dimensional problems. It has to be mentioned that DEwSAcc, in addition to search space decomposition, uses multiple mutation schemes and self-adaptive control parameters, a variation on the original DE which in itself improves performance of the algorithm compared to the original DE. PaCoDDE, on the contrary, uses a single mutation scheme and static control parameter values, as does PDE.

All the algorithms in this study have been run with populations of 200 individuals. In PDE and PaCoDDE, these individuals are organized into 5 subpopulations of 40 individuals, a setup which, according to our preliminary study, leads to the best average performance over the test functions. For DEwSAcc, a single population of 200 individuals is used. The search space was decomposed as follows. The dimensions of the problems, indexed from 1 to 500, were split into 5 intervals $C_k = \{100k - 99, \ldots, 100k\}$ for $k = 1, \ldots, 5$. Each interval C_k was then assigned to subpopulation P_k. Each algorithm has undergone 50 independent runs for each test problem. Each single run has been performed for 500,000 fitness evaluations. Table 2 shows the average of the final results detected by each algorithm ± the standard deviations. The algorithm achieving the best result for each test problem is highlighted in bold face. An example of the performance trend is shown in Fig. 2.

To prove the statistical significance of the results, the Wilcoxon Rank-sum test has been applied according to the description given in [14] for a confidence level of 0.95. Table 3 shows results of the test. A

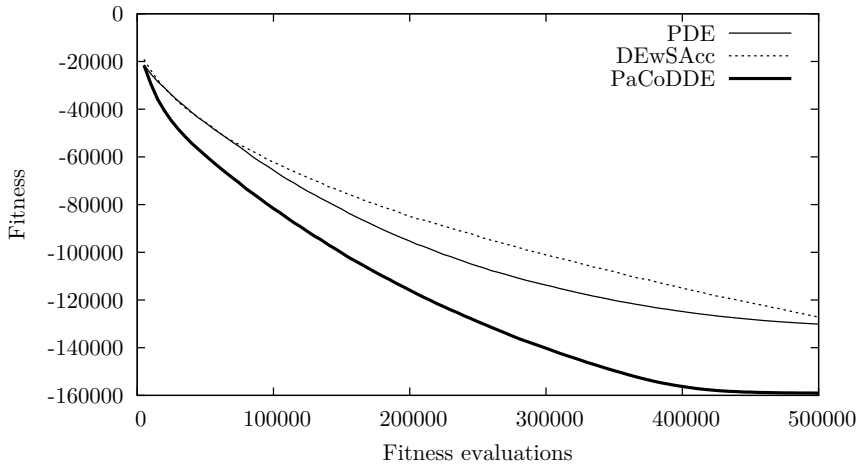


Figure 2. Example of a performance trend (Schwefel)

"+" indicates a case where PaCoDDE statistically outperforms, for the corresponding test problem, the algorithm mentioned in that column, a "−" indicates that PaCoDDE is outperformed by the algorithm it is compared to, and a "=" indicates that a pairwise comparison leads to success of the Wilcoxon test i.e., that the two algorithms have the same performance. The results presented in Table 3 show that PaCoDDE outperforms PDE for all ten test problems. It moreover outperforms DEwSAcc in seven cases out of the ten.

Table 3. Wilcoxon Rank-Sum test (Comparison with PaCoDDE)

	PDE	DEwSAcc
Ackley	+	-
Alpine	+	+
Sphere	+	-
Michalewicz	+	+
Rastrigin	+	+
Schwefel	+	+
Rt. Ackley	+	-
Rt. Michalewicz	+	+
Rt. Rastrigin	+	+
Rt. Schwefel	+	+

To strengthen the statistical significance of the results, the Holm procedure [5] has been applied by following the description in [4]. Consider-

ing the results in Table 2, the three algorithms under analysis have been ranked on the basis of their average performance calculated over the ten test problems, assigning to each algorithm a score R_i for $i = 0, \ldots, N_A - 1$ (where N_A is the number of algorithms under analysis, $N_A = 3$ in our case). With the calculated R_i values, the PaCoDDE has been taken as the reference algorithm. The values z_i have then been calculated as

$$z_i = (R_0 - R_i)/\sqrt{N_A(N_A + 1)/(6N_{TP})}$$

where R_0 is the rank of PaCoDDE and N_{TP} is the number of test problems in consideration ($N_{TP} = 10$ in our case). The cumulative normal distribution values p_i corresponding to the z_i values have then been calculated and compared with the corresponding α/i values where α is the confidence threshold, set to 0.05 in our case. Table 4 displays z_i values, p_i values, and corresponding α/i. Moreover, it is indicated whether the null-hypothesis (when the two algorithms have indistinguishable performances) is "Rejected" i.e., the PaCoDDE statistically outperforms the algorithm under consideration, or "Accepted" if the distribution of values can be considered the same (no algorithm is outperformed). The Holm procedure thus confirms that the PaCoDDE performs significantly better than the other algorithms in this study.

Table 4. Results of the Holm procedure (Comparison with PaCoDDE)

i	Optimizer	z	p	α/i	Hypothesis
2	PDE	-2.68e+00	3.65e-03	2.50e-02	Rejected
1	DEwSAcc	-2.01e+00	2.21e-02	5.00e-02	Rejected

To carry out a numerical comparison of the convergence speed performance for each test problem, the average fitness values J and G returned by the best performing algorithm respectively at the beginning and at the end of the optimization process have been computed. The threshold value $THR = J - 0.95(J - G)$ has then been calculated and represents 95% of the decay in the fitness value of the best performing algorithm. If during a certain run an algorithm succeeds in reaching the value THR, the run is said to be successful. For each test problem, the average amount of fitness evaluations $\bar{ne}$ required for each algorithm to reach THR has been computed. Subsequently, the Q-test (Q stands for Quality) described in [3] has been applied. For each test problem and each algorithm, the Q measure is computed as: $Q = \bar{ne}/R$ where the robustness R is the percentage of successful runs. It is clear that, for each test problem, the smallest value equals the best performance in terms

of convergence speed. The value "∞" means that $R = 0$, i.e., the algorithm never reached the *THR*. Table 5 shows the Q values for the ten problems and the best results are highlighted in bold face. These show

Table 5. Results of the Q-test

	PDE	DEwSAcc	PaCoDDE
Ackley	4.41e+03	**2.96e+03**	3.42e+03
Alpine	9.98e+03	5.35e+03	**3.61e+03**
Sphere	2.38e+03	**1.42e+03**	2.06e+03
Michalewicz	∞	∞	**4.41e+03**
Rastrigin	∞	2.48e+05	**3.92e+03**
Schwefel	∞	∞	**3.62e+03**
Rt. Ackley	4.88e+03	**3.03e+03**	4.01e+03
Rt. Michalewicz	∞	∞	**4.18e+03**
Rt. Rastrigin	∞	2.47e+05	**3.94e+03**
Rt. Schwefel	∞	∞	**3.33e+03**

that the PaCoDDE has the best performance in terms of convergence speed in seven cases out of the ten test problems considered. Most importantly, the PaCoDDE algorithm, throughout all considered test problems, is never characterized by an ∞ value of the Q-measure, which demonstrates that the proposed algorithm is always competitive with the other algorithms in the benchmark and is never outperformed. In summary, the algorithmic behavior of PaCoDDE is extremely promising in terms of algorithmic robustness.

Results show that sub-component decomposition of the search space applied to PDE is an improvement over the original algorithm. This confirms the results presented in [6] about search space decomposition and sub-component optimization, and extends them to structured-population DE schemes.

4. Conclusion

This paper proposes an improved variant of the Parallel Differential Evolution for high-dimensionality problems, namely Parallel Component Decomposition Differential Evolution. The search space of the problem is decomposed into disjoint subspaces, each of which is associated to one subpopulation of a Parallel Differential Evolution. Each subpopulation runs a modified Differential Evolution algorithm, where the crossover function limits its action on the considered individual to the components belonging to the associated subspace. Individuals displaying the best performance are then given the possibility to migrate to

the neighboring subpopulation, where another of their sub-component is optimized. Numerical results show that the proposed algorithmic logic leads to significant improvements in terms of algorithmic performance with respect to standard Parallel Differential Evolution. Despite the fact that our algorithm employs a single mutation scheme, static control parameter values and a static sub-component decomposition, on average it outperforms the DEwSAcc algorithm which employs multiple mutation schemes, self-adaptive control parameters and a random, dynamic decomposition scheme. The integration of such features into a Parallel Differential Evolution framework seems to be a promising research path.

References

[1] J. Brest, S. Greiner, B. Bošković, M. Mernik, and V. Žumer. Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems. *IEEE T. Evolut. Comput.*, 10(6):646–657, 2006.

[2] J. Brest and M. Sepesy Maučec. Population size reduction for the differential evolution algorithm. *Appl. Intell.*, 29(3):228–247, 2008.

[3] V. Feoktistov. *Differential Evolution: In Search of Solutions*, pages 83–86. Springer, 2006.

[4] S. Garca, A. Fernndez, J. Luengo, and F. Herrera. A study of statistical techniques and performance measures for genetics-based machine learning: accuracy and interpretability. *Soft Comput.*, 13(10):959–977, 2008.

[5] S. Holm. A simple sequentially rejective multiple test procedure. *Scand. J. Stat.*, 6(2):65–70, 1979.

[6] M. A. Potter and K. A. De Jong. A cooperative coevolutionary approach to function optimization. In *Proc. Third Conference on Parallel Problem Solving from Nature*, pages 249–257, 1994.

[7] K. V. Price, R. Storn, and J. Lampinen. *Differential Evolution: A Practical Approach to Global Optimization*. Springer, 2005.

[8] A. K. Qin, V. L. Huang, and P. N. Suganthan. Differential evolution algorithm with strategy adaptation for global numerical optimization. *IEEE T. Evolut. Comput.*, 13:398–417, 2009.

[9] D. K. Tasoulis, N. G. Pavlidis, V. P. Plagianakos, and M. N. Vrahatis. Parallel differential evolution. In *Proc. IEEE Congress on Evolutionary Computation (CEC)*, pages 2023–2029, 2004.

[10] V. Tirronen, F. Neri, T. Kärkkäinen, K. Majava, and T. Rossi. An enhanced memetic differential evolution in filter design for defect detection in paper production. *Evol. Comput.*, 16(4):529–555, 2008.

[11] Y. Wang, B. Li, and X. Lai. Variance priority based cooperative co-evolution differential evolution for large scale global optimization. In *Proc. IEEE Congress on Evolutionary Computation (CEC)*, pages 1232–1239, 2009.

[12] M. Weber, F. Neri, and V. Tirronen. Distributed differential evolution with explorative-exploitative population families. *Genet. Program. Evol. M.*, 10(4):343–371, 2009.

[13] M. Weber, F. Neri, and V. Tirronen. Scale factor inheritance mechanism in distributed differential evolution. *Soft Comput.*, 2009.

[14] F. Wilcoxon. Individual comparisons by ranking methods. *Biom. Bull.*, 1(6):80–83, 1945.

[15] Z. Yang, K. Tang, and X. Yao. Differential evolution for high-dimensional function optimization. In *Proc. IEEE Congress on Evolutionary Computation (CEC)*, pages 3523–3530, 2007.

[16] Z. Yang, K. Tang, and X. Yao. Large scale evolutionary optimization using cooperative coevolution. *Inform. Sciences*, 178(15):2985–2999, 2008.

[17] A. Zamuda, J. Brest, B. Bošković, and V. Žumer. Large scale global optimization using differential evolution with self-adaptation and cooperative co-evolution. In *Proc. IEEE World Congress on Computational Intelligence (WCCI)*, pages 3719–3726, 2008.

[18] K. Zielinski and R. Laur. Stopping criteria for differential evolution in constrained single-objective optimization. In U. K. Chakraborty (Eds). *Advances in Differential Evolution. Studies in Computational Intelligence*, pages 143:111–138. Springer, 2008.

[19] K. Zielinski, P. Weitkemper, R. Laur, and K.-D. Kammeyer. Parameter study for differential evolution using a power allocation problem including interference cancellation. In *Proc. IEEE Congress on Evolutionary Computation (CEC)*, pages 1857–1864, 2006.

THE MULTI-OBJECTIVE DISTINCT CANDIDATES OPTIMIZATION APPROACH

Rasmus K. Ursem
Structural and Fluid Mechanics, Grundfos Management A/S
Bjerringbro, Denmark
ursem@cs.au.dk

Peter Dueholm Justesen
Department of Computer Science
Aarhus University, Denmark
juste@cs.au.dk

Abstract Traditional multi-objective optimization algorithms typically return several hundred non-dominated candidate solutions. From practical point-of-view, a small set of 5-10 distinct candidates is often preferred because post-processing of many solutions may be too costly, too time-consuming, too difficult to compare design differences, or similar solutions turn out to be statistically equal in prototyping and manufacturing. Interestingly, these limitations apply to most real-world problems. In this paper, we introduce Multi-objective Distinct Candidates Optimization (MODCO) as an approach to find a user-defined number of clearly different solutions wrt. performance and design. In MODCO, we distinguish between generalized and domain-specific preferences, where generalized preferences address the aforementioned limitations and the domain-specific preferences cover the DM's whishes if available.

Keywords: Distinct candidates, Diversity management, Multi-objective optimization

1. Introduction

Successful application of multi-objective optimization to a real-world problem typically consists of two steps. First, the optimization step where the problem is set up, the chosen algorithm is executed, and all non-dominated solutions are gathered. Second, the decision mak-

ing step where the single solution to implement is chosen among the non-dominated solutions found in step 1, see Fig. 1 and Deb [9, p. 5].

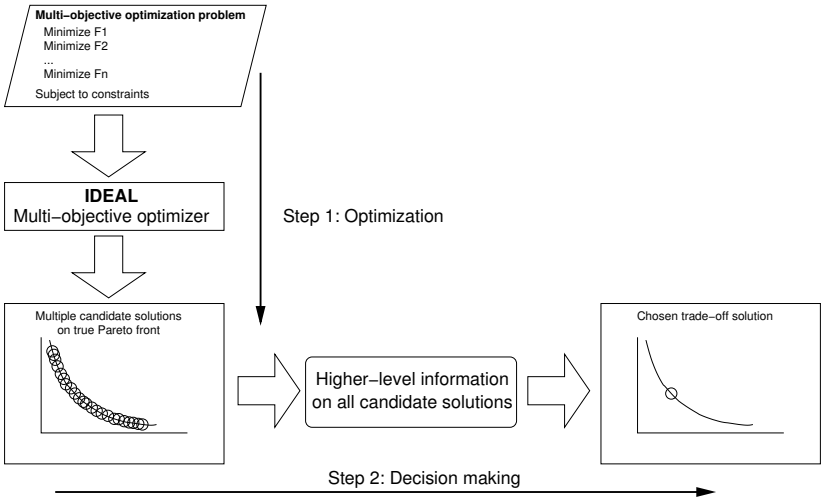


Figure 1. Process for application of multi-objective optimization.

In this process, the decision maker (DM) has to apply his preferences among the objectives to select the final solution. Veldhuizen and Lamont categorize the point in the process where the DM apply his preferences into three categories; 1) *a priori* – before the optimization is initiated, 2) *progressive* – during the optimization, and 3) *a posteriori* – after the optimization is finished [18]. Algorithms in category 1 typically transform the multi-objective problem into a single objective by specifying an utility function combining the multiple objectives. The weighted sum approach is the most widely known algorithm in this category. The progressive algorithms in category 2 usually incorporate the DM's preferences in form of decision support systems, see [8] for a survey. Finally, category 3 algorithms exclude the DM's preferences from the search. Instead, they typically produce a large set of Pareto-optimal solutions for the DM to choose from in step 2.

The drawback of approach 1 and 2 is that the DM has to make a choice regarding the importance of the involved objectives *prior* to the actual optimization, which may be difficult before the DM has seen any solutions. In addition, such choices are highly domain-specific and problem dependent, and algorithms are thus hard to generalize for a broader range of applications. In contrast, the traditional MO algorithms in category 3 are generally applicable. However, these algorithms produce hundreds or thousands of solutions and the notion is to leave it to the

DM to gather the "higher-level information" in step 2 on this set and choose the actual solution to implement. The often large set returned by *a posteriori* algorithms pose a serious problem because it may be *impossible* to gather "higher-level information" on such a large set. In short, time, money, and other reasons may prevent the application of the higher-level information gathering methods (further simulation, prototype construction, testing, etc.) on a set of more than 5–10 solutions. Consequently, we consider the current algorithms as either too domain-specific (category 1 or 2) or too general (category 3), because a huge set of candidate solutions is returned. Naturally, pruning the set using the DM's preferences is the obvious remedy for this drawback. However, this approach poses another problem because it may be difficult for the DM to state his preferences as explicit decision making rules. In our view, selection from a huge set or pruning the set tend to make the DM focus on the performance (objective space) and neglect the design differences (search space). In contrast, only investigating a few solutions promotes a better balance between performance and design. Furthermore, the low number of solutions allows the DM to apply preferences, use decision rules, and evaluate objectives not stated explicitly.

The MODCO approach address these challenges by incorporating *generalized preferences* into the algorithm with the goal of finding a small set of 5–10 distinct candidates to make step 2 manageable *without* stating explicit preference relations. In MODCO, the concept *generalized preferences* covers a *priori* considerations that are relevant to most if not all real-world applications. This analysis answers the following questions (further elaborated in Sections 2 and 3):

1 *Number of candidates:* $K_{NC} \in [1 : \infty] \subseteq \mathbb{N}$
How many candidates is it practically and economically feasible to inspect, analyze, and compare in post-processing?

2 *Performance distinctiveness:* $K_{PD} \in [0.0 : 1.0] \subset \mathbb{R}$
How different should the candidates be performance-wise?

3 *Design distinctiveness:* $K_{DD} \in [0.0 : 1.0] \subset \mathbb{R}$
How different should the candidates be design-wise?

4 *Simulator accuracy:* $K_{SA} \in [0.0 : 1.0]^M \subset \mathbb{R}^M$
What is the accuracy of the involved simulators?

In MODCO, the parameters K_{NC}, K_{PD}, K_{DD}, and K_{SA} constitute the generalized preferences, and they may be implemented as the secondary selection criterion in an algorithm. Hence, MODCO algorithms aim at reducing the DM's task in step 2 by dividing the higher-level information

into two groups, *generalized preferences* as an a priori analysis to step 1 and the *domain-specific information gathering* as a precursor to the decision making in step 2. In this, the domain-specific information gathering includes further investigations such as visual inspection, detailed simulation, and prototype testing on the distinct candidates followed by evaluation of the DM's implicit or explicit preferences regarding the objectives. Thus, a MODCO algorithm combines category 1 and 3 by integrating the generalized preferences a *priori* and leaving the domain-specific part to a *manageable* second step a *posteriori*.

The paper is structured as follows. Section 2 provides the motivation for the MODCO approach by summarizing 6 years of observations from real-world industrial MO problems. Section 3 lists the features of the ideal MODCO algorithm and the goals of MODCO. After having introduced the MODCO ideas, we provide a survey of related research in Section 4. Finally, Section 5 concludes the paper.

2. Motivation for MODCO Algorithms

The application of multi-objective optimization in an industrial context raises a number of interesting challenges, dilemmas, and unforeseen obstacles. The following observations are gathered from more than 6 years of work at Grundfos R&T solving more than 30 multi-objective optimization tasks in very diverse engineering disciplines including fluid mechanics, motor design, motor control, structural mechanics, electronics, robust design optimization, value-chain optimization, production optimization, etc. Discussions with the DMs involved in these projects have lead to a number of considerations that form the basis for the MODCO approach. In short, the arguments for incorporating generalized preferences into the optimization algorithm fall in three categories; 1) post-processing many Pareto-optimal solutions, 2) physical realization of a solution, and 3) decision making among large sets of solutions. These categories are discussed and elaborated in the following sections.

2.1 Post-Processing Many Pareto-Optimal Solutions

In an industrial application context, the multi-objective optimization is often only a small step in a large design process. Hence, the solutions found by an MO algorithm need to be further investigated to verify each solution against results from other parts of the design process. Consequently, post-processing a large number of solutions is typically infeasible for the following reasons: i) It is too expensive, ii) it is too time-consuming, and iii) the optimization problem may only cover a part of the full design.

Regarding economy, it is often rather expensive to construct a physical prototype of a simulated design. For instance, a 3D prototype print and performance test of a single pump housing may cost up to 5000 Euro. Thus, investigating 100 candidates is out of the question.

Concerning time, a large set of candidate solutions is also problematic if it takes a lot of time to post-process a single solution. As mentioned above, the optimization is often only a small step in a larger design process and further extended simulation may be needed to verify the design in greater detail. For example, conducting a full-range CFD simulation on a design may take several days or weeks, which makes post-processing of large sets impossible. Furthermore, such an investigation often only constitutes a part of the simulations carried out on a design. Other types of analyses include stress analysis, cost calculation, robust design investigations, etc.

In addition to economy and time, the optimization may only be addressing a part of the total design. For example, the optimization problem may represent a sub-circuit of a pump controller circuit. Thus, additional unmodeled features may have to be handled in post-processing. The reasons for only looking at a partial problem are numerous. Some points include: i) No sufficiently accurate model exists, e.g., electromagnetic noise in relation to circuits, ii) impossible to build a formal model of the consequences of a design, iii) misguiding of the search by problematic pairing of a high-accurate model for some objectives with a low-accurate model for other objectives, and iv) a desire to run the development project in parallel sub-teams, e.g., design pump hydraulics and electromotor at the same time.

2.2 Physical Realization of a Single Solution

In an industrial design process, the simulation of a solution is usually followed by a prototype testing and then finally creation of machining tools for mass production. These three steps: i) simulation, ii) prototype testing, and iii) mass production each pose some challenges that make multiobjective optimization in its traditional form less favorable.

Regarding simulation, finding a large Pareto-optimal set is not always meaningful. Two nearby solutions may differ by e.g. 0.1% in performance, but the simulator may have an expected inaccuracy in the range of 1%–5% when the simulated solution is compared with a physical test of the design. Hence, it does not make sense to report a lot of performance-wise similar solutions since prototype testing of them may prove them to be statistically equal.

In prototype testing, one problem is to make a 100% accurate representation of the simulated design. All prototyping methods have tolerances. Consequently, there will be small differences between the simulated design and a corresponding prototype. In addition to this, no testing equipment is 100% accurate. Thus, repeating test will not give the exact same result.

Finally, preparing the design for mass production also renders some challenges. In simulation and prototyping, the "design changes" were caused by inaccuracies in simulator, prototype construction, and measurement. However, realization of the design may require small changes in the design, e.g., to make the part castable with a bi-directional casting process. Furthermore, mass produced parts also are not completely identical since all production processes introduce small variations and it is very costly to approach a 0% tolerance. Furthermore, the process itself may introduce some random variation such as size of welding seams which is not modeled in the simulation. Hence, the produced parts differ from the simulated even if no deliberate modifications have been made to adapt the design to production.

2.3 Decision Making Among Large Sets of Solutions

Challenges related to decision making often come as either i) an abstract or general statement from a development project such as "we would like to evaluate the pros and cons of 3–5 clearly different designs", ii) local selection among similar Pareto-optimal solutions, or iii) as the DM's difficulties in interpreting the results and stating explicit preference rules.

General statements from development projects are often seen in concept studies where it is important to investigate and compare significantly different designs instead of fine-tuning the performance of a design. Hence, the DM is more interested in a small set of clearly different designs rather than a large set covering a smooth transition.

Concerning the local selection among Pareto-optimal solutions, DMs tend to favor solutions residing in the so-called knee regions of the Pareto front if no domain-specific preferences can be determined a priori. Hence, solutions not in knee regions can be omitted from the set of returned solutions as the DM is typically not willing to accept a large decrease in one objective to gain a small increase in another.

Regarding the DM's abilities, it is often the case that such persons have a non-optimization background and in some cases even a non-technical background. Thus, the DM may not have the technical skills to make the right decision among a set of 500–1000 solutions represent-

ing a smooth transition from one design type to another. Presenting a low number of 5–10 clearly different solutions is more feasible because it allows the DM to inspect each solution in great detail and make a decision without explicitly stating preference relations.

3. Features of the Ideal MODCO Algorithm

The observations presented in the previous section can be condensed into a number of desirable features for the ideal MODCO algorithm:

1 *Return Pareto-optimal solutions.*
This is identical to goal 1 of traditional MO algorithms.

2 *User-defined maximal number of returned solutions.*
Resources such as money and time sets a firm limit on the number of solutions manageable in decision making. Any MODCO algorithm should have an upper limit on number of returned solutions.

3 *User-defined distinctiveness of the returned solutions.*
The application often state if the goal is to perform an exploratory search, to return a set of fine-tuned similar alternative solutions, or something in between. Furthermore, the DM or domain expert typically knows if the goal is to find clearly different designs (diversity in search space), performance-wise different solutions (diversity in objective space), or a combination of these two.

4 *User-defined accuracy of simulators.*
The accuracy of the simulators plays a key role in the later decision making step. The MODCO algorithm should incorporate any known simulator accuracies to ensure that the differences in simulated performance also show in later post-processing. More precisely, if A dominates B in optimization, then this should hold for subsequent steps (prototyping, test, etc.) in the design process.

5 *Return solutions in knee regions or according to user-defined preferences.*
DMs are rarely (if ever) interested in solutions not located in knee regions of the Pareto front unless domain-specific preferences can be defined. Hence, the ideal MODCO algorithm should return solutions located in knee regions or alternatively solutions that comply with the user-defined preferences.

Feature 1 is obviously required by any multi-objective algorithm. Features 2–4 constitute general preferences that can be analyzed and defined *a priori* to the optimization. Feature 5 is motivated by observations on

the decision making process in more than 30 multi-objective optimization tasks carried out at Grundfos R&T. Please note that feature 3 *does not* express the DM's preferences regarding the objectives importance, but only *how* diverse the returned solutions should be.

Regarding features 2–4, we define the user-variables for setting these requirements on the returned set. Table 1 lists the four variables supporting the above mentioned features. Naturally, it is up to the MODCO algorithm to transform the given value to an actual *approach* for implementing the feature. The parameters K_{DD} and K_{PD} for defining distinctiveness should be interpreted as shown in Fig. 2. As seen, a value of 0.0 indicates a user-preference for a low distinctiveness, which corresponds to a preference for very similar solutions. A value of 1.0 corresponds to high distinctiveness, i.e., very different solutions. An intermediate value of, e.g., 0.5 represents a desire for a medium level of distinctiveness. In contrast to methods with explicit formulation of preferences, this approach allows incorporation of rather vague statements from the DM or domain expert. For example, a domain expert may say *"For this problem, I know that many somewhat different solutions have roughly the same performance."* In MODCO, such a statement can be transformed into $K_{PD} = 0.0$ meaning "roughly same performance" and $K_{DD} = 1.0$ or perhaps $K_{DD} = 0.5$ representing a desire for highly or somewhat different solutions. To allow refined control over performance distinctiveness, the K_{PD} parameter may also be stated as a vector. In this case, the DM may specify the level of distinctiveness for each objective.

Table 1. The list of user-variables supporting feature 2–4.

Feat.	Description	Variable
2	No. of distinct candidates	$K_{NC} \in [1 : \infty) \subseteq \mathbb{N}$
3	Design distinctiveness	$K_{DD} \in [0.0 : 1.0] \subset \mathbb{R}$
3	Performance distinctiveness	$K_{PD} \in [0.0 : 1.0] \subset \mathbb{R}$ or $[0.0 : 1.0]^M \subset \mathbb{R}^M$
4	Simulator accuracy	$K_{SA} \in [0.0 : 1.0]^M \subset \mathbb{R}^M$

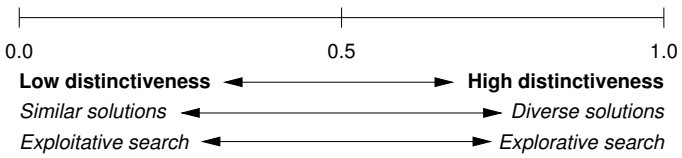


Figure 2. Distinctiveness range for parameters K_{DD} and K_{PD}.

Regarding simulator accuracy, the vector K_{SA} contains a value for each objective where 1.0 corresponds to a 100% accurate simulator and 0.0 means the returned result is no better than a random value.

3.1 Three Goals of MODCO Algorithms

The previous sections containing motivation and MODCO features form a basis for reformulating the three goals of traditional MO algorithms, closeness, distribution and spread, to goals for MODCO algorithms. In short, we suggest the following three goals:

1 *Closeness* – the distinct candidates should be on the true Pareto front (or as close as possible).

2 *Global distinctiveness* – the candidate set should have a user-defined max size K_{NC} and contain clearly distinct solutions found in accordance with K_{DD}, K_{PD}, and K_{SA}.

3 *Local multi-objective optimality* – each candidate should be of primary interest to the DM, e.g., in a knee region if no explicit preferences are given.

Goal 1 of MODCO algorithms is to find solutions on the true Pareto front. This goal represent feature 1 and is identical to goal 1 in MO.

Goal 2 represents the DM's desire for a set of clearly distinct solutions with a user-defined maximal number of solutions in the set. The *a priori* analysis of feature 2–4 can be summarized in one goal stating how many and how different the returned solutions should be.

Goal 3 expresses the observation mentioned in Section 2.3 regarding the DM's preference for solutions located in knee regions of the Pareto front or alternatively in accordance with the DM's explicit preferences. Thus, goal 3 represents feature 5 of the ideal MODCO algorithm.

4. Survey of MODCO-Related Algorithms

The MODCO approach was first studied in our earlier paper [10]. The paper contains a brief introduction to the approach, the Cluster-Forming Differential Evolution (CFDE), and a detailed study on performance and effects of changing the MODCO parameters K_{PD} and K_{DD}. A real-world study on mechanical and electrical engineering problems is presented in [11].

The five features of the ideal MODCO algorithm have received some attention by other researchers.

Feature 1, closeness to true Pareto front, is undoubtedly the most investigated since it is one of the goals of traditional MO research, for surveys see [6, 7, 17].

Feature 2 about returning a limited set of distinct solutions has been investigated in the approach known as "modeling to generate alternatives (MGA)" suggested by Brill [3] and later applied in several studies [5, 4, 13, 19]. In short, the main purpose in MGA is to present a set of maximally different solutions to allow the DM to evaluate non-modeled objectives and implicit preferences and decision rules. The idea in MGA is to first find a starting solution, then define a maximal allowed drop in fitness(es), and search one-by-one for other solutions that are maximally different in search space from previously found solutions while fulfilling the constraint of having the fitness(es) above the defined relaxation target. Although applicable for generating alternatives, the approach excludes the prime paradigm of MO – to find trade-offs wrt. the objectives. In the MGA approach, a solution just has to lie in the hypercube spanned by the relaxation thresholds. Thus, finding the maximally different solutions to accommodate the *unmodeled* objectives may result in suboptimal solutions wrt. the *modeled* objectives. Feature 2 has also been investigated in studies incorporating the DM's preferences into the algorithm, for a survey see [8]. To the authors' knowledge, no paper suggests to integrate *general* preferences into the multi-objective algorithm with the goal of returning a user-defined maximal number of candidate solutions. However, several authors suggest to modify a traditional MO algorithm with various kinds of subpopulation schemes to increase the search performance or parallelize the MOEA, for example [2, 15]. Regarding on-the-fly clustering of solutions, Koch and Zell suggest the MOCS algorithm [12]. MOCS cluster the solutions in objective space with the aim of diversifying the search.

Feature 3 regarding setting a user-defined design and performance distinctiveness is to the authors knowledge also not investigated. Generally, traditional MO research focuses at obtaining an even population distribution, but not distinctiveness among a small set of candidates.

Feature 4 on simulator accuracy has, to some extent, been investigated in a related form known as robust design optimization. It should be noted that this research primarily deals with the manufacturing step of the design process mentioned in Section 2.2, i.e., limiting the problems originating from the tolerances in mass production. Several papers have been published on robust design and some methods incorporate the link back to the design variables, e.g., Li's work on robust optimization [14].

Feature 5 about finding solutions in knee regions has been investigated by a couple of authors in the MO research community. The most

interesting study is presented by Branke et al. [1] who define two metrics (angle and a utility function) for detecting knee regions and favor solutions in knee regions. Another study is published by Rachmawati and Srinivasan [16] who suggest using a weighted sum similar to the utility function of Branke et al. [1]. Regarding incorporation of user-preferences, numerous studies have been performed, see [8].

5. Conclusions

In this paper, we have introduced the MODCO approach, which is motivated by a number of observations from real-world problems from the manufacturing industry. The goal in MODCO is to return a small set of clearly distinct candidate solutions to allow a feasible in-depth post-processing of the candidates. The main idea in MODCO is to refine the multi-objective search by incorporating general preferences from the DM and the domain expert's knowledge of the problem at hand. Through a few parameters, rather vague statements from the DM or domain expert are transformed into desired characteristics of the returned set of solutions. In addition to this, the MODCO approach allows incorporation of any known simulator inaccuracies to further refine the search.

The MODCO approach may be seen as an extension or modification of the traditional MO concept. Thus, a reformulation of the three goals closeness, distribution, and spread of traditional MO is also in place. The three MODCO goals are closeness, global distinctiveness, and local multi-objective optimality. The closeness goal states a desire to find solutions on the Pareto front, which is identical to the first goal of traditional MO. The global distinctiveness goal states that the returned set of candidate solutions should comply with the user's general preferences on number of solutions, performance distinctiveness, design distinctiveness, and known simulator accuracy. Finally, the local multiobjective optimality goal expresses the DM's preferences for finding solutions in knee regions or according to domain-specific preferences.

References

[1] J. Branke, K. Deb, H. Dierolf, and M. Osswald. Finding knees in multi-objective optimization. In *Proc. 8th International Conference on Parallel Problem Solving from Nature (PPSN)*, pages 720–729, 2004.

[2] J. Branke, H. Schmeck, K. Deb, and R. Maheshwar. Parallelizing multi-objective evolutionary algorithms: Cone separation. In *Proc. IEEE Congress on Evolutionary Computation (CEC)*, pages 1952–1957, 2004.

[3] E. D. Brill Jr. The use of optimization models in public-sector planning. *Manage. Sci.*, 25(5):413–422, 1979.

[4] E. D. Brill Jr., S.-Y. Chang, and L. D. Hopkins. Modeling to generate alternatives: The hsj approach and an illustration using a problem in land use planning. *Manage. Sci.*, 28(3):221–235, 1982.

[5] E. D. Brill Jr., J. M. Flach, L. D. Hopkins, and S. Ranjithan. Mga: A decision support system for complex, incompletely defined problems. *IEEE T. Syst. Man Cyb.*, 20(4):745–757, 1990.

[6] C. A. Coello Coello. A comprehensive survey of evolutionary-based multiobjective optimization techniques. *Knowl. Inf. Syst.*, 1(3):269–308, 1999.

[7] C. A. Coello Coello. An updated survey of GA-based multiobjective optimization techniques. *ACM Comput. Surv.*, 32(2):109–143, 2000.

[8] C. A. Coello Coello. Handling preferences in evolutionary multiobjective optimization: A survey. In *Proc. IEEE Congress on Evolutionary Computation (CEC)*, vol. 1, pages 30–37, 2000.

[9] K. Deb. *Multi-Objective Optimization using Evolutionary Algorithms.* John Wiley & Sons, Chichester, 2001.

[10] P. D. Justesen and R. K. Ursem. Multiobjective distinct candidates optimization (modco): A cluster-forming differential evolution algorithm. In *Proc. Fifth International Conference on Evolutionary Multi-Criterion Optimization (EMO)*, vol. 1, pages 525–539, 2009.

[11] P. D. Justesen and R. K. Ursem. Preference-based multi-objective distinct candidates optimization. In *Proc. Fourth International Conference on Bioinspired Optimization Methods and their Applications (BIOMA)*, pages 117–129, 2010.

[12] T. E. Koch and A. Zell. MOCS: Multi-objective clustering selection evolutionary algorithm. In Proc. Genetic and Evolutionary Computation Conference (GECCO), pages 423–430, 2002.

[13] D. H. Loughlin, S. R. Ranjithan, E. D. Brill Jr., and J. W. Baugh Jr. Genetic algorithm approaches for addressing unmodeled objectives in optimization problems. *Eng. Optimiz.*, 33(5):549–569, 2001.

[14] M. Li. Robust optimization and sensitivity analysis with multi-objective genetic algorithms: Single- and multi-disciplinary applications. Ph.D. dissertation, University of Maryland, 2007.

[15] A. Molyneaux, G. Leyland, and D. Favrat. A new, clustering evolutionary multiobjective optimisation technique. In *Proc. Third International Symposium on Adaptive Systems, Evolutionary Computation and Probabilistic Graphical Models*, pages 41–47, 2001.

[16] L. Rachmawati and D. Srinivasan. A multi-objective evolutionary algorithm with weighted-sum niching for convergence on knee regions. In *Proc. Conference on Genetic and Evolutionary Computation (GECCO)*, pages 749–750, 2006.

[17] M. Reyes-Sierra and C. A. Coello Coello. Multi-objective particle swarm optimizers: A survey of the state-of-the-art. *Int. J. Comput. Intel. Res.*, 2(3):287–308, 2006.

[18] D. A. V. Veldhuizen and G. B. Lamont. Multiobjective evolutionary algorithms: Analyzing the state-of-the-art. *Evol. Comput.*, 8(2):125–147, 2000.

[19] E. Zechman and S. R. Ranjithan. An evolutionary algorithm to generate alternatives (eaga) for engineering optimization problems. *Eng. Optimiz.*, 36(5):539–553, 2004.

PARAMETER ESTIMATION IN AN ENDOCYTOSIS MODEL WITH BIOINSPIRED OPTIMIZATION ALGORITHMS

Katerina Tashkova, Peter Korošec, Jurij Šilc
Computer Systems Department
Jožef Stefan Institute, Ljubljana, Slovenia
{katerina.taskova; peter.korosec; jurij.silc}@ijs.si

Ljupčo Todorovski
Faculty of Administration
University of Ljubljana, Slovenia
ljupco.todorovski@fu.uni-lj.si

Sašo Džeroski
Department of Knowledge Technologies
Jožef Stefan Institute, Ljubljana, Slovenia
saso.dzeroski@ijs.si

Abstract The (re)construction of biological systems is of fundamental importance to the emerging field of systems biology. Because of the complexity of these systems, computational systems biology strongly relies on the principles of mathematical modeling as an essential tool for determining the behavior of numerous and simultaneous time- and space-dependent processes. In general, the considered models are parametric and the unknown parameters have to be estimated using experimental data, a task known as parameter estimation. Parameter estimation is essentially an optimization task, that in the highly nonlinear and constrained dynamics of biological models can turn into a hard problem for traditional local search optimization methods. Motivated by this challenge, the paper addresses the task of parameter estimation in a nonlinear dynamic model (described by ordinary differential equations) with 18 parameters: The model describes a key cell regulatory system that switches between cargo transport and maturation in early, respectively late endosomes

in the endocytosis pathway. We approached the problem by using two bio-inspired metaheuristics for global optimization and one direct local search method for maximum-likelihood optimization. To asses the performance of the applied methods, the parameters were estimated from pseudo-experimental (simulated) data.

Keywords: Metaheuristic optimization, Ordinary differential equations, Parameter estimation

1. Introduction

The (re)construction of biological systems is of fundamental importance to the emerging field of computational systems biology. In general, these systems exhibit complex nonlinear dynamic, which is usually, modeled by ordinary differential equation (ODE) models. In a typical approach to ODE modeling of a biological system, a human domain expert specifies the structure of the system and the functional form of the ODEs. The task of determining appropriate values for the constant parameters based on time course data (observed or simulated) is called parameter estimation.

Due to the nonlinear and constrained systems dynamics, parameter estimation of biological models is quite demanding and computationally expensive. The problem is usually multimodal (local search methods fail to find the global solution), high-dimensional (computationally expensive) and constrained in the parameter space, black-box (unknown structure of the system), and further confounded by short time course noisy data (experiments in cell and molecular biology). Related work in the domain of parameter estimation in system biology [12] has shown that, to overcome the above mentioned difficulties, global optimization (GO) methods should be used, with a focus on stochastic GO (as solvers of black-box real-size problems) and hybrid methods (GO with local search solvers).

This work addresses the task of parameter estimation in a model of the endocytosis dynamics, a key regulatory system that switches between cargo transport and maturation in early, respectively late endosomes [11, 15]. The underlying process is the conversion of the Rab5 protein to the Rab7 protein, where the Rab5 and Rab7 domain proteins are mutually exclusive. The theoretical and experimental approach undertaken to model this process [3], proved that a cut-out switch ODE model best fits the biological observations.

Assuming the cut-out switch structure and given the initial condition for Rab species, we reconstructed the parameters of the original study, based on simulated pseudo-experimental data, by employing a recently

proposed metaheuristic Differential Ant-Stigmergy Algorithm (DASA) [6, 7, 8]. Since DASA has never been used for parameter estimation in ODE systems, we compare the results with the already well established metaheuristic Differential Evolution (DE) [13] and the direct local search method Algorithm 717 [1] for nonlinear parameter estimation. The optimization problem was formulated as a maximum-likelihood problem, under the assumption of normally distributed and independent residuals with constant variance. This maps the problem into a least-squares problem, where we search for parameter values minimizing the sum of squared (residuals) errors.

The rest of the paper is structured as follows. Section 2 defines the parameter estimation task in general and in the specific case of ODE-based models. Section 3 addresses the parameter estimation problem in the Rab5-to-Rab7 conversion model. Section 4 describes in brief the methods used for parameter estimation in the endocytosis model. Section 5 first describes the setup of the empirical evaluation, then presents and discusses the obtained results and finally outlines possible directions for further work. Section 6 summarizes this study and concludes.

2. Parameter Estimation in ODE Models

Given are a model structure $m(c)$, which includes a set of adjustable parameters $c = \{c_1, \ldots, c_D\}$, and a set of observation data d. The task of parameter estimation is to fit the model parameters to values c^{opt} that define a model which reproduces the observed data in the best possible way. This is performed by minimizing a cost (objective) function that measures the goodness of fit.

2.1 Non-Linear Least-Square Estimation

Among several suggested cost metrics, the maximum-likelihood estimator [5] introduced by R. A. Fisher in 1912, stands out for being the one that maximizes the probability of observing the given data d if the model $m(c^{\text{opt}})$ is chosen. The likelihood function depends on the probability of the measurements. Assuming the measurements follow a product of normal distributions with constant variance, the maximum-likelihood parameter estimation maps into a non-linear least-square estimation of the parameters, which minimizes the sum of squared errors (residuals) given as

$$\text{SSE}\left(m(c)\right) = \sum_{i=1}^{M} \sum_{j=1}^{N} \left(Y_i[j] - \hat{Y}_i[j] \right)^2$$

for observed data $d = \{Y_i[j], 1 \leq i \leq M, 1 \leq j \leq N\}$, where M is the number of measured outputs, N is the number of samples per observed output, and $\hat{Y}_i[j]$ is the j-th sample of the i-th predicted output.

2.2 Simulation of ODE Models

We represent the underlying problem as a system described with a set of input/output, divided into *exogenous variables*, E, and *system variables*, S. More precisely, E are observed variables on which the model depends (that are on the right-hand side of the ODEs) and S are dependent variables that we would like to model (the left-hand side of the ODEs). Then the dynamic behavior F of the system described with ODEs, given initial values $S(t_0)$ and $E(t)$ on the observed time interval $[t_0, t_{N-1}]$, will be of the form

$$\frac{d}{dt}S = (S(t), E(t), c)\,,$$

where t is time, the variable on which the derivation is applied and c is the set of fitted parameters.

Note that, the system variables might be additionally classified as observed/unobserved. If the variable can be measured it is called observed, and if it can not be measured/observed it is called unobserved. The output from the ODE model in general can be formulated as

$$\hat{Y}(t) = (S(t), E(t)).$$

This means that the evaluation of the chosen ODE-based model, defined by the parameter c, depends indirectly on the calculation of the system variables. In order to find $S(t)$, numerical approximation methods for ODE integration have to be applied. Therefore, we used the CVODE package [2], a general-purpose solver written in C for the initial value problem of stiff/non-stiff ODEs. CVODE uses the adaptive-step Adams-Moulton and backward differentiation formula method for integration.

3. Endocytosis Model

This work addresses the task of parameter estimation in a specific endocytosis model. The model captures the cellular mechanisms of endosome maturation and cargo transport. It is based on biochemical protein-protein interactions of the Rab5 and Rab7 protein domains [11, 15].

The theoretical and experimental approach undertaken to model the endocytosis rely on the mutually exclusiveness of the Rab5 and Rab7 domains. It has proved that the cut-out switch model best fits the

biological observations [3]. This model, defined by four ODEs and 18 kinetic parameters, describes the behavior of four variables (species), that is the active (GTP-bound) and inactive (GDP-bound) forms of the Rab5 and Rab7 proteins. The variables are r_5 (Rab5-GDP), R_5 (Rab5-GTP), r_7 (Rab7-GDP), and R_7 (Rab-GTP).

The mathematical formulation of the Rab5-to-Rab7 conversion is given below, where $v_1, \ldots, v_{10}$ denote different biochemical reactions in which the observed Rab5 and Rab7 proteins are interacting, while $c_1, \ldots, c_{18}$ are the kinetic rates that are to be estimated. The model formulated in this form does not depend on any independent input, meaning the system is completely defined by four system variables and no exogenous variables. Further, we assumed that the system is completely observed and that the system variables represent the system output.

$$
\begin{array}{lcllcl}
v_1 & = & c_1 & v_2 & = & \frac{c_2\ r_5\ t}{(100+t)(1+e^{(c_3-R_5)\ c_4})} \\
v_3 & = & c_5\ r_5 & v_4 & = & c_6 \\
v_5 & = & \frac{c_7\ r_7\ R_7^{c8}}{c_9+R_7^{c8}} & v_6 & = & \frac{c_{10}\ r_7}{1+e^{(c_{11}-R_5)\ c_{12}}} \\
v_7 & = & \frac{c_{13}\ R_5}{1+e^{(c_{14}-R_7)\ c_{15}}} & v_8 & = & c_{16}\ r_7 \\
v_9 & = & c_{17}\ R_5 & v_{10} & = & c_{18}\ R_7
\end{array}
$$

$$
\begin{aligned}
\tfrac{d}{dt} r_5 &= v_1 + v_7 + v_9 - v_2 - v_3 \\
\tfrac{d}{dt} R_5 &= v_2 - v_7 - v_3 \\
\tfrac{d}{dt} r_7 &= v_4 + v_{10} - v_5 - v_6 - v_7 \\
\tfrac{d}{dt} R_7 &= v_5 + v_6 - v_{10}
\end{aligned}
$$

In order to compare the performance of different optimization algorithms for the given problem, artificial experimental data were generated by simulation of the model with the parameters values c^* set as follows:

$c_1 = 1$	$c_2 = 0.3$	$c_3 = 0.1$	$c_4 = 2.5$	$c_5 = 1$	$c_6 = 0.483$
$c_7 = 0.21$	$c_8 = 3$	$c_9 = 0.1$	$c_{10} = 0.021$	$c_{11} = 1$	$c_{12} = 3$
$c_{13} = 0.31$	$c_{14} = 0.3$	$c_{15} = 3$	$c_{16} = 0.483$	$c_{17} = 0.06$	$c_{18} = 0.15$

and initial conditions $S^*(t_o)$, suggested as optimal by Del Conte-Zerial et al. [3].

$$
\begin{aligned}
S^*(t_o): \quad & r5[t_0] = r7[t_0] = 1 \tfrac{\text{mol}}{\text{l}} \\
& R5[t_0] = R7[t_0] = 0.001 \tfrac{\text{mol}}{\text{l}}
\end{aligned}
$$

Moreover, we defined two versions of the problem, P_1 and P_2:

P_1, 18-dimensional optimization problem with given initial conditions $S^*(t_0)$ and parameters bounds $c_i \in (0,4), 0 \leq i \leq 18$,

P_2, 22-dimensional problem defined with $c_i \in (0,4), 0 \leq i \leq 18$, and initial conditions of the species taken as additional parameters to be estimated, $c_i \in (0,1.7), 19 \leq i \leq 22$.

Having in mind that the simulated data are exact data (generated in the absence of noise), unlike real experimental data, we also considered the parameter estimation problems under normal Gaussian noise, $N(0,1)$, relatively added to the exact data in a quantity defined by the percentage factor s

$$Y_{\text{noisy}} = Y\ (1 + s\ N(0,1)).$$

4. Methods

This section describes the optimization approaches used to solve the nonlinear parameter estimation problem in the Rab5-to-Rab7 conversion endocytosis model. We chose one deterministic local search optimization method for solving non-linear least-squares problems, and two bioinspired population-based metaheuristics for global, robust optimization.

4.1 Algorithm 717

Algorithm 717 (ALG717) denotes a set of FORTRAN 77 modules for solving the parameter estimation problem in nonlinear regression models like nonlinear least-squares, maximum likelihood and some robust fitting problems. The basic method is a generalization of NL2SOL – An Adaptive Nonlinear Least-Squares Algorithm [4], which uses a model/trust-region technique for computing trial steps along with adaptive choice of the Hessian model. The algorithm is a variation of the Newton's method (augmented Gauss-Newton method), in which a part of the Hessian is computed exactly and a part is approximated by a secant (quasi-Newton) updating method. So the algorithm sometimes reduces to the Gauss-Newton or Levenberg-Marquardt method.

To promote convergence from poor starting guesses, the method employs the idea of having a local quadratic model q_i of the objective function f at the current best solution c_i and an estimate of an ellipsoidal region centered at c_i in which q_i is trusted to represent f. So the next point, c_{i+1}, or the next trial step, is chosen to approximately minimize q_i on the ellipsoidal trust-region. The information obtained for f at c_{i+1} is used for model updating and also to resize and reshape the trust region.

Among the modules, one can chose the variants DGLF and DGLG for solving unconstrained optimization, or DGLFB and DGLGB that use simple bounds constraints on the parameters. Using the DGLF and the DGLFB modules means approximate computation of the needed derivatives by finite differences, while the DGLG and DGLGB modules expect the derivatives of the objective function to be provided by the routine that calls them.

Since ALG717 is not a global search algorithm, we wrapped the original procedure in a loop of restarts with randomly chosen initial points, providing in some way a simple global search. The number of restarts was defined to result in an equal number of function evaluations as set for the other two methods.

4.2 Differential Evolution

Differential evolution (DE) is a simple and efficient population-based heuristic for optimizing real-valued multi-modal functions, introduced by Storn and Price in the 1990s [13]. It belongs to the class of Evolutionary algorithms (EA) inspired by the nature of evolution, meaning it is based on the idea of simulating the evolution of individuals (candidate solutions) via processes of selection, mutation and crossover.

The main difference between standard EA and DE is in the reproduction step, where for every candidate solution an offspring is created with a simple arithmetic (differential) mutation operation over three (or more) parents.

In addition, the rate at which the population evolves can be controlled by a scale factor, F, a user-defined positive real number. To complement the differential mutation strategy, DE employs uniform crossover (also known as discrete recombination) over the candidate and mutated solutions, where a user-specified crossover factor, $CR \in [0, 1]$, is used to control the fraction of parameter values copied form the mutated solution. Finally, the offspring solution is evaluated and substitutes its parent in the population if its fitness is better.

Depending on the specific mutation and crossover procedure, one can chose among several DE schemas "DE/x/y/z". Here, "DE/x/y/z" indicates DE for Differential Evolution, x represents a string denoting the vector to be perturbed (rand: random vector; best: best vector), y is the number of difference vectors considered for perturbation of x and z stands for the type of crossover being used (exp: exponential; bin: binomial).

Since the original code of the DE algorithm [10] does not check if the new generated solutions are feasible (within the prescribed bound

constraints) we had to slightly modify the code. The modification is simple: if the solution is outside the specified bounds it is set to the closest bound.

4.3 Diffrential Ant-Stigmergy Algorithm (DASA)

The basic concept of DASA is as follows [6, 7, 8]. First, we translate the multi-parameter problem into a search graph. We then use an optimization technique to find the cheapest path in the constructed graph. This so called offsets-path consists of the values of differences belonging to the optimized parameters.

Prior to the actual optimization, an initial amount of pheromone is deposited according to the Cauchy probability density function in all the vertices in the search graph. There are m ants in a colony, all of which begin simultaneously from the start vertex. The probability with which they choose the next vertex depends on the amount of pheromone in the vertices. The ants repeat this action until they reach the ending vertex. Then, for each ant, based on the chosen offsets path $\Delta x_1 \Delta x_2 \ldots \Delta x_n$ and currently best solution x', a new solution $x = [x'_1 + \Delta x_1, x'_2 + \Delta x_2, \ldots x'_n + \Delta x_n]$ is constructed. If a better solution is found, it replaces the current best solution. Furthermore, in such a case, the pheromone amount is redistributed according to the associated offsets-path that led to this improvement. The new probability density functions have maxima over the path's vertices and the scale factor is accordingly decreased to improve convergence.

Afterwards, pheromone evaporation from all the vertices occurs, i.e., the amount of pheromone is decreased by some predetermined percentage, ρ, on each probability density function. The whole procedure is then repeated until some ending condition is met (e.g., some predetermined number of iterations).

5. Experimental Evaluation

This section describes the setup, the outcome and the evaluation of the initial experiments conducted to investigate the performance of ALG717, DE and DASA when applied to the task of parameter estimation in the non-linear dynamic model of Rab5-to-Rab7 conversion, using exact measured (simulated) data and simulated measurement noise.

5.1 Setup

The setup of the testing procedure included 25 runs of every experiment (combination of algorithm and problem) and a fixed number of

500.000 function evaluations per run. Next, we used a resampling procedure for handling the noise in the measured data, where the "true" model measurements were taken as mean value calculated from 10 different values of the noisy observed output Y_{noisy}. Furthermore, two noise scenarios for $s = 5\%$ and $s = 20\%$ were considered. Finally, based on manual tuning, the algorithms parameters were set as follows.

In ALG717, the DGLGB routine with user-supplied derivatives of the objective function is applied, the number of restarts is set to 25.000 (with 20 evaluations per restart) and 20.000 (25 evaluations per restart) for P_1 and P_2, respectively.

In DE, the strategy "DE/rand-to-best/1/exp" was used, population size was set to $NP = 200$, weight factor to $F = 0.85$, and crossover factor to $CR = 1.0$.

In DASA, the number of ants was set to $m = 8$, the pheromone evaporation factor to $\rho = 0.2$, the maximum parameter precision to $\epsilon = 10^{-15}$, the discrete base to $b = 10$, the global scale increase factor to $s_+ = 0.07$, and the global scale decrease factor to $s_- = 0.02$.

Note that manual tuning included testing of a few parameter combination and was applied to DE and DASA. In order to generate the set of parameters, we discretized the continuous domain of parameter values with a certain step (chosen according to suggestions found in the literature). In this way, we run DE with the following parameters: strategy = ["DE/best/1/exp", "DE/rand/1/exp", "DE/rand-to-best/1/exp", "DE/rand/2/exp", "DE/rand/1/bin", "DE/best/2/bin"], $F = [0.5, 0.6, 0.7, 0.85]$, and $CR = [0.3, 0.5, 0.7, 1.0]$. Population size was fixed to 200 in all cases. For every parameter combination the algorithm was run twice. In a similar way, we executed two runs of DASA on the following sets of parameters: $s_+ = [0.01, 0.03, 0.07]$, $\rho = [0.2, 0.4, 0.6]$, and $m = [4, 6, 8, 10]$. The rest of the parameters were kept fixed to the default values suggested by the authors of DASA.

5.2 Evaluation Criteria

We compare the methods according to the *sum of squared errors* (SSE), as defined in Section 2.1, and the *convergence curve* based on the average results over 25 runs for each problem. In addition, we use the following two performance metrics standardly used for model validation.

The root mean squared error (RMSE) of a model measures the difference between values predicted by the model $\hat{Y}$ and values actually observed from the system being modeled Y, on the same scale as the

modeled outputs.

$$\mathrm{RMSE}(m(c)) = \sqrt{\frac{1}{N}\sum_{i=1}^{M}\sum_{j=1}^{N}\left(Y_i[j] - \hat{Y}_i[j]\right)^2} = \sqrt{\frac{1}{N}\mathrm{SSE}(m(c))}$$

The correlation coefficient (R) is a standard measure that determines how well the predictive model fits the given data in terms of linear dependence and is given as:

$$\mathrm{R} = \frac{1}{M}\sum_{i=1}^{M}\frac{E[(Y_i - E[Y_i])(\hat{Y}_i - E[\hat{Y}_i])]}{\left(E[Y_i - E[Y_i]]\right)^2\left(E[\hat{Y}_i - E[\hat{Y}_i]]\right)^2},$$

where $E[\cdot]$ denotes mathematical expectation. Correlation close to 1 means that (the shape of) the predicted data are identical to the measured data but might have a different scale or baseline. Correlation of -1 means that the predicted data and the measured data are similar in a mirrored fashion, while 0 means dissimilar data.

5.3 Results and Discussion

Experimental results from parameter estimation of the Rab5-to-Rab7 conversion model with ALG717, DE and DASA using simulated experimental (exact and noisy) data are presented in Table 1. The rows Best, Median and Worst represent the values of the measures (SSE, RMSE and R) for the best, median and worst solution found with respect to SSE, a specific method and specific problem, over all 25 runs. In a similar way, the Mean and Std rows outline the average value and standard deviation of the corresponding performance measures with respect to the specific method and specific problem, over 25 runs as well.

Note that the optimum in the case of noise is not zero anymore, as given in Table 1. Since the observed data are artificially generated, including the noise added in the data as described in Section 2, the optimum can be artificially calculated as the sum of squared errors of the noisy observations regarding the non-noisy observations.

According to Table 1, DE comes almost to the optimum (zero error), confirmed by the high (almost 1) correlation in half of the runs while solving both problems using non-noisy data. In all experiments, the DE best solutions were better than the one obtained by DASA and ALG717.

Table 1. Experimental results from parameter estimation of the Rab5-to-Rab7 conversion model based on pseudo-experimental data with and without noise, $N(0,1)$

s [%]		SSE			RMSE			R		
		ALG717	DE	DASA	ALG717	DE	DASA	ALG717	DE	DASA
		problem P_1								
$0^{1)}$	Best	948.55	0.08	9.18	0.58	0.01	0.06	0.288	0.998	0.982
	Median	1375.01	0.62	21.73	0.70	0.02	0.09	0.179	0.981	0.921
	Worst	1679.68	337.98	89.89	0.78	0.35	0.18	0.090	0.495	0.682
	Mean	1366.84	22.69	24.50	0.70	0.04	0.09	0.171	0.948	0.825
	Std	164.68	77.80	16.85	0.04	0.08	0.03	0.087	0.132	0.114
$5^{2)}$	Best	827.60	2.74	8.07	0.55	0.03	0.05	0.255	0.583	0.570
	Median	1378.80	4.43	26.05	0.70	0.04	0.10	0.165	0.578	0.519
	Worst	1524.86	10.30	74.91	0.74	0.06	0.16	0.125	0.517	0.505
	Mean	1330.55	4.66	32.52	0.69	0.04	0.10	0.172	0.575	0.538
	Std	162.88	1.92	19.40	0.04	0.01	0.03	0.068	0.014	0.021
$20^{3)}$	Best	1144.49	42.54	50.18	0.64	0.12	0.13	0.216	0.497	0.495
	Median	1400.29	52.71	63.35	0.71	0.14	0.15	0.211	0.490	0.493
	Worst	1594.18	479.34	89.99	0.76	0.42	0.18	0.111	0.259	0.482
	Mean	1400.06	184.76	67.26	0.71	0.23	0.15	0.140	0.451	0.489
	Std	119.84	170.56	12.08	0.03	0.12	0.01	0.049	0.060	0.004
		problem P_2								
$0^{1)}$	Best	1022.16	0.10	10.99	0.61	0.01	0.06	0.190	0.998	0.850
	Median	1415.33	0.68	41.11	0.71	0.02	0.12	0.080	0.977	0.559
	Worst	1665.88	699.99	88.71	0.77	0.50	0.18	0.114	0.334	0.705
	Mean	1398.96	29.32	39.82	0.71	0.04	0.12	0.141	0.830	0.727
	Std	178.23	139.73	19.79	0.05	0.10	0.03	0.060	0.172	0.111
$5^{2)}$	Best	980.18	2.75	13.07	0.59	0.03	0.07	0.164	0.583	0.571
	Median	1411.62	3.80	41.40	0.71	0.04	0.12	0.101	0.530	0.531
	Worst	1732.96	92.86	104.11	0.79	0.18	0.19	0.097	0.472	0.522
	Mean	1396.33	9.14	45.75	0.71	0.05	0.12	0.128	0.554	0.528
	Std	156.20	18.36	22.46	0.04	0.03	0.03	0.032	0.033	0.020
$20^{3)}$	Best	1131.03	42.74	52.37	0.64	0.12	0.14	0.182	0.498	0.494
	Median	1499.67	55.28	77.69	0.73	0.14	0.17	0.117	0.488	0.484
	Worst	1679.07	909.64	108.07	0.78	0.57	0.20	0.098	0.199	0.474
	Mean	1465.36	220.12	77.01	0.73	0.24	0.17	0.140	0.431	0.486
	Std	141.37	243.64	14.94	0.04	0.14	0.02	0.082	0.095	0.006

[1)] optimal SSE = 0, R = 1 (non-noisy case)

[2)] optimal SSE = 2.6530, R = 0.584

[3)] optimal SSE = 42.4473, R = 0.4982

Nevertheless, the presence of noise in the measured data visibly influenced the performance of DE, especially in the case with 20% of added noise, where the Mean, Std and Worst values of the objective function are far larger than the one obtained by DASA. While DASA is relatively far (2 orders of magnitude) compared to DE best solutions, it finds solutions with almost consistent accuracy (Std) and is far less influenced by the percentage of added noise in the measured data. Both DE and DASA outperform ALG717 based on the statistics in Table 1: the presence of the measurement noise does not influence the ALG717 performance in a visible way (ALG717 is so far from the optimum that noise does not influence the SSE noticeably).

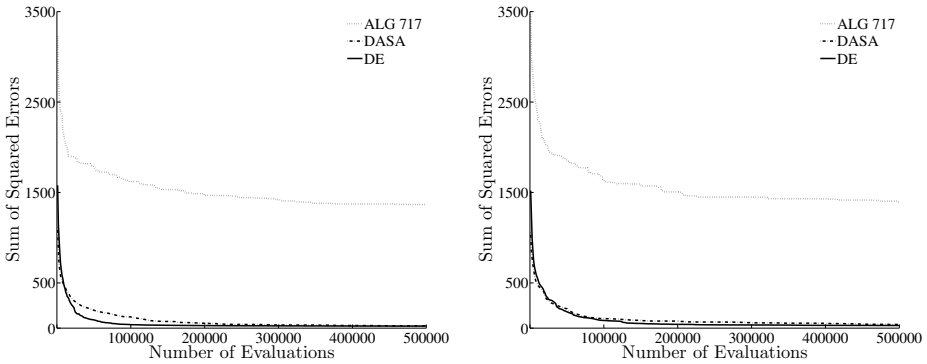


Figure 1. Convergence curves for the non-noisy case: a) problem P_1, b) problem P_2.

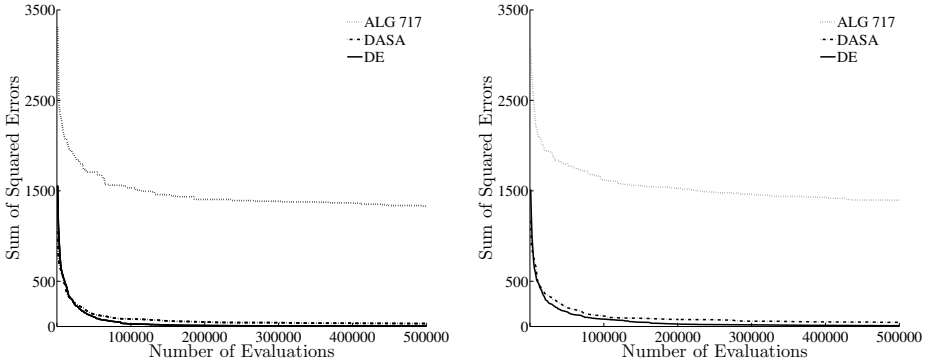


Figure 2. Convergence curves for data perturbed with 5% Gaussian noise: a) problem P_1, b) problem P_2.

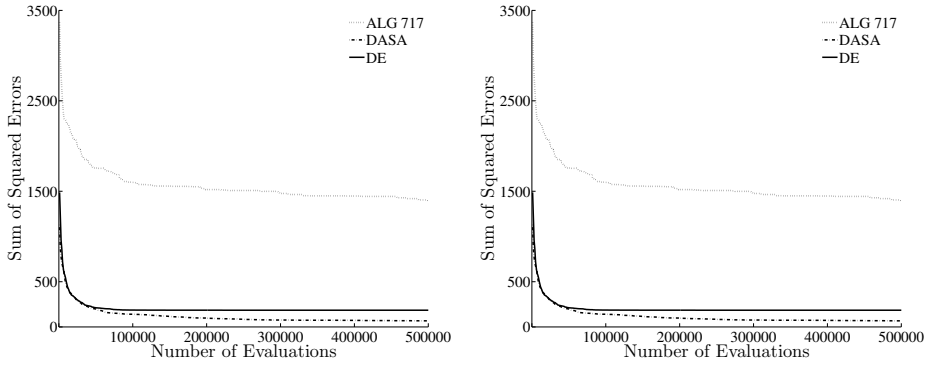


Figure 3. Convergence curves for data perturbed with 20% Gaussian noise: a) problem P_1, b) problem P_2.

The graphs in Figures 1, 2, and 3 represent the convergence curves of the algorithms for the specific problem, based on the mean of the best value of the objective function from 25 runs over the number of evaluations. Based on the convergence performance, DE and DASA outperform ALG717 in all cases. When compared to each other in the case of noise-free data (Fig. 1) and in the case of slight perturbation with 5% of measurement noise (Fig. 2), both have similar convergence with a slightly faster convergence of DE to the optimum. However, in the case of 20% of measurement noise (Fig. 3), DASA shows visibly better convergence than DE.

As our main goal was to reconstruct the dynamic of the Rab5-to-Rab7 conversion model, we visualized the dynamic of the predicted model to validate qualitatively the results from Table 1. Figure 4 gives a comparison of the algorithms on predicting the behavior of the R_5 protein concentration with the best estimated parameters for the P_1 problem obtained using noise-free data and data perturbed with 20% noise, respectively. It is evident that there is a very good correlation between the pseudo-experimental data and the predicted data for DE (almost overlapping) and DASA (the trend and the shape are preserved) in the non-noisy case. Moreover, DE and DASA show also very good performance in the presence of noise, successfully dealing with the measurement noise in parameter estimation (Fig. 4b). The behavior of the other three variables r_5, r_7 and R_7 is quite similar and is not included here. For the same reasons, the visualization of the P_2 dynamics is omitted.

A test of statistical significance was performed to check the difference in the quality of the obtained solutions by ALG717, DE, and DASA.

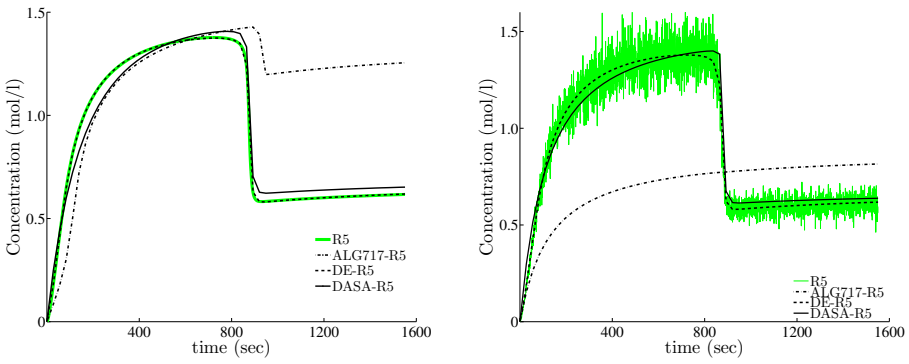


Figure 4. Experimental behavior vs. predicted behavior of Rab5 (R5) protein with the best estimated parameters for the P_1 problem using a) noise-free data, b) data with 20% Gaussian noise

Based on the mean values of SSE, we performed pairwise comparisons using the Wilcoxon signed-ranks test [14]. The test confirms that for a 5% significance level both DE and DASA are significantly better than ALG717 and there is no significant difference in the quality of solutions generated with DE and DASA.

5.4 Future Work

Additional experiments are needed in order to confirm the conclusions drawn from the initial experimental evaluation results. Possible directions for further work could take into consideration:

1 Investigation of the algorithm performance with regard to the number of resampled sets of experimental data. In reality, the reparation of the measurements is determined (limited) by the experimental costs, meaning we have to handle the parameter estimation based on the real data with more robust optimization methods that can cope with sparse and noisy data;

2 Empirical evaluations on real experimental data;

3 Address the problem with state-of-the-art algorithms used for parameter estimation in the systems biology domain [9, 12];

4 Modification of the existing methods to cope with the computationally expensive model simulations, resulting in faster convergence towards the global optimum.

6. Conclusions

This paper presents an initial study on parameter estimation in an ODE-based biological model of the important endocytotic regulatory system from simulated experimental data using bioinspired population based metaheurstics, such as DE and DASA. The comparison with the local search method ALG717, which fails to reconstruct the model parameters, confirmed that bioinspired metaheuristics are powerful methods for global nonlinear multi-dimensional optimization. We would like to emphasise the remarkable performance of DE in the case of noise-free experimental data and the promising results of DASA based on the more accurate parameter estimation in the presence of higher level of noise. The initial results open a wide space of possible directions for further work, starting with validation of the drawn conclusions on real experimental data and continuing with modification of the recently proposed DASA approach to obtain even better convergence.

References

[1] D. S. Bunch, D. M. Gay and R. E. Welsch. Algorithm 717: Subroutines for maximum likelihood and quasi-likelihood estimation of parameters in nonlinear regression models. *ACM T. Math. Software*, 19(1):109–130, 1993.

[2] S. D. Cohen and A. C. Hindmarsh. CVODE, A stiff/nonstiff ODE solver in C. *Comput. Phys.*, 10(2):138–143, 1996.

[3] P. Del Conte-Zerial, L. Brusch, J. C. Rink, C. Collinet, Y. Kalaidzidis, M. Zerial and A. Deutsch. Membrane identity and GTPase cascades regulated by toggle and cut-out switches. *Mol. Syst. Biol.*, 4:206, 2008.

[4] J. E. Dennis, D. M. Gay and R. E. Welsch. Algorithm 573: NL2SOL–An adaptive nonlinear least-squares algorithm. *ACM T. Math. Software*, 7(3):369–383, 1981.

[5] N. Gershenfeld. *The Nature of Mathematical Modeling.* Cambridge University Press, 1999.

[6] P. Korošec and J. Šilc. The Differential Ant-Stigmergy Algorithm for large scale real-parameter optimization. *Lect. Notes Comput. Sc.*, 5217:413–414, 2008.

[7] P. Korošec and J. Šilc. High-dimensional real-parameter optimization using the differential ant-stigmergy algorithm. *Int. J. Intell. Comput. Cybernetics*, 2(1):34–51, 2009.

[8] P. Korošec and J. Šilc. A stigmergy-based algorithm for continuous optimization tested on real-life-like environment. *Lect. Notes Comput. Sc.*, 5484:675–684, 2009.

[9] C. G. Moles, P. Mandes and J. R. Banga. Parameter estimation in biochemical pathways: a comparison of global optimization methods. *Genome Research*, 13:2467–2474, 2003.

[10] K. Price, R. Storn, J. Lampinen. *Differential Evolution - A Practical Approach to Global Optimization.* Springer, 2005.

[11] J. Rink, E. Ghigo, Y. Kalaidzidis, M. Zerial. Rab conversion as a mechanism of progression from early to late endosomes. Cell., 122:735–49, 2005.

[12] M. Rodriguez-Fernandez, J. A. Egea and J. R. Banga. Novel metaheuristic for parameter estimation in nonlinear dynamic biological systems. *BMC Bioinformatics*, 7:483, 2006.

[13] R. Storn and K. Price. Differential Evolution – A simple and efficient heuristic for global optimization over continuous spaces. *J. Global Optim.*, 11:341–359, 1997.

[14] F. Wilcoxon. Individual comparisons by ranking methods. *Biometrics Bull.*, 1:80–83, 1945.

[15] M. Zerial, H. McBride. Rab proteins as membrane organizers. *Nat. Rev. Mol. Cell. Biol.*, 2:107–17, 2001.

HOW SLOW IS SLOW? SFA DETECTS SIGNALS SLOWER THAN THE DRIVING FORCE

Wolfgang Konen, Patrick Koch
Institute for Informatics
Cologne University of Applied Sciences, Gummersbach, Germany
{wolfgang.konen; patrick.koch}@fh-koeln.de

Abstract Slow feature analysis (SFA) is a bioinspired method for extracting slowly varying driving forces from quickly varying nonstationary time series. We show here that it is possible for SFA to detect a component which is even slower than the driving force itself (e.g. the envelope of a modulated sine wave). It depends on circumstances like the embedding dimension, the time series predictability, or the base frequency, whether the driving force itself or a slower subcomponent is detected. Interestingly, we observe a swift phase transition from one regime to another and it is the objective of this work to quantify the influence of various parameters on this phase transition. We conclude that *what* is perceived as slow by SFA varies and that a more or less fast switching from one regime to another occurs, perhaps showing some similarity to human perception.

Keywords: Driving force, Nonstationary time series, Phase transition, Slow feature analysis

1. Introduction

The analysis of nonstationary time series plays an important role in the data understanding of various phenomena such as temperature drift in an experimental setup, global warming in climate data, or varying heart rate in cardiology. Such nonstationarities can be modeled by underlying parameters, referred to as driving forces, that change the dynamics of the system smoothly on a slow time scale or abruptly but rarely, e.g. if the dynamics switches between different discrete states [11].

Often, e.g. in EEG-analysis or in monitoring of complex chemical or electrical power plants, one is particularly interested in revealing the

driving forces themselves from the raw observed time series since they show interesting aspects of the underlying dynamics, for example the switching between different dynamic regimes.

Several methods for detecting and visualizing driving forces have been developed; based on recurrence plots [2], feedforward ANNs with extra input unit [9] or, as Wiskott [11] recently proposed, by Slow Feature Analysis (SFA), a versatile, robust, and fast algorithm. SFA has been originally presented in context of a bioinspired model for unsupervised learning of invariances in the visual system of vertebrates [10] and is described in detail in [11, 12]. SFA works fully unsupervised, just by searching nonlinear combinations of the input signals which vary as slowly as possible in time.

What is "slow" in the driving forces compared to the raw observed time series? Often it might be the case that a driving force contains components on different time scales and it is crucial to understand which time scale will be selected by the driving force algorithm. As an example we consider driving forces made up of two overlayed frequencies $f_1 < f_2$. Will the driving force detection algorithm detect the slower one of the frequencies, f_1, thus being more slow, or the combined driving force made up of f_1 and f_2, thus being more accurate? With this paper we try to deepen our understanding which parameters influence whether the first or the second choice is taken.

2. Slow Feature Analysis

We briefly review here the SFA approach described in [11]. The general objective of SFA is to extract slowly varying features from a quickly varying multidimensional signal. For a scalar output signal and an N-dimensional input signal $\boldsymbol{x} = \boldsymbol{x}(t)$ where t indicates time and $\boldsymbol{x} = [x_1, \ldots, x_N]^T$ is a vector, the question can be formalized as follows: Find the input-output function $g(\boldsymbol{x})$ that generates a scalar output signal

$$y(t) := g(\boldsymbol{x}(t)) \tag{1}$$

with its temporal variation as slowly as possible, measured by the variance of the time derivative:

$$\text{minimize} \quad \Delta(y) = \langle \dot{y}^2 \rangle \tag{2}$$

with $\langle \cdot \rangle$ indicating the temporal mean. Wiskott and Sejnowski [12] propose a closely related slowness indicator η proportional to $\sqrt{\Delta(y)}$. Low η-values indicate slow signals, high η-values fast signals.

To avoid the trivial constant solution, the output signal has to meet the following constraints:

$$\langle y \rangle = 0 \quad \text{(zero mean)}\,, \tag{3}$$
$$\langle y^2 \rangle = 1 \quad \text{(unit variance)}\,. \tag{4}$$

This is an optimization problem of variational calculus and as such difficult to solve. But if we constrain the input-output function to be a linear combination of some fixed and possibly nonlinear basis functions, the problem becomes tractable with the mathematical details given in [11]. A typical choice for the nonlinear basis functions are monomials of degree 2, but other choices, e.g. monomials of higher degree or radial basis functions could be used as well. Basically, SFA searches the eigenvector in the expanded space with the smallest eigenvalue and projects the expanded signal onto this eigenvector to obtain the output signal, which we denote here by y or y_1.

3. Experiments

In the following we present examples with time series $w(t)$ derived from the well-known logistic map [7, 11] to illustrate the properties of SFA. The underlying driving force is always denoted by γ and may vary between -1 and 1 smoothly and considerably slower (as defined by the variance of its time derivative (2)) than the time series $w(t)$. The approach follows closely the work of Wiskott [11] but with more systematic variations in the driving force.

We consider here a driving force that is made up of two frequency components

$$\gamma(t) = \frac{1}{2}(\underbrace{\sin(0.0005\nu_f t)}_{=\gamma_S(t)} + \underbrace{\sin(0.0047\nu_f t)}_{=\gamma_F(t)}) \quad \in \quad [-1,1], \tag{5}$$

where the first component γ_S is roughly ten times slower than γ_F. The question is whether SFA as the driving force detector detects solely the slower component γ_S of the driving force (in an attempt to minimize η) or the full driving force γ (in an attempt to extract the underlying system dynamics as accurately as possible). A second question is whether a phase transition between the two choices might occur as we vary the base frequency ν_f.

In order to visually inspect the agreement between a slow SFA-signal and the driving force γ we must bring the SFA-signal into alignment with γ (since the scale and offset of the slow signal $y(t)$ formed by SFA is fixed by the constraints and the sign is arbitrary). Therefore we define

a γ-*aligned signal*

$$A_\gamma(y(t)) = ay(t) + b \tag{6}$$

where the free parameters a and b are chosen in such a way that the signal $A_\gamma(y(t))$ is in best possible alignment with $\gamma(t)$.

The following simulations are based on 6000 data points each and were done with MATLAB 7.0.1 using the SFA toolkit `sfa-tk` [1].

3.1 Logistic Map in Chaotic and in Predictable Regime

We consider a time series derived from a logistic map

$$w(t+1) = (4.0 - q + 0.1\gamma(t))w(t)(1 - w(t))\,, \tag{7}$$

which maps the interval $[0, 1]$ onto itself and has the shape of an upside-down parabola crossing the abscissa at 0 and 1. The logistic map exhibits an interesting and complex dynamic behaviour, since its parameter $q \in [0.1, 3.9]$ controls different forms of predictabilty: For $q < 0.33$ the map is fully in its chaotic regime (a map with no visible structure, see Fig. 1), for $0.33 < q < 0.53$ we have a mixture of chaotic and predictable periods and for $0.53 < q < 3.9$ it is long-term predictable.

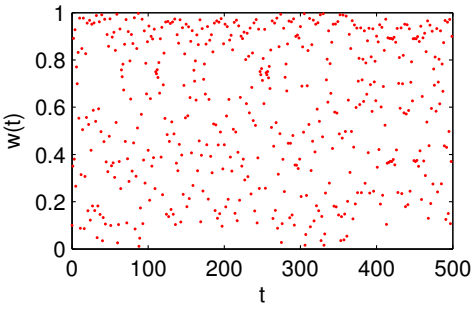


Figure 1. Time series $w(t)$ derived from the logistic map with driving force according to (7) for $\nu_f = 20$ and $q = 0.1$. For all $q < 0.33$, no structure from the driving force is directly visible in the map.

$$\boldsymbol{x}(t) := [w(t - s_\tau),\, w(t - (s_\tau - 1)),\, ...,\, w(t + s_\tau)]^T \tag{8}$$

with delay τ, odd dimension m and $s_\tau := \tau(m-1)/2$. Centering the embedding vectors results in an optimal temporal alignment between estimated and true driving force.

Fig. 2 shows the estimated driving force (from SFA with $m = 19$, $q = 0.1$, $\tau = 1$ and second order monomials) and the true driving force. At the higher frequency $\nu_f = 60$ the estimated driving force is in alignment with the slower component $\gamma_S(t)$. This is remarkable since the slower component is not directly visible in the driving force, only indirectly as envelope of the solid curve. Quite clearly there is a phase transition occuring around $\nu_f = 40$.

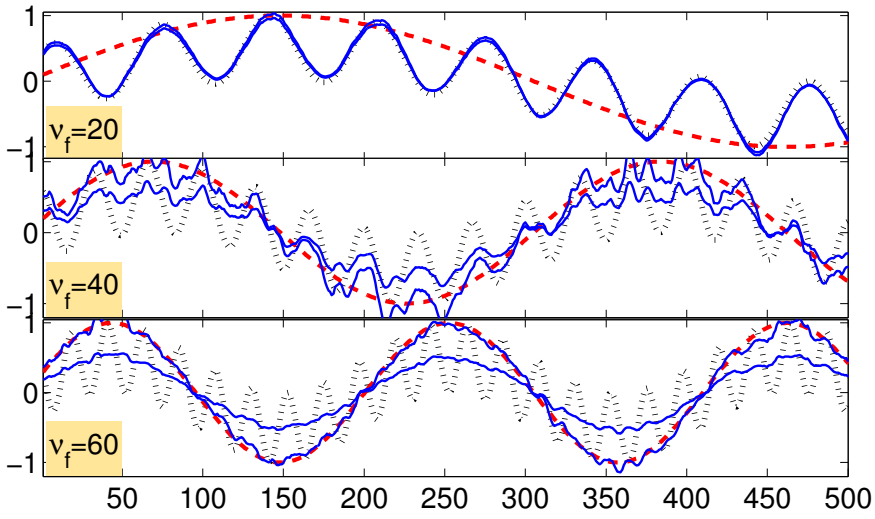


Figure 2. SFA outputs $y_1(t)$ (solid lines) aligned to the driving forces (see (6)) for base frequencies $\nu_f = 20, 40, 60$ clearly show a phase transition from the complete driving force $\gamma(t)$ (dotted line) to its slower subcomponent $\gamma_S(t)$ (dashed line). We see two solid curves since we align the slowest SFA signal once with $\gamma(t)$ and once with $\gamma_S(t)$. For clarity only the first 500 time steps out of 6000 are shown.

In Fig. 3 we vary the base frequency $\nu_f \in [4, 80]$ and we see a swift phase transition. The transition frequency $\nu(P.T.)$ is the crossover point of the two correlation curves, shown in Fig. 3 as black dot. For small $q = 0.1$ (fully chaotic w; left part of Fig. 3) a phase transition occurs at $\nu(P.T.) = 34$ (black dot) while for larger $q = 0.4$ (mix of chaotic and non-chaotic periods in w; right part of Fig. 3) the phase transition happens earlier and occurs swifter at $\nu(P.T.) = 17$.

3.2 The Phase Transition as a Function of q and m

How does the phase transition frequency $\nu(P.T.)$ vary as a function of the predictability q and the embedding dimension m of the SFA-input signal? Both parameters are varied systematically over a broad range and the results are depicted in Fig. 4. First of all it is interesting to note

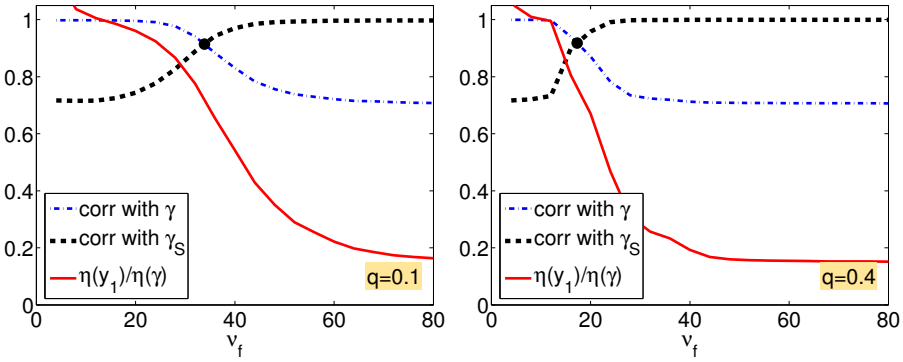


Figure 3. We show the correlation of the SFA-output y_1 with the driving force γ (dash-dotted line) and with its slow component γ_S (thick dashed line). The black dot indicates the phase transition at $\nu(P.T.)$. The slowness quotient $\eta(y_1)/\eta(\gamma)$ (solid line) drops largely near the phase transition. Left: $q = 0.1$, phase transition at $\nu(P.T.) = 34$. Right: $q = 0.4$, phase transition at $\nu(P.T.) = 17$.

that the SFA algorithm, being basically parameter-free, works very well over this broadly varying input material, which makes SFA a robust and versatile algorithm.

A second remark is necessary concerning the SFA implementation `sfa-tk` [1]: While it worked well for small embedding dimensions m, larger m led quite inevitably to numerical instabilities resulting in wrong "slow" signals y_1 which were neither slow nor did they respect the unit variance condition $\langle y^2 \rangle = 1$. We presented in [6] a slightly modified implementation (closer along the lines of [12]) and based on SVD which successfully avoids these numeric instabilities. This modified implementation is used throughout the experiments in this paper.

4. Discussion

It is important for driving force analysis with SFA to understand the mechanisms by which the slowest signal is selected. If the driving force contains two components of different frequencies, two interesting things might happen: If the base frequency ν_f is large enough then SFA will return the slower component as the slowest signal. This is quite remarkable, since SFA detects a signal with a smaller η than the driving force itself. Recall that this slower component is not directly visible in the driving force, only indirectly as the modulation. But after all, it is also quite understandable: If we view the dynamical system as a two-stage process where the slow component γ_S is considered as a modulating force acting on the other (faster) component γ_F with the output of this

stage acting on the dynamical system, then in such a system description, the slower component γ_S becomes directly visible.

Surprisingly, if we lower the base frequency ν_f, we reach the point where the slow component comes "out of sight" and the slowest signal returned by SFA is well-aligned with the driving force itself (slow plus fast component). Why is the slow component alone no longer detected by SFA? We hypothesize that two reasons are responsible for this:

1 If we lower the base frequency ν_f, the fast component γ_F becomes slower and thus contains less information within a given embedding horizon m. This makes the reconstruction of the slow component γ_S more and more noisy. We finally reach the point where for a given embedding dimension m the smoother reconstruction of γ gets a smaller η (becomes slower) than the noisy reconstruction of γ_S. Increasing m should make the reconstruction of γ_S smoother, thus making γ_S again detectable as the slow component.

2 Another reason might be the chaotic nature of the logistic map. In the chaotic region of the map $w(t)$, noise is amplified and makes the reconstruction of the slow component γ_S noisier until it again comes to the point where the noisy reconstruction has a larger η than the (smoother) reconstruction of γ. If this is true, then moving to a better predictable region of the logistic map (increasing q) should make the slow component again detectable.

Both hypotheses are well-supported by the results shown in Fig. 4. On the left-hand side we see the location of the phase transition. For most input signals which are a function of q and ν_f there seems to be a sufficiently large m so that the slow component becomes detectable. For $q = 0.7$ this occurs already at very low frequencies. The curve for $q = 0.6$ (not shown) is for $m > 10$ very similar to $q = 0.7$, which is well-understandable if we recall that all $q > 0.53$ make the time series long-term predictable, thus even a very slow subcomponent becomes detectable. On the right-hand side of Fig. 4 we see that both methods, increasing m or increasing q, finally lead to a reliable detection of the slow subcomponent as it is claimed by our hypotheses.

Hypothesis 1 is also supported by the following experiment: If we lower the frequency of the slow component γ_S but keep the fast component γ_F the same, then SFA will always reliably detect the slow component γ_S, even if only a quarter of its wave length appears in the time series data. This is because the same γ_F allows a reconstruction of γ_S at always the same smoothness level.

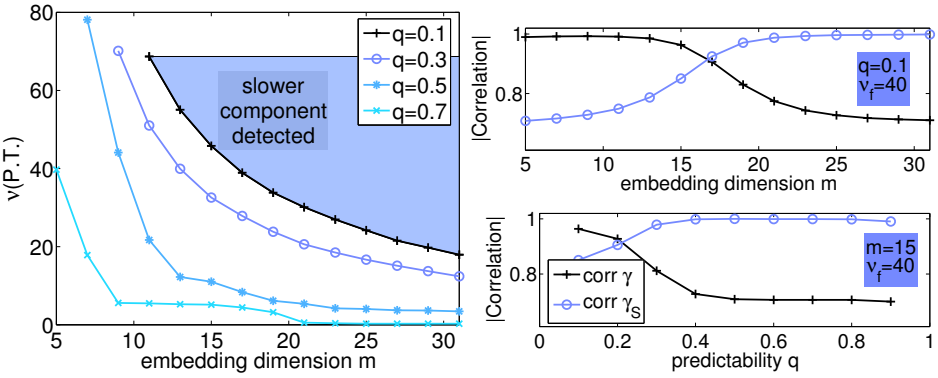


Figure 4. Left: Phase transition frequency $\nu(P.T.)$ as a function of q and m. Right: Absolute values of correlation at fixed $\nu_f = 40$ when varying either m or q.

Nonlinear Regression. Hypotheses 1 and 2 are also supported by the following nonlinear regression experiment: For the set of nonlinear basis functions used by SFA (e.g. monomials of degree 2) and for a given output signal (e.g. γ and γ_S) we seek the best reconstruction in the least-square sense. Decreasing m or q leads to more and more noisy reconstructions of γ_S. We find empirically that quite precisely at the same phase transition points as in Fig. 3 the reconstruction of γ_S gets a higher η (becomes less slow) than the reconstruction of γ. This is remarkable since the slowness principle was not used at all in this nonlinear regression experiment.

Connection to Human Perception. Since SFA has been originally developed as a model for neural information processing [10], it might be natural to ask, whether the observed switch between components and its phase transition has any parallel to human perception and motion coordination. Several phenomena with switching effects are discussed in the literature:

The well-known backward spinning-wheel illusion [8] occurs frequently in movies or under stroboscopic lighting conditions and it shows the transition from a fast forward rotation detection to a slow backward rotation detection. This effect is usually explained by the snapshot-like presentation of the percept which has ambiguous motion interpretations. Somewhat less known is that a similar, although harder to perceive effect can occur under plain sunlight and direct view with the eye [5, 8]. No snapshot-like explanation is possible here, the percept is continous having a greater resemblance to the smoothly varying driving force of our SFA experiments. A possible explanation of the sunlight spinning-

wheel illusion is that rivalry between different motion detectors in the brain occurs [5].

Another well-known phase transition occurs in bimanual motion coordination when performing certain movements with the index fingers of both hands [4]. For the observed phenomena there exists a theoretical model, the Haken-Kelso-Bunz model [3], which describes the phase transition and certain hysteresis effects.

SFA has shown similar capabilities in the sense that the same setup can learn to synchronize with different components of a driving force, depending on the experimental conditions. It remains however to be studied, whether *one* trained SFA system can (without further learning) switch between different components when applied to signals with smoothly varying base frequency and whether a hysteresis effect can be observed.

5. Conclusion

In this paper we have investigated the notion of *slowness* in slow feature analysis (SFA). It has been verified that SFA can reliably detect slow driving forces or their subcomponents over a broad range of parameters in nonstationary time series, even in the presence of chaotic motion.

However it has also been seen that what is perceived as *slow* can vary for driving forces made up of components on different time scales. Depending on the embedding dimensions and the predictability of the underlying dynamical system we observe phase transitions where the slowest SFA-signal moves from alignment to a slow subcomponent to alignment with the (faster varying) complete driving force. Notably, when alignment to the slow subcomponent occurs, SFA is capable of detecting slow signals with an η-indicator considerably lower than the η-value of the true driving force. We found that the slow subcomponent is lost precisely in the moment when its reconstruction in the expanded function space used by SFA has more temporal variation than the reconstruction of the complete driving force.

In real world data it is often not possible to vary the base frequency or the degree of nonlinearity in the observed dynamical system systematically. Therefore, one advice from the present study should be to vary the embedding dimension over a broad range in order to detect possible slow signals which otherwise might be hidden. In any case, SFA has shown to be robustly working on a broad range of input data and it is able to reveal subtle components in the driving forces, thus making it a versatile tool for driving force detection.

Acknowledgments

We are grateful to Laurenz Wiskott for helpful discussions on SFA and to Pietro Berkes for providing the MATLAB code for `sfa-tk` [1]. This work has been supported by the Bundesministerium für Bildung und Forschung (BMBF) under the grant SOMA (AIF FKZ 17N1009) and by the Cologne University of Applied Sciences under research focus grant COSA.

A slightly extended preprint version of this article is available in the arXiv.org e-Print archive at `http://arxiv.org/abs/0911.4397`.

References

[1] P. Berkes. sfa-tk: Slow Feature Analysis Toolkit for MATLAB (v.1.0.1). `http://people.brandeis.edu/\~berkes/software/sfa-tk`, 2003.

[2] M. C. Casdagli. Recurrence plots revisited. *Physica D*, 108(1-2):12–44, 1997.

[3] H. Haken, J. A. S. Kelso, and H. Bunz. A theoretical model of phase transitions in human hand movements. *Biol. Cybern.*, 51(5):347–356, 1985.

[4] J. A. S. Kelso. On the oscillatory basis of movement. *B. Psychonomic Soc.*, 18:63, 1981.

[5] K. Kline, A. Holcombe, and D. Eagleman. Illusory motion reversal is caused by rivalry, not by perceptual snapshots of the visual field. *Vision Res.*, 44(23):2653–2658, 2004.

[6] W. Konen. On the numeric stability of the SFA implementation `sfa-tk`. arXiv.org e-Print archive, `http://arxiv.org/abs/0912.1064`, Dec. 2009.

[7] R. M. May. Simple mathematical models with very complicated dynamics. *Nature*, 261:459–467, 1976.

[8] D. Purves, J. A. Paydarfar, and T. J. Andrews. The wagon-wheel illusion in movies and reality. *P Natal Acad. Sci USA*, 93:3693–3697, 1996.

[9] P. F. Verdes, P. M. Granitto, H. D. Navone, and H. A. Ceccatto. Nonstationary time-series analysis: Accurate reconstruction of driving forces. *Phys. Rev. Lett.*, 87(12):124101, 2001.

[10] L. Wiskott. Learning invariance manifolds. In *Proc. 5th Joint Symposium on Neural Computation*, vol. 8, pages 196–203, 1998.

[11] L. Wiskott. Estimating driving forces of nonstationary time series with slow feature analysis. arXiv.org e-Print archive, `http://arxiv.org/abs/cond-mat/0312317/`, Dec. 2003.

[12] L. Wiskott and T. Sejnowski. Slow feature analysis: Unsupervised learning of invariances. *Neural Comput.*, 14(4):715–770, 2002.

SELF-ORGANIZING COGNITIVE ARCHITECTURE

Oscar Javier Romero López
Fundación Universitaria Konrad Lorenz
Bogotá, Columbia
oscarj.romerol@fukl.edu.co

Abstract Integrating different kinds of micro-theories of cognition in intelligent systems when a huge amount of variables are changing continuously, with increasing complexity, is a very exhaustive and complicated task. Our approach proposes a hybrid cognitive architecture that relies on the integration of emergent and cognitive approaches using evolutionary strategies, in order to combine implicit and explicit knowledge representations necessary to develop cognitive skills. The proposed architecture includes a cognitive level controlled by autopoietic machines and artificial immune systems based on genetic algorithms, giving it a significant degree of plasticity. Furthermore, we propose an attention module which includes an evolutionary programming mechanism in charge of orchestrating the hierarchical relations among specialized behaviors, taking into consideration the global workspace theory for consciousness. Additionally, a co-evolutionary mechanism is proposed to propagate knowledge among cognitive agents on the basis of memetic engineering. As a result, several properties of self-organization and adaptability emerged when the proposed architecture was tested in an animat environment, using a multi-agent platform.

Keywords: Artificial immune systems, Cognitive architectures, Gene expression programming, Memetics, Neural nets

1. Introduction

In the last fifty years, the study of artificial cognitive systems have involved a number of disciplines such as artificial intelligence, cognitive science, psychology and more, in order to determine the necessary, sufficient and optimal conditions and resources for the development of agents exhibiting emergent intelligence. There are several theories of cognition, each taking a different position on the nature of cognition, what a cognitive system should do, and how a cognitive system should be analyzed

and synthesized. From these, it is possible to discern three broad classes: the cognitive approach based on symbolic information processing representational systems; the emergent systems approach embracing connectionist systems, dynamical systems, and enactive systems, all based on a lesser or greater extent of principles of self-organization [1, 2]; and the hybrid approach which combine the best of the emergent systems and cognitive systems [3]. Some of the most relevant cognitive architectures which follow a cognitive approach are: SOAR [4], ACT-R [5], ICARUS [3], and EPIC [3]. Some of the architectures of the emergent approach of major importance are: GW [6], SASE [3], and DARWIN [3]. The hybrid approach architectures are known as CEREBUS [3], KISMET [7], CLARION [8], Polyscheme [9], and LIDA [10]. Some of these architectures deal with aspects of cognitive modeling and representation; some others include learning modules, inference and knowledge generalization; and there are others that try to go further and add motivational and meta-cognition components. The hybrid approach is more complex and of greater interest to us since it seeks to unify the different dichotomies of symbolic vs. sub-symbolic models, explicit vs. implicit learning, and cognitive vs. emergent approaches. However, a common weakness in the hybrid approach architectures is that they usually abridge the system functionality into a rigid structure of symbolic and sub-symbolic components resulting in a poor ability to self-organize and adapt to new environments. The present research focuses on implementing a hybrid architecture for cognitive agents supported by both cognitive and emergent approaches. On the one hand, the cognitive approach provides an explicit knowledge representation through the use of symbolic AI techniques. On the other hand, the emergent approach defines three evolutionary strategies as observed in nature [11]: Epigenesis, Ontogenesis, and Phylogenesis, endowing the architecture with implicit knowledge learning, sub-symbolic representations, and emergent behavior guided by bio-inspired computational intelligence techniques. As the cognitive approach is well known, we will briefly describe it here before elaborating on the emergent approach. The remainder of this paper is organized as follows. The description of the proposed architecture is detailed in Section 2. Sections 3, 4, and 5 describe in more detail each module of the emergent approach according to the three evolutionary strategies. Section 6 outlines and discusses the results of the experiments. The concluding remarks are shown in Section 7.

2. Proposed Hybrid Cognitive Architecture

Figure 1 gives an overview of the hybrid architecture which has six main modules: Attention module, Procedural module, Intentional/Declarative module, Motor module, Motivational module, and Co-evolutionary module. Each module is composed of sub-modules with more specific functionalities which are communicated to each other by broadcasting mechanisms. Architecture is distributed in two dimensions: horizontal and vertical dimensions. At horizontal dimension, modules belong to either emergent or cognitive level, whereas modules at vertical dimension are distributed according to their functionality (attention, procedural reasoning, intentions, motor processing etc.).

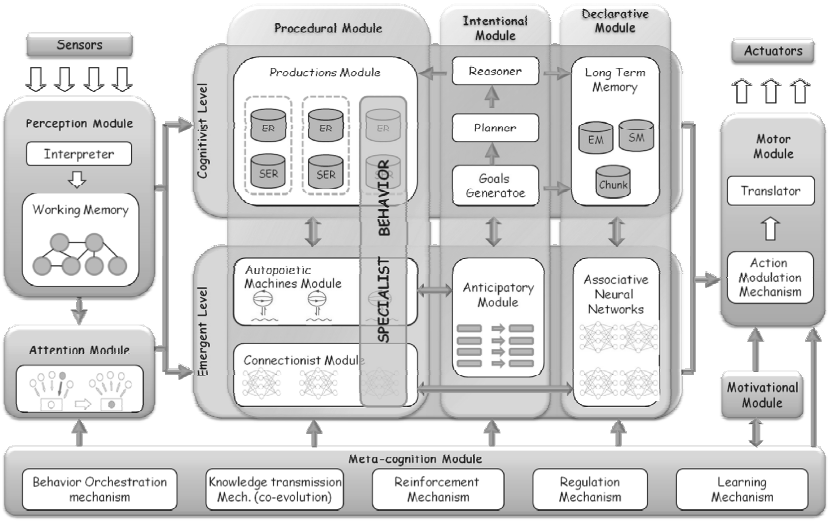


Figure 1. Hybrid Cognitive Architecture.

We will first give a brief description of all the modules of the architecture and then provide a more detailed description of those modules that have been developed so far. Initially, our work has focused on developing the procedural and co-evolutionary modules and their interaction with attention and motor modules. The remainder of the modules will be considered in subsequent stages of the research, and therefore are not described in this work. The Procedural module corresponds to an area of the mammalian brain called Basal Ganglia [5] which is in charge of functions such as rule matching, conflict resolution, cognition, learning, and selection and the execution of actions. This module is composed of several components called Specialist Behaviors (SB), which are organised horizontally, and three sub-modules which

are distributed vertically. The three sub-modules are: the connectionist module, the autopoietic machines module, and the productions module. The horizontally-formed components of the procedural module will be explained in the next section. The Connectionist module (found at the emergent level of the diagram) models innate skills, which require less processing time in comparison with other deliberation processes. It therefore uses the Backpropagation Neural Networks (BNN) which is more appropriate for enacting reactive reasoning. The Autopoietic Machines module is formed by multiple self-organized and self-regulated systems: Artificial Immune Systems [17], where each one models a set of sub-symbolic rules on the basis of autopoietic principles [11]. The Productions module manages different sets of symbolic rules which simulate either the innate knowledge passed on genetically by an evolutionary process, or the knowledge acquired by a previous experience. The Intentional module represents the agent's intentionality through goal and plan generation at the cognitive level, as well as prospection strategies and internal simulation at the emergent level. This module will have a declarative knowledge representation composed of chunks of semantic and episodic memories which are accessed indirectly, as proposed in [8]. This module is able to predict the outcomes of the actions produced by the system and construct a model of events for controlling perception through stimuli anticipation. The Attention module has the responsibility of interpreting the perceived information through the sensors and transforming it into percepts (sensory inputs translated into property/value pairs). The Attention module is based on Global Workspace theory and Theater Metaphor for Consciousness [6]. This module has the responsibility for coordinating the execution of several SBs, which compete and cooperate in order to get the attention focus (consciousness). The most innovative aspect of this module is the behavior orchestration managed by a mechanism that uses Gene Expression Programming (GEP), an evolutionary programming algorithm proposed by Ferreira [12]. This mechanism will be discussed later in Section 4.

3. Epigenic Approach

The epigenesis refers to heritable changes in phenotype (appearance) or gene expression, caused by mechanisms other than changes in the underlying DNA sequence. Therefore, the epigenesis represents the final tuning process by means of each individual adapts efficiently to its environment from the abilities included in its genetic code. In our work, the epigenetic approach references to the mechanisms that allow agent modifying some aspects of its both internal and external structure as a

result of interacting with its environment, in other words, "learning". Therefore, we propose the development of two main approaches which intend to simulate the most evident epigenetic systems observed in nature: the central nervous system (CNS) and the immune system. In our work, the connectionist module which represents the CNS is organized in groups of Backpropagation Neural Networks (BNN), each one representing the sub-symbolic specialization of a task as in [13]. Each specialized BNN is classified according to its purpose, in order to filter the perceived stimuli from environment and select the respective reactive actions. We propose an AIS as autopoietic machine [11] which starts with an sensory input data set (antigens) that stimulate an immune network, and then goes through a dynamic process until it reaches some type of stability. Specifically, each autopoietic machine is based on AiNet [15], a model which implements a discrete immune network that has been developed for data compression and clustering and later for optimization.

3.1 Vertical Integration: Specialist Behaviors

The Specialist Behaviors (SB) are proposed as hybrid units of procedural processing which are in charge of specializing the cognitive skills of the architecture. These specialists are hybrid because of incorporation of both symbolic and sub-symbolic elements at the procedural module. In particular, the procedural module arranges a set of SBs distributed vertically, every one made up of each horizontally-formed component (i.e., an specific SB has one BNN, one AIS, one ER set, and one SER set, as in Fig. 1). Thus, SBs help the architecture to articulate the set of skills because of each SB attends on a specific set of stimuli signals and gives an appropriated response to the environment. Accordingly, each SB can be formalized as follows:

$$\mathrm{SB} = \mathrm{ER} \cup \mathrm{SER} \cup \mathrm{AM} \cup \mathrm{BNN}.$$

The purpose of including multiple components in an SB is that each one compensates the weaknesses of the rest. For example, BNN are often better at making forward inferences about object categories than ERs, whereas ERs are often better at making forward inferences involving causal change than neural networks. AIS is able to make both kind of inferences from implicit representations but it involves more processing time discovering new rules than the other two components. In addition, a reinforcement signal (as a factor of learning in the procedural module) is used to modify the future responses of the agent. This is achieved through adjusting the connections in BNNs, rewarding the activated antibodies in AISs, and extracting sub-symbolic knowledge from emergent level in SERs.

4. Otogenetic Approach

Ontogenetic principles involve all the mechanisms in charge of developing an agent on the basis of the stored information in its own genetic code without interposing the environment influence. Additionally, it defines the history of structural change in a unity without the lost of organization that allows that unity to exist. Some outcomes of these principles as self-replication and self-regulation properties in biological systems can be valued. In our work, the ontogenetic approach is simulated through the interaction among different modules: the Attention module, the Goal module, the Anticipatory Module, and the SBs in Procedural module. The main idea in this approach is that the attention module supported by Global Workspace theory, orchestrates the different SBs in such a way that either cooperate or compete among them. The attention module defines a set of attention machines (AM), which are systems implemented as attention fixations that execute algorithms by sequences of SBs. Each AM has a set of premises that describe the pre-conditions of AM activation, the stream of SBs, and the post-conditions that will have to guarantee after the execution of the stream. Te pre-conditions indicate the goals, expectations, emotions, and stimuli (provided by the working memory) which the agent will have to process and satisfy at any given time. The stream of SBs is a sequence of SBs and relations among them which describes how the agent must behave in a particular situation. Finally, post-conditions are a set of states and new goals generated after the execution of the stream. The main question that addresses the development of the attention module is how it will be able to arbitrate autonomously the execution of SBs in each AM given a set of stimuli, expectations, goals, and emotions? As a result, we propose an evolutionary model based on GEP [12] that is used to evolve the AMs in order to generate an appropriated behavior orchestration without defining a priori the conditions of activation about each SB. GEP uses two sets: a function set and a terminal set. Our proposed function set is: IFMATCH, AND, OR, NOT, INHIBIT, SUPRESS, AGGREGATE, COALITION, and SUBORDINATION. The AND, OR and NOT functions are logic operators used to group or exclude subsets of elements (SBs, goals, working memory items, etc.). The conditional function IFMATCH is an applicability predicate that matches specific stimuli. This function has three arguments; the first argument is the rule's antecedent, an eligibility condition which correspond with a subset of sensory inputs, motivational indicators (internal states, moods, drives, etc.), and working memory elements, which model the agent's current state. All elements of these subsets are connected with logic operators.

If the whole set of conditions exceeds a threshold, then the second argument, the rule's consequent, is executed, otherwise the third argument is executed. Second and third argument should be a set of functions such as INHIBIT, SUPPRESS, AGGREGATE, COALITION, or SUBORDINATION, or maybe an AND/OR function connecting more elements when is necessary. The INHIBIT, SUPPRESS and AGGREGATE functions have two SBs as arguments (SBA, SBB) and indicate that SBA inhibits, suppresses, or aggregates SBB. The COALITION/SUBORDINATION functions, instead of binomial functions mentioned above, perform a set of SBs. The COALITION function describes a cooperation relationship among SBs where actuators may activate multiple actions. The SUBORDINATION function defines a hierarchical composition of SBs which are activated in a specific sequence. In addition to, the terminal set is composed by the SB set, the motivational indicators, the goal set, and the working memory elements. Additionally "do not care" elements are included so whichever SB, motivational indicator, goal, or working memory item can be referenced. Each agent has a multigenic chromosome, that means, each chromosome has a gene set where each gene is an eligibility rule like in the example, so the agent has several rules (genes) as part of its genotype and each one is applied according to the situation that matches the rule antecedent. Each gene becomes to a tree representation and afterwards some genetic operators are applied among genes of the same agent and genes of other agents, as in [12]. Some of these genetic operators are: selection, mutation, root transposition, gene transposition, two-point recombination and gene recombination, in order to evolve chromosomal information. After certain number of evolutionary generations, valid and better adapted AMs are generated. A roulette-wheel method is used to select individuals with most selection probability derived from its own fitness. Fitness represents how good interaction with environment during agent's lifetime was.

5. Phylogenetic Approach

In biology, phylogenesis (evolution) collects all those mechanisms which, leaded by natural selection, have given place to the broad variety of species observed in nature. Evolutionary mechanism operates in populations and as a result, it gets a genetic code which allows individuals of a concrete population to adapt to the environment where they live in. On the basis of phylogenetic theory [11], a co-evolutionary mechanism is proposed to evolve fine-grained units of knowledge through the multiagent system, taking the foundation of meme and memetic algorithms. The term "meme" was introduced and defined by Dawkins [16]], as the

basic unit of cultural transmission or imitation that may be considered to be passed on by non-genetic means. In our work, each meme contains a symbolic and sub-symbolic representation of knowledge, and also a set of indicators such as demotion, reliability, rule support and fitness. As a result, our co-evolutionary mechanism is based on a Memetic Algorithm [16] which is inspired by both Darwinian principles of natural evolution and Dawkins' notion of a meme. This mechanism can be viewed as a population-based hybrid genetic algorithm coupled with an individual learning procedure capable of performing local refinements. Most evolutionary approaches use a single population where evolution is performed; instead of this, in our co-evolutionary approach, the SBs are discriminated in categories and make them evolve in separate pools without any interaction among themselves. After certain period of time a co-evolutionary mechanism is activated. For each behavior pool, a stochastic selection method is executed, where those SBs that had the best performance (fitness) will have more probability to reproduce. Then, a crossover genetic operator is applied among each pair of selected SBs and some memes are both selected and interchanged with other ones.

6. Experimentation

In order to evaluate the proposed cognitive model, following aspects were considered: (1) Analysis of eligibility rules evolved by GEP in the attention module, and (2) Learning convergence of the co-evolutionary mechanism.

An artificial life environment called Animat (animal + robot) is proposed to test the experiments. The environment simulates virtual agents competing for getting food and water, avoiding obstacles, and so forth. Each animat, driven by an agent, disposes a set of 8 proximity sensors around itself. In Fig. 2 is depicted the three environments and the desired paths that the agent should cross. The environment in Fig. 2a was a basic learning scenario where the agent had to follow a simple path. In Figure 2b. some little changes were included, and in Fig. 2c the environment was more complex because of a new path of food and more elements that were introduced. The experiments were made using three agent behaviors: looking for food (SB-eat), avoiding obstacles (SB-avoid-obstacles), and escaping from predators (SB-escaping-from-predators). Thus, some experiments designed to evaluate the performance aspects mentioned above are described next.

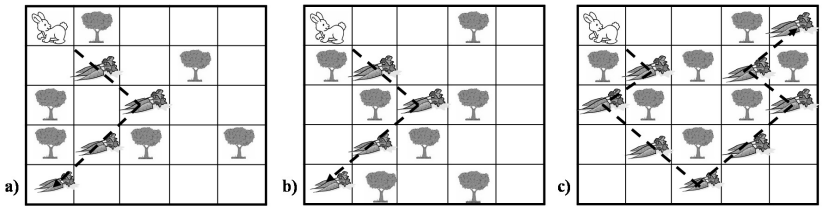


Figure 2. Animat environments: a) basic environment, b) modified environment, and c) complex environment.

6.1 Analysis of Eligibility Rules Evolved by GEP

After the attention machines in the attention module have evolved during a specific number of generations, we analyze the final eligibility rules of the best adapted agents where emergent properties arose. Initially, we present an initial eligibility rule which has syntax conflicts; therefore an evolved eligibility rule syntactically well-formed emerges from GEP. We have chosen an eligibility rule from a non-trained agent and afterwards we show the same evolved eligibility rule but now it has no syntax conflicts and also it's better well-suited than its predecessor.

Eligibility Rule at generation 0:

```
IFMATCH:
   {food},{tree},{empty},{empty},{empty},{empty},
   {empty},{tree} AND {goal-is-eat}
THEN:
    {SB-eat} INHIBITS {SB-avoid-obstacles} AND
    {SB-avoid-obstacles} SUPRESSES {SB-eat}
ELSE:
 SUBORDINATION {SB-avoid-obstacles} AND
    {SB-eat}
```

The above eligibility rule means that when the agent senses "food" around it, it must do something to get the food while is avoiding obstacles, but is contradictory because `{SB-eat}` can't inhibit {`SB-avoid-obstacles`} while `{SB-avoid-obstacles}` is suppressing `{SB-eat}` at the same time. So, the evolved consequent of the eligibility rule after 17 epochs is:

```
IFMATCH:
   {food},{tree},{empty},{empty},{empty},{empty},
   {empty},{tree} AND {goal-is-eat}
THEN
```

```
   COALITION {SB-eat} AND {SB-avoid-obstacles}
ELSE
   {SB-avoid-obstacles} INHIBITS {SB-eat}
```

It is important to notice that evolved eligibility rule does not present any syntax conflict and is a valid rule which forms a coalition among `SB-avoid-obstacles` and `{SB-eat}` behaviors when the agent reads food and obstacles around it. Otherwise, the agent always will execute the rule: `{SB-avoid-obstacles}` inhibits `{SB-eat}`, focusing the agent attention on obstacles because of `{SB-eat}` behavior has a lower priority and is less reactive than `{SB-avoid-obstacles}` behavior.

6.2 Learning Convergence of the Co-Evolution Process

This experiment examines if the fitness of every separate behavior pool increments gradually until it reaches a convergence point while evolution takes place. The experiment was carried out with the parameters on Table 1. Three behavior pools were selected for the experiment: avoiding-obstacles, looking-for-food, and escaping-from-predators. The results are depicted in Fig. 3.

Table 1. Co-evolution learning parameters

Parameter	Value
Epochs	50
Number of epochs per run	50
Crossover probability	0.7
Mutation probability	0.3
Mutation rate η	0.85
Mutation rate θ	0.25
Mutation rate κ	1.03
Mutation rate γ	0.01

Figure 3 depicted some differences in each learning curve, because of environmental conditions, however the pools always tried to converge and reach certain knowledge stability at the same number of epochs (approximately after 30 epochs), that means the evolution has been effective and each behavior pool has established a coherent knowledge base getting a consensus among its own behavior instances and about what the "behavior category" should do.

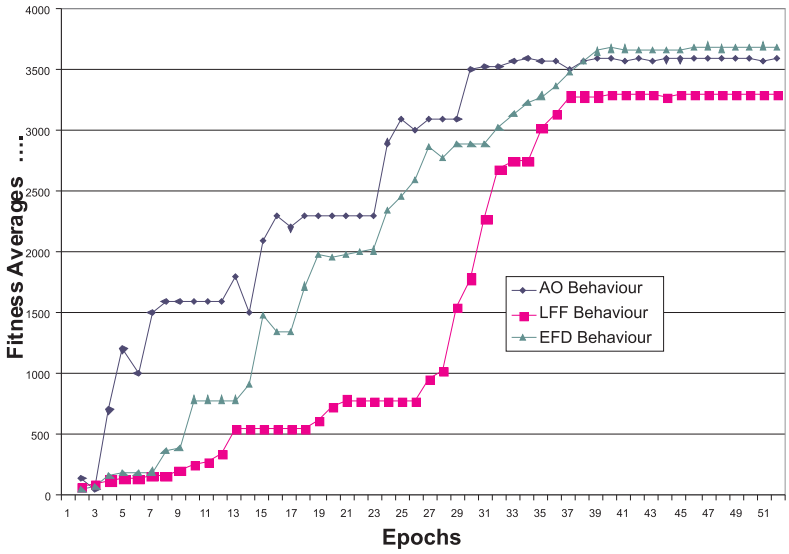


Figure 3. Evolution convergence rate in 3 behaviour pools.

7. Conclusions

The evolutionary mechanisms used in this work, provided a plasticity feature allowing the agent to self-configure its own underlying architecture; thus, it can avoid creating exhaustive and extensive knowledge bases, pre-wired behavior structures of behaviors, and pre-constrained environments. Instead of this, the cognitive agents which use our architecture only need to interact with an arbitrary environment to adapt to it and take decisions in both a reactive and deliberative fashion. In the experimentation, the emergent properties were difficult to discover because it took a lot of time to evolve the overall system despite of using a multi-agent platform with a distributed configuration. Maybe, it can be similar to the natural evolution where adaptation occurs slowly and sometimes produces poor adapted creatures. In our future work we expect to continue working on designing more adaptive and self-configurable architectures, incorporating intentional and meta-cognition modules. One concrete application of this research will be the development of a cognitive module for Emotive Pedagogical Agents where the agent will be able to self-learn of perspectives, believes, desires, intentions, emotions and perceptions about itself and other agents, using the proposed approach which will be responsible of driving the cognitive architecture.

References

[1] M. L. Anderson. Embodied cognition: A field guide. *Artif. Intell.*, 149(1):91–130, 2003.

[2] A. Berthoz. *The Brain's Sense of Movement*. Harvard Univ. Press, Cambridge, 2000.

[3] D. Vernon, G. Metta, and G. Sandini. A survey of artificial cognitive systems: Implications for the autonomous development of mental capabilities in computational agents. *IEEE T. Evol. Compu.*, 11(2):151–179, 2007.

[4] P. Rosenbloom, J. Laird, and A. Newell (Eds). *The Soar Papers*. MIT Press, 1993.

[5] J. R. Anderson, D. Bothell, M. D. Byrne, S. Douglass, C. Lebiere, and Y. Qin. An integrated theory of the mind. *Psy. Rev.*, 111(4):1036-1060, 2004.

[6] M. P. Shanahan and B. Baars. Applying global workspace theory to the frame problem. *Cognition*, 98(2):157–176, 2005.

[7] C. Breazeal. Emotion and sociable humanoid robots. *Int. J. Hum.-Comput. St.*, 59(1):119–155, 2003.

[8] R. Sun, E. Merrill, and T. Peterson. From implicit skills to explicit knowledge: A bottom-up model of skill learning. *Cognitive Sci.*, 25(2):203–244, 2001.

[9] N. L. Cassimatis. Adaptive Algorithmic Hybrids for Human-Level Artificial Intelligence. In *Frontiers in Artificial Intelligence and Applications*, 157:94–110, 2007.

[10] S. Franklin and F. G. Patterson Jr. The LIDA architecture: Adding new modes of learning to an intelligent, autonomous, software agent. In *Proc. Integrated Design and Process Technology*, 2006.

[11] H. R. Maturana and F. J. Varela. *Autopoiesis and Cognition: The Realization of the Living*. Boston Studies on the Philosophy of Science, 1980.

[12] C. Ferreira. Gene expression programming: A new adaptive algorithm for solving problems. *Complex Syst.*, 13(2):87–129, 2001.

[13] D. Rumelhart and J. McClelland. Parallel Distributed Processing: Explorations in the Microstructures of Cognition. MIT Press, 1986.

[14] C. Watkins and P. Dayan. Q-learning. *Mach. Learn.*, 8(3):279–292, 1992.

[15] L. N. de Castro and J. Timmis. *Artificial Immune Systems: A New Computational Intelligence Approach*. Springer, 2002.

[16] R. Dawkins. *The Selfish Gene*. Oxford University Press, 1989

[17] P. Moscato. On Evolution, Search, Optimization, Genetic Algorithms and Martial Arts: Towards Memetic Algorithms. California Institute of Technology, Pasadena, CA, Tech. Rep. Caltech Concurrent Computation Program, Report. 826, 1989.

[18] D. Romero and L. Niño. An immune-based multilayered cognitivemodel for autonomous navigation. In *Proc. Congress on Evalutionary Compution (CEC)*, pages 1115-1122, 2006.

III

APPLICATIONS

MFGA: A GA FOR COMPLEX REAL-WORLD OPTIMIZATION PROBLEMS

Alessandro Turco, Carlos Kavka
Esteco srl
Trieste, Italy
{turco; kavka}@esteco.com

Abstract We present a Multi Objective Genetic Algorithm called Magnifying Front Genetic Algorithm (MFGA) designed in order to treat complex real-world optimization problems. A first source of complexity is the presence of different classes of input variables (real, discrete and categorical). MFGA is able to treat appropriately each of them as well as any combination. Moreover, real-world applications often require a long time to evaluate objective values from input variables. Therefore two issues arises: (i) to control and to tune properly the balance between exploration and exploitation capabilities of the algorithm and (ii) to exploit as much as possible computing resources. We deal with the first issue working on elitism and we introduce parallel steady-state evolution schemes to leave idle as few computational resources as possible. We test the algorithm on a problem arising in Multi-Processor System-on-Chip (MP-SoC) design. This field is characterized by discrete and more often categorical variables. We consider a problem with 18 categorical variables and a search space of $3^5 \cdot 4^3 \cdot 9^{10} \simeq 5 \cdot 10^{13}$ points.

Keywords: Elitism, Genetic algorithms, MP-SoC design, Parallel computing

1. Introduction

The acronym MFGA of this algorithm stands for Magnifying Front Genetic Algorithm since its main purpose is to work iteratively on the non-dominated front obtained so far in three directions: towards (approaching the true Pareto front), laterally (obtaining a wider front) and internally (enhancing the uniformity of the front samples).

With the introduction of elitism, genetic algorithms such as NSGA-II [6] and SPEA2 [11] found a very good answer to the problem of fast convergence. The question we try to answer in this paper is how to

converge better, without slowing down. Elitism indeed is also considered [3, 7] as the main cause of the generation of fronts too concentrated in certain regions leaving holes in others. However, avoiding this kind of strategies, the request for quality cannot even be addressed, since the algorithm would converge too slowly.

The paper presents also an application to a real-world categorical problem with a large design space. We study a mapping problem related to System-on-Chip design [10]. This problem is a severe test for optimization algorithms and it shows the reliability of MFGA: we enhance the results presented in [10] and we manage to perform better than NSGA-II [6] which is widely accepted as the state of the art for general purpose GA. Moreover, we chose this application since any new result in the expanding field of chip design contributes to encourage confidence towards algorithmic optimization, which has been only recently accepted as a necessary tool [2].

The paper is organized in sections. Section 2 contains a description of MFGA with a particular attention to the implementation of elitism (Section 2.1), genetic operators (Section 2.2) and of the steady state evolution (Section 2.3). The mapping problem is described in Section 3, while the results of the optimization are collected in Section 4. Some conclusions (Section 5) and acknowledgments conclude the paper.

2. Algorithm Description

MFGA is designed to solve the classical multi-objective optimization problem, with M objectives and N input variables, in the form:

$$\min(f_1(x_1, \ldots, x_N), \ldots, f_M(x_1, \ldots, x_N)),$$

possibly subject to variable bounds, J inequality constraints and K equality constraints of the type:

$$\begin{aligned} g_j(\bar{x}) &\geq 0 \qquad & j &= 1, 2, \ldots, J; \\ h_k(\bar{x}) &= 0 \qquad & k &= 1, 2, \ldots, K; \\ x_i^L \leq x_i &\leq x_i^U \qquad & i &= 1, 2, \ldots, N. \end{aligned}$$

The input variables vector $\bar{x} = (x_1, \ldots, x_N)$ can contain different kinds of values: the algorithm can deal with continuous, discrete and categorical variables. In the latter case a fictitious integer value is assigned to each choice available in the catalog. MFGA uses binary encoding for discrete variables, while it keeps the continuous form (up to machine precision) for the remaining case.

The backbone of the algorithm is the classical generational genetic scheme. We use different selection methods, but each of them requires

the computed points to be sorted according to the same criterion. We use constraint-dominance sorting [6]. This criterion induces only a partial ordering and the sorting procedure simply divides the points into different fronts. In order to select the best points within a front a second sorting procedure is required: we use crowding distance [6], which helps in maintaining diversity.

2.1 Elitism

Elitism is the key to speed up convergence within a GA [11]. Although crossover and mutation operators can in principle explore the entire design space, their implementation usually generates children not too different (with respect to input variables) from parents with a high probability. For example, one point binary crossover and SBX [5] act in this manner. Hence, if the best points found so far are kept in the parent population, their good properties can be exploited by the children. An elitist procedure performs exactly this task.

The state of the art for GA is NSGA-II [6], which implements a very efficient elitism preservation. The population size is kept fixed and the new parent population is chosen taking every point in the sorted fronts, starting from the best one. When the inclusion of an entire new front would exceed the population size, only a fraction is selected according to the crowding distance computations. This procedure is very robust and it promotes, in a wide range of problems, a fast convergence towards the Pareto front. However, if only a few points manage to reach it, they became a sort of attractors for the subsequent generations since crowding distance will act only on less effective fronts.

Controlled elitism [7] is a routine which sorts points according to the NSGA-II procedure, but then it performs a different selection. The idea is to assign a geometrically decreasing percentage of new parents to each front, choosing the individuals with better crowding distance values. The starting percentage can be arbitrary chosen, the remaining ones are scaled in order to sum up to 100%. If a front contains less points than required, the subsequent front will fill the gaps. The resulting algorithm shows very good convergence properties as well as improved exploration capabilities.

Another criticism to standard elitism is addressed in [3]. Here the problem is once more the quality of the best obtained front, but they consider the opposite situation of an overcrowded front. In this case, the crowding distance sorting may promote exploitation towards fewer directions than existing ones. The proposed solution, called *variable population size* updating, consists in allowing increments (over a fixed

lower bound) of the population size, since all the points in the first front will become parents of the next generation. In this way both accuracy and uniformity of the front are improved.

MFGA combines these two strategies focusing on the ratio between exploration and exploitation phases of the algorithm. A maximum population size L^{max} is identified according to the complexity of the problem and the number of evaluations we are allowed to execute. Every time the elitism operator is invoked, the actual population (parents plus children) is sorted by constraint domination and crowding distance. The choice of the new parents is performed according to the number of points present in the first front, say l:

- If $l < \frac{1}{3}L^{max}$, the new population size will be exactly the integer value nearest to $\frac{1}{3}L^{max}$ and the new parents will be selected according to the controlled elitism operator. This situation is typical of the first stages of the optimization process and it can also occur working with high complexity problems. In both cases a more explorative algorithm can help and controlled elitism guarantees better performances than classical choices.

- If the actual best front is composed by $\frac{1}{3}L^{max} \leq l \leq \frac{2}{3}L^{max}$ points we switch to the variable size routine. All l first-front points (and only them) become parents for the subsequent generation. This happens usually in the middle stage of the run, when the algorithm found some good solutions sufficiently spread along the front. The balance should now move towards exploitation in order to converge faster to the Pareto front and the variable size routine can achieve this result without loosing uniformity and extent of the front.

- Finally, when l exceeds $\frac{2}{3}L^{max}$, it is expected to be in an advanced stage, almost near the solution, so the controlled elitism strategy is restored keeping L^{max} as population size. This should refine the computed front in terms of both convergence and uniformity.

The hypothesized relationship between the number of points in the first sorted set and the converged stage of the optimization run may not always be true. For example, at an early stage the first front can be composed by many far from optimal points, since exploration tends to wider the sampled region. Also the converse may happen, if the Pareto front is very small: the optimization run can be at the end of the convergence history having only few points in the first front.

However, the proposed scheme is able to self repair thanks to the efficiency of both elitism algorithm: they can favor exploration or exploitation, respectively, but they are sufficiently balanced in order to

work well also in difficult situations, as demonstrated in [3, 7]. The reliability of the proposed mixed strategy is analyzed in Sections 4 and 5.

2.2 Operators

Starting from [8], it is a common practice in GA to treat mixed-integer problems using variable-wise operators. MFGA uses one-point crossover and bit-wise mutation for discrete variables. Continuous one are treated with SBX [5] and the classical probability-based mutation [8]. Categorical variables can be treated as discrete ones after having assigned an integer value to each choice allowed by the catalog. However, locality loses its meaning in this case and a larger and fairer exploration can be very useful (see Section 4 for a concrete example of that). MFGA keeps using one-point crossover also for these special variables, but it implements a different mutation operator: the new value is chosen among the possible ones with uniform probability.

The algorithm is designed in order to accept a single mutation probability parameter p_{mut} for both discrete and categorical variables. In the discrete case, each bit of the variable binary representation has a probability p_{mut} to be switched from 0 to 1 or vice versa. Hence the probability to observe a mutation on a single variable, encoded with n bits, is equal to $1 - (1 - p_{mut})^n$. The number l can be computed also for categorical variables, even if they are not encoded in binary form; it will be an indicator of the dimension of the catalog. Knowing this, we assign a mutation probability of $n \cdot p_{mut}$ to each categorical variable. For reasonably small values of p_{mut} this choice provides a wider, but not excessive exploration.

2.3 Steady-State Evolution

The computational cost of a GA algorithm can become relevant on large problems, since sorting is an expensive task. However real-world applications usually are orders of magnitude more demanding. Therefore the efficiency of an algorithm is not measured on its internal routines, but instead considering the usage of the external solver which produces output (objectives and constraints) values from input parameters.

MFGA tries to optimize this effort working in two directions. (i) The improved elitism procedure and the specialized operators reduce the number of points to be sampled in order to converge to a good representation of the true Pareto front. (ii) The proposed steady-state evolution enhances the exploitation of the available computing resources.

Although different kind of solvers may exist, a large part of the applications requires another computer program as solver. Therefore it is natural to speak of queuing systems, grid computing, etc. However, these concepts can be generalized and the proposed algorithm can be applied equivalently. Within a steady-state evolution a new evaluation is launched as soon as an idle solver is found [9]. MFGA implements a moderately-expensive steady-state evolution as well as an expensive, but more efficient one.

In the first case, the same parent population generates new children until the required number of evaluations has been completed. Then the update is performed according to the selected elitism procedure, combining the parent population and all the points evaluated after the previous update (and while some other points are still being evaluated, without leaving any solver idle). In the second case the parents population is updated inserting the new obtained information, every time a solver ends its computation (hence the elitism procedure is applied to the union of the parent population and the new individual). This results in a more expensive algorithm, which however has better convergence properties since new data are exploited as soon as they are obtained.

The dimension of the problem and the availability/capabilities of the solvers should determine the choice between the two possibilities. The problem discussed in Section 3 is borderline, but since we are trying to solve it using as fewer evaluations as possible, we used the second version of the steady-state evolution.

3. A Mapping Problem in MP-SoC Design

We test MFGA against NSGA-II on a large, complex problem involving 18 categorical variables (ten with a catalog of 9 values, three with 4 and five with 3), 3 objectives and many constraints. We choose this kind of problem considering it, from the mathematical point of view, as one of the worst possible cases, in order to prove once more the reliability of GA and to show the capability of the proposed implementation. Moreover, the research field of MP-SoC design met optimization only recently and it still needs to increase confidence. There is the need of more research on optimization algorithms involving discrete and categorical variables, as well as empiric but strong proofs of the capabilities of algorithmic optimization in the field [2].

A detailed description of the chosen problem can be found in [10]. The authors describe a mapping problem for the allocation of work loads within a Multi Processor System-on-Chip, which can be simulated through the Sesame [4] framework. The application replicated is a

Motion-JPEG encoder. The problem is solved using CPLEX [1], NSGA-II [6] and SPEA2 [11] and the results are compared highlighting the good performance of both GA employed. The algorithms are compared using standard evaluation metrics (D-metric, $\Delta-$metric, $\nabla-$metric [10]) and choosing the 18 solution points obtained by CPLEX as reference set.

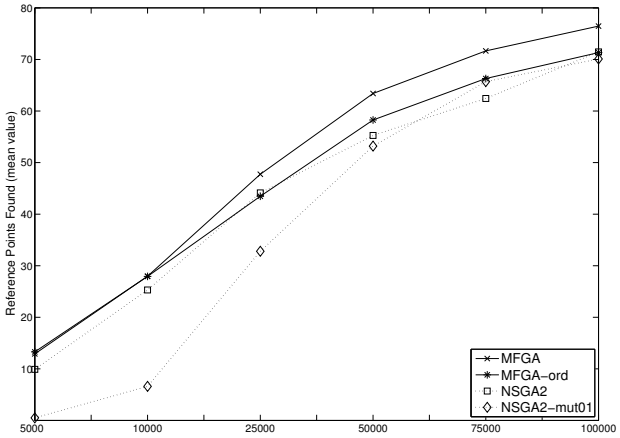


Figure 1. Number of reference points found at given number of evaluations.

We cannot compare directly MFGA with the results proposed in [10] since they do not provide the reference set. They list only two points coming from the set, but our best computed front dominates both. Hence we rerun NSGA-II simulations and compare the results with MFGA considering as reference front the set of non-dominated points among all the computed ones. This front consists of 1688 different points on the design space, but they represent only 88 points on the objective space. We perform the comparison looking only at the objective space, which is more relevant from the applicative point of view.

Another reason for rerunning the test is related to the constraint handling. This mapping problem has a high number of constraints which simply determine if an input configuration is feasible or not. Constraints violations do not exists: a design can either be feasible or its objective values cannot be computed. Therefore in [10] different repair mechanisms are tested in order to transform efficiently unfeasible designs. We want to show that this procedure can be avoided. The fact that our front outperforms the two given points is a first proof together with the higher number of solution points found. Moreover, the evaluation of the constraints is done before entering the (relative) long evaluation process

and therefore we can wait the algorithm to produce directly a feasible point, with the additional advantage to sample a wider region.

We choose NSGA-II, although not explicitly designed for categorical variables, since it is widely accepted as the reference state of the art implementation among GA. The comparison is therefore unfair from this point of view, but performing better than NSGA-II can be considered as the necessary condition for an algorithm to be scientifically interesting.

4. Test Results

We run each algorithm 30 times and we compare only mean values. We consider the already cited evaluation metrics in order to test the accuracy, the uniformity and the extent of the computed fronts. However, since the problem is discrete, the uniformity metric, $\Delta-$metric, loses part of its meaning. A clearer indicator of the quality of a tentative front is the number of reference points found.

NSGA-II requires only a few parameters. The crossover probability was set to 0.7. We followed the standard choice for the mutation probability: one divided by the bit-length of the encoding (i.e. $\simeq$0.02). As a further comparison we run 10 repetitions of NSGA-II using a mutation probability of 0.1. MFGA uses $p_{mut} = 0.05$ and the same crossover probability. We assigned 5 parallel computation to the steady state evolution. The initial population is 100 Sobol points for both algorithm, for MFGA $L^{max} = 100$. The label *MFGA-ord* refers to the results obtained by MFGA considering all the variable as discrete but not categorical.

Table 1 contains the mean values obtained by the four implementation and Fig. 1 highlights the data relative to the number of reference points found. We compute all the values at intermediate steps: 5 000, 10 000, 25 000, 50 000, 75 000 and 100 000 requested points. These numbers consider also unfeasible points as well as possibly repeated requests. The table reports the number of truly evaluated points, which is another interesting indicator of the explorative capabilities of the algorithms.

5. Discussion and Conclusions

We presented MFGA a genetic algorithm designed to deal with complex real-world problems. We cared the balance between exploration and exploitation and the efficiency of the parallel implementation. We tested it on a mapping problem related to MP-SoC design improving existing results.

The test shows the reliability of the proposed algorithm and it demonstrates that categorical variables need a particular attention. Indeed, MFGA performed globally better than NSGA-II, but it makes the dif-

Table 1. Results for the quality evaluation of the computed fronts (mean values)

Step	*Ev. Points*	*# Solutions found*	*D-metric*	$\Delta-$*metric*	$\nabla-$*metric*
NSGA-II	mut$\simeq$0.02				
5000	1681.4	13.0	0.0984	0.0795	1.48e9
10000	2609.9	29.3	0.0749	0.0552	1.69e9
25000	4886.1	44.8	0.0393	0.0489	1.70e9
50000	7885.5	56.6	0.0126	0.0536	2.23e9
75000	10435.0	64.4	0.0036	0.0532	1.96e9
100000	12788.8	72.8	0.0023	0.0499	2.15e9
NSGA-II	mut=0.1				
5000	1800.6	0.5	0.1034	0.124	2.85e9
10000	3141.9	6.6	0.0724	0.0859	2.47e9
25000	6010.9	32.8	0.030	0.0511	2.21e9
50000	9595.4	53.2	0.0068	0.0481	1.62e9
75000	12395.3	65.7	0.004	0.0493	2.03e9
100000	14808.6	70.1	0.0017	0.0478	2.11e9
MFGA					
5000	1744.0	12.9	0.0659	0.0908	1.61e9
10000	27.65.8	27.9	0.0336	0.0701	1.59e9
25000	5222.9	47.7	0.0173	0.0583	2.25e9
50000	8594.1	63.4	0.0076	0.0518	2.18e9
75000	11471.5	71.6	0.0061	0.0509	2.17e9
100000	14007.7	76.5	0.0013	0.0501	2.28e9
MFGA-ord					
5000	1704.0	13.3	0.0861	0.0744	1.43e9
10000	2652.1	27.9	0.0601	0.0606	1.68e9
25000	4990.7	43.4	0.0353	0.0517	1.94e9
50000	8155.2	58.3	0.01	0.0521	2.15e9
75000	10803.1	66.3	0.0045	0.0528	2.08e9
100000	13143.1	71.3	0.0034	0.0514	2.20e9

ference only if the special operators for unordered discrete variables were turned on.

We consider this work a starting point rather than a final landing-stage. Research in chip design needs optimization and only ad hoc implementations can provide the definitive proof of this (it is particularly remarkable in this direction noting that in [10] the 18 points found by a classical algorithm were chosen as reference set, while NSGA-II and MFGA in our tests could find many more). Moreover, a lot of work can

be done on the algorithmic point of view, for example working on the crossover operator or again on elitism.

The algorithm will be tested on new real-world problems in industry for the design of multi-core, low-power processors and advanced platforms for embedded multimedia applications.

Acknowledgments

This research project has been carried on within the European Community FP7 project *MULTICUBE* [2]. We thank Sara Bocchio from STMicroelectronics for introducing us in this applicative research field. We thank also the reviewers for their careful reading of the paper and helpful comments.

References

[1] ILOG CPLEX, `http://ilog.com/products/cplex`.

[2] FP7 research project MULTICUBE, `http://www.multicube.eu`.

[3] T. Aittokoski and K. Miettinen. Efficient evolutionary method to approximate the Pareto optimal set in multiobjective optimization. In *Proc. International Conference on Engineering Optimization (EngOpt)*, 2008.

[4] J. E. Coffland and A. D. Pimentel. A software framework for efficient system-level performance evaluation of embedded systems. In *Proc. ACM Symposium on Applied Computing*, pages 666–671, 2003.

[5] K. Deb and S. Agraval. Simulated binary crossover for continuous search space. *Complex System*, 9(2):115–148, 1995.

[6] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. KanGal Report, 200001, 2000.

[7] K. Deb and T. Goel. Controlled Elitist Non-dominated Sorting Genetic Algorithms for Better Convergence. KanGal Report, 200004, 2001.

[8] K. Deb and M. Goyal. A combined genetic adaptive search (GeneAS) for engineering design. *Comput. Sci. Infor.*, 26(4):30–45, 1996.

[9] J. J. Durillo, A. J. Nebro, F. Luna, and E. Alba. A study of master-slave approaches to parallelize NSGA-II. In *Proc. 22nd IEEE International Symposium on Parallel and Distributed Processing (IPDPS)*, pages 1–8, 2008.

[10] C. Erbas, S. Cerav-Erbas, and A. D. Pimentel. Multiobjective optimization and evolutionary algorithms for the application mapping problem in multiprocessor system-on-chip design. *IEEE T. Evol. Comp.*, 10(3):358–374, 2006.

[11] E. Zitzler, M. Laumanns, and L. Thiele. SPEA2: Improving the strength Pareto evolutionary algorithm for multiobjective optimization. In *Evolutionary Methods for Design, Optimisation and Control with Application to Industrial Problems*, pages 95–100, 2002.

PREFERENCE-BASED MULTI-OBJECTIVE DISTINCT CANDIDATE OPTIMIZATION

Peter Dueholm Justesen
Department of Computer Science
Aarhus University, Denmark
juste@cs.au.dk

Rasmus K. Ursem
Structural and Fluid Mechanics, Grundfos Management A/S
Bjerringbro, Denmark
ursem@cs.au.dk

Abstract Traditional evolutionary multi-objective optimization (EMO) algorithms typically return several hundred non-dominated candidate solutions. From a practical point-of-view, a small set of 5-10 distinct candidates is often preferred because post-processing of several hundred solutions may be too costly, too time-consuming, too difficult to compare design differences, or similar solutions may turn out to be statistically equal in prototyping and manufacturing. Interestingly, these limitations apply to most if not all real-world problems. In this paper we introduce a novel approach to incorporating preferences in order to make an EMO algorithm return a small set of clearly different solutions with respect to performance and design. Here, we distinguish between generalized and domain-specific preferences, where generalized preferences address the aforementioned limitations and the domain-specific preferences cover the wishes of the decision maker. We further suggest the General Cluster-Forming Differential Evolution (GCFDE) algorithm complying to the approach of returning a small, diverse result set. The algorithm is tested on five well-known mechanical engineering problems and a real-world many-objective problem from electrical engineering. On all test problems, GCFDE located distinct optimal solutions, which shows that this is a promising approach for handling preferences in both multi- and many-objective settings.

Keywords: Decision making, Distinct candidates, Diversity management, MCDM, MODCO, Multi-objective optimization, Many-objective optimization

1. Introduction

The application of multi-objective evolutionary algorithms (MOEAs) on a real-world problem typically consists of two steps. First, the optimization step where the problem is set up, the MOEA is run and all non-dominated solutions are gathered. Second, the decision making step where a few solutions to further investigate and perhaps implement are chosen among the non-dominated solutions found in the first step. In this process, the decision maker (DM) has to apply his preferences among the objectives to select the final solutions. Three popular approaches of doing this are:

1 *Aggregating* multiple objectives into one.

2 *Incorporating* decision support systems into the MOEA.

3 *Pruning* the result set of the MOEA.

However, there are several challenges connected to each of these approaches. For algorithms using approach 1, aggregating several objectives into one omits the possibility of exploring trade-offs between objectives by collapsing a population to a single point. For algorithms using approach 2, it can be hard for a DM to express preferences as decision support systems, as well as it is near impossible to distinguish between several hundred non-dominated solutions, which is necessary for a DM using algorithms complying to approach 3. Thus, these authors are inspired by approaches for evaluating non-dominated solutions, see [2].

In this paper, we introduce a novel approach for EMO algorithms addressing the aforementioned problems by directly incorporating different preferences into the algorithm. We deal with the challenges just described in the following ways:

1 Return only a *low number* of solutions.

2 Enable easy incorporation of *domain-specific preference functions*.

3 Ensure *performance and design distinctiveness* of solutions.

The small branch of MO termed Multi-Objective Distinct Candidates Optimization (MODCO) incorporates both *generalized preferences* and *domain-specific preferences* into the algorithm with the goal of finding a small set of 5–10 distinct candidates to make step 2 of the optimization process manageable. See [10] for extensive practical motivation, problem formulation and a survey of related research on the MODCO approach.

In MODCO, the concept *generalized preferences* ensures that the algorithm returns a *low-cardinality result set with distinct solutions*, which is

desirable for most if not all real-world applications. In addition, *domain-specific preferences* covers indicator functions, directing the search toward *preferred areas* of the objective space.

The MODCO parameters K_{NC}, K_{PD} and K_{DD} constitute the generalized preferences, and must be implemented in MODCO algorithms to control result set cardinality and distinctiveness in objective and design space. Setting K_{PD} or K_{DD} to 0 express no demand for distinctiveness, while setting K_{PD} or K_{DD} to 1 express a wish for maximal distinctiveness in either objective or design space, respectively. For inbetween settings, K_{PD} and K_{DD} may express a demand for an intermediate level of distinctiveness.

1 *Number of candidates:* $K_{NC} \in [1 : \infty] \subseteq \mathbb{N}$
How many candidates is it practically and economically feasible to inspect, analyze, and compare in post-processing?

2 *Performance distinctiveness:* $K_{PD} \in [0.0 : 1.0] \subset \mathbb{R}$
How different should the candidates be in performance space?

3 *Design distinctiveness:* $K_{DD} \in [0.0 : 1.0] \subset \mathbb{R}$
How different should the candidates be in design space?

This paper present a novel algorithm complying to the MODCO goal of returning a few distinct candidates for constrained multi-objective problems. The algorithm is tested on five well known multi-objective constrained benchmark problems from mechanical engineering, as well as one many-objective real world problem from electrical engineering.

This paper is structured as follows. Section 2 introduces the General Cluster-Forming Differential Evolution algorithm (GCFDE), which is an extension of our earlier published CFDE [6]. In Section 3, we describe test problems and provide performance comparison. Finally, Section 4 concludes the paper.

2. The General Cluster-Forming Differential Evolution Algorithm

The General Cluster-Forming Differential Evolution (GCFDE) algorithm is based on evolving K_{NC} subpopulations using Differential Evolution, each defining an objective and a design space *centroid*.

The search is based on a primary selection criterion (PSC) and a secondary selection criterion (SSC), which together defines a total ordering of individuals. The primary fitness is based on discrete Pareto-ranking, while the secondary fitness is applied according to diversity. The GCFDE algorithm variants are named as GCFDE/PSC/SSC, with SSC denoting the domain-specific preference based function.

GCFDE extends the first concrete algorithm complying to the goals of MODCO, CFDE [6], which was tested on unconstrained multi-objective problems with two and three objectives. To address the challenges of constrained many-objective optimization GCFDE differs from CFDE on a few accounts. All calculations now takes place in normalized objective and design space to ensure interval-independence. Further, we have:

1 An alternate centroid definition is introduced.

2 Primary selection now incorporates constraint handling.

3 Secondary selection now also handles design distinctiveness.

Algorithm 1 lists the pseudocode of the GCFDE algorithm, and nomenclature is found in Table 1. In pseudocode, $minObjDist(CO_i)$ denotes the function returning the minimum Euclidean distance from centroid CO_i to the nearest other centroid in normalized objective space, and $minDesDist(CD_i)$ denotes the corresponding function for normalized design space.

Table 1. Nomenclature for GCFDE

Symbol	**Meaning**	**Symbol**	**Meaning**
P	Population	P_i	Subpopulation i
CO_i	Objective space centroid of P_i	CD_i	Design space centroid of P_i
$f(x)$	Objective vector of x	$d(x)$	Design vector of x
x	Individual	$x_{i,j}$	x at the j'th index in P_i

2.1 Centroid Definition

To enhance convergence in a many-objective setting, an average of the non-dominated individuals with the best secondary fitness in P_i now defines the centroids both wrt. objectives and design parameters. Thus, when solving many-objective problems, we use the single individual with the highest secondary fitness in P_i to define centroids CO_i and CD_i, by setting $CO_i = f(x_{i,1})$ and $CD_i = d(x_{i,1})$ after truncation in the main loop. For multi-objective problems, we use the average placement of all N/K_{NC} individuals in each P_i as objective and design centroids, see [6].

2.2 Primary Selection Criteria

The primary selection criteria (PSC) assigns a rank to the individuals and it thereby determines the individuals to place in the highest ranked front, i.e., those individuals that are selected wrt. the SSC.

In this paper, we use the global constraint-domination **(GCD)** relation from GDE3 [7], which states that an individual x constraint-dominates individual y iff:

- x is feasible and y is not.
- x and y are infeasible and x dominates y in constraint space.
- x and y are feasible and x dominates y in objective space.

2.3 Secondary Selection Criteria

The secondary selection criteria (SSC) determines which individuals are chosen from the highest ranked front to be included in the next generation. In GCFDE, the SSC firsts ensures performance distinctiveness, and then design distinctiveness. However, if both performance and design distinctiveness is achieved, GCFDE performs domain-specific preference based search. This allows concurrent application of both generalized and domain-specific preferences with MODCO parameters controlling the balance between the two. It is important to realize that the preference based search may be defined by *any* domain-specific preference criterion without violating the desire for returning distinct candidates.

As seen in Algorithm 1, the secondary fitness assignment may differ from subpopulation to subpopulation. Subpopulations that are partly overlapping in objective space selects next generation based on performance distinctiveness, while subpopulations not violating neither performance nor design distinctiveness performs preference based search.

GCFDE uses the secondary fitness assignment to fulfill distinctiveness requirements using the centroid distance measure introduced in [6]:

$$SF(x) = min(\{dist(f(x), CO_j), j = 1..K_{NC}, j \neq i\}) \quad (1)$$

That is, we assign the minimal Euclidean distance in normalized objective space to another centroid to each individual in P_i. As this measure is to be maximized, individuals close to another subpopulation centroid will be penalized, guiding subpopulations away from each other if they are too close wrt. K_{PD}. This also enhances clustering as individuals far away from their own centroid are more likely to be penalized.

If performance distinctiveness is achieved, but design distinctiveness is not, the procedure is performed in design space, assigning secondary

Algorithm 1 General Cluster-Forming Differential Evolution

Require: Population size N, K_{NC}, K_{PD}, K_{DD}
Ensure: K_{NC} distinct, feasible, non-dominated individuals.
1: Initialize K_{NC} subpopulations with N/K_{NC} random individuals.
2: Assign to all individuals their rank as primary fitness.
3: Assign to all individuals $SF(x)$ given by preference based function.
4: **while** Halting criterion has not been met **do**
5: Calculate all subpopulation centroids CO_i, CD_i
6: Perform global $DE/rand/1/bin$ mating with replacement - store incomparable offspring.
7: Migrate incomparable offspring to nearest subpopulation wrt. Euclidean distance to objective space centroids.
8: Assign to all individuals their rank as primary fitness.
9: **for** All $P_i \in P$ **do**
10: **if** $minObjDist(CO_i) < K_{PD}/K_{NC}$ **then**
11: $\forall\, x_{i,j} \in P_i$ assign $SF(x)$ according to Equation 1.
12: **else if** $minDesDist(CD_i) < K_{DD}/K_{NC}$ **then**
13: $\forall\, x_{i,j} \in P_i$ assign $SF(x)$ according to Equation 2.
14: **else**
15: $\forall\, x_{i,j} \in P_i$ assign $SF(x)$ given by preference based function.
16: **end if**
17: **end for**
18: Truncate subpopulations to a size of N/K_{NC} by sorting wrt. rank first, then secondary fitness.
19: **end while**
20: Return K_{NC} distinct solutions, by making a final sorting of each P_i wrt. rank, then preference based secondary fitness, and returning $x_{i,1}$ for $i = 1...K_{NC}$.

fitness to each $x \in P_i$ according to Equation 2. Along with ensuring design distinctiveness, K_{DD} also controls in which extent the DM wishes a 1:1 correspondence between result set design and performance, by guiding individuals towards clusters in both objective and design space, if both K_{PD} and K_{DD} are high. Note, that these divergent SSCs are analogous, working in objective space and decision space, respectively.

$$SF(x) = min(\{dist(d(x), CD_j), j = 1..K_{NC}, j \neq i\}) \tag{2}$$

If both performance and design distinctiveness wrt. K_{PD} and K_{DD} is achieved, GCFDE performs preference based search. In this paper, we experiment with two well known domain-specific preference functions. The first such is the *weighted sum* (WS) function, aggregating multiple

objectives into one fitness, by summing the weighted contribution of each using a weight vector supplied by the DM. The WS function guides the search towards the areas of objective space which are the most biased wrt. the weight vector supplied by the DM.

Another preference based utility function is the *knee utility (KNEE)* function proposed by Branke et al. [1], which is designed to discover knee regions by calculating an average fitness value for a large number of randomly sampled weight vectors, 100 vectors in our experiments. If the average fitness is good, the individual is more likely to reside in a knee-region. The knee utility function is generally applicable, and guide the search towards regions with good trade-offs between objectives.

3. Experiments and Results

In our introducing article [6], the CFDE algorithm was tested on a large suite of standard benchmark problems, including ZDT [3], DTLZ [5] and knee problems [1]. On these problems, CFDE showed superior convergence compared to DEMO versions [8]. Further, parameter usage, diversity of results, and knee search were demonstrated.

In this paper, we compare GCFDE with GDE3 on a set of well-known benchmark problems from mechanical engineering and on a many objective real world problem from electrical engineering. This problem models a part of the control circuit for the Grundfos Alpha Pro pump, which is a small circulation pump for heating in private houses. This circuit is also used in the Alpha2 pump, see more at `www.grundfos.com/alpha2`.

3.1 Mechanical Engineering Problems

To investigate the *convergence capability* of GCFDE on constrained multi-objective problems, we use the Two Member Truss Design (TMTD), the Gear Train Design (GTD), the Multiple Disk Clutch Design (MDCD), the Spring Design (SD) and the Welded Beam Design (WBD) problem from [4]. These are all constrained and bi-objective.

To compare convergence performance, we investigate to which extent the GCFDE/GCD/KNEE result sets dominate the most similar solutions from the returned populations of GDE3. 20 runs have been performed for both GDE3 and for GCFDE on each test problem. For each generated result set of GCFDE, we compare each of the K_{NC} GCFDE individuals to their most similar counterpart from each of the GDE3 populations, i.e. the GDE3 individual being closest in normalized Euclidean objective space. This yields $K_{NC} \cdot 20 \cdot 20$ comparisons per problem, e.g. 2000 for $K_{NC} = 5$. This gives a percentage of the amount of dominating,

dominated, incomparable and equal individuals produced by GCFDE, with relations defined as in [3]. Results are shown in Table 2.

For all problems and on both algorithms, we used a population size $N = 100$ along with DE parameters $F = 0.5$ and $CF = 0.3$. On all problems except the GTD problem, we have performed only 100 generations of both algorithms, whereas for the GTD problem 200 generations were performed to ensure convergence. For the GCFDE algorithm, we have used $K_{NC} = 5$, $K_{PD} = 0.5$ and $K_{DD} = 0.0$ on all runs.

Table 2. GCFDE/GCD/KNEE versus GDE3.

GCFDE vs. GDE3	TMTD	GTD	MDCD	SD	WBD
Dominates (%)	27.5	0.0	2.0	55.5	14.5
Variance (%)	± 5.0	± 0.0	± 5.0	± 4.8	± 8.3
Dominated (%)	3.0	5.7	0.0	32.4	18.0
Variance (%)	± 2.1	± 5.5	± 0.0	± 6.4	± 3.3
Incomparable (%)	69.5	1.3	8.4	12.1	67.5
Variance (%)	± 4.4	± 3.8	± 5.4	± 4.4	± 8.0
Equal (%)	0.0	93.0	91.4	0.0	0.0
Variance (%)	± 0.0	± 8.0	± 5.5	± 0.0	± 0.0

As seen in Table 2 the GCFDE algorithm outperforms the GDE3 algorithm on the TMTD and SD problems with the highest percentage of solutions dominating the GDE3 counterparts. Due to the continous objectives of these problems, there is also a high percentage of incomparable solutions on the TMTD problem.

For the problems with some discrete objectives, GTD and MDCD, the two algorithms find roughly identical solutions, and only few are incomparable. Here, the two algorithms seems to have equal performance. Taking variance into account, this also goes for the WBD problem with a similar percentage dominating/dominated solutions produced by GCFDE, along with a high percentage of incomparable solutions.

Overall, the GCFDE algorithm appears to perform equal to or better than the GDE3 algorithm on the problem set, giving confidence in the ability of GFCDE to converge to the true Pareto-front of constrained multi-objective problems.

To illustrate the *diversity* of the result sets of GCFDE wrt. K_{PD}, the GCFDE algorithm was run with different K_{PD} settings on the TMTD problem, see Fig. 1. As expected, the distance between candidates decreases when K_{PD} is lowered, while extreme solutions are found and

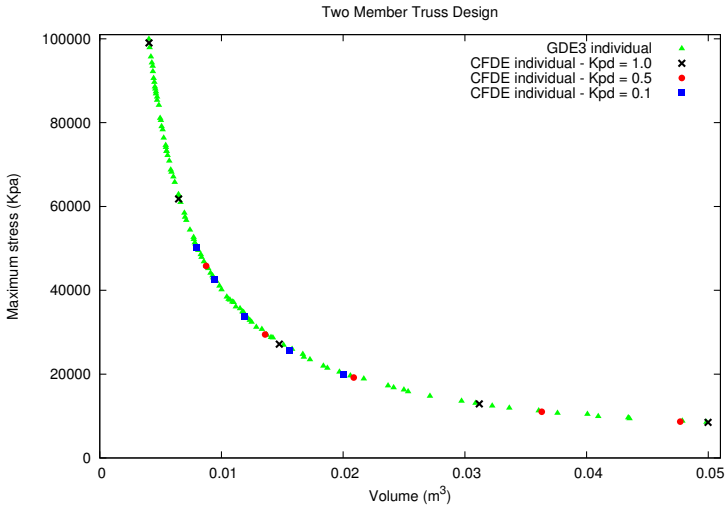


Figure 1. GDE3 and GCFDE/GCD/KNEE result sets on the TMTD problem.

maintained when K_{PD} is maximal, confirming our earlier observations from thorough diversity tests on artificial benchmark problems, see [6].

3.2 Circuit Design for the Alpha Pro Pump

The objective in the circuit design problem is to find component values for a number of resistors and capacitors resulting in a circuit matching the desired functionality. The subcircuit being optimized is a low-pass filter with DC rescaling functionality. It may be possible to address this problem analytically. However, this will most likely produce a solution having component values not available in the standard rows for resistors and capacitors. Rounding such an analytical solution to standard row values will typically decrease circuit performance making this approach less attractive. Instead, optimization may directly find the component values from the standard rows.

The circuit is illustrated in Fig. 2. The components R1, R3, R4, R5, R6, R7, R8, R9, C1, and C6 are subject to optimization. In this, the available resistor and capacitor values are limited to those in the Grundfos stock, i.e., components used in other Grundfos products.

The part of the circuit subject to optimization has both AC and DC functionality. The AC-functionality of the sub-circuit is to provide low-pass filtering of Idc=0.3A with at least -40dB dampening at 125Hz. The DC-functionalities of the circuit are to attenuate V_{DC} as much as possible

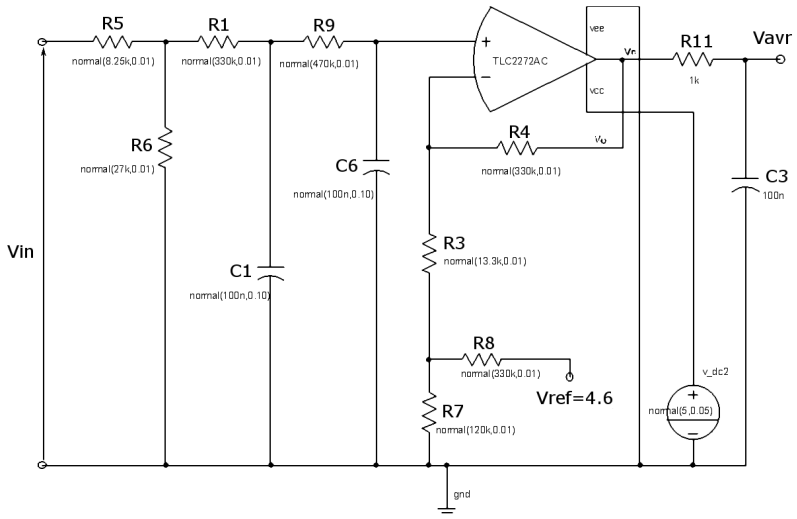


Figure 2. Circuit layout for the low-pass anti-aliasing filter.

and to amplify simultaneously the DC-component, i.e., the average value of the signal as much as possible.

It is out of the scope of this paper to describe the problem in full details. However, a complete description with figures is available in a technical report [9]. The circuit is simulated using the Saber simulator from Synopsys. In the following, the $M(f_i)$ is the Saber magnitude function (in dB) at frequency f_i. The optimization problem has five objectives and five constraints, which are mainly for ensuring that valid component values are selected.

The first objective $F1$ captures the deviation from the desired AC functionality on three frequencies, $f_i = (0.01, 2.0, 5.0)$.

$$F1 = \sum_{f_i} \begin{cases} 20\log\left(M(f_i)\right) - v_{i,max} & 20\log\left(M(f_i)\right) > v_{i,max} \\ v_{i,min} - 20\log\left(M(f_i)\right) & 20\log\left(M(f_i)\right) < v_{i,min} \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

The second objective $F2$ models the dampening at 125Hz and it is achieved by minimizing the slope around 40–50Hz.

$$F2 = \frac{20(\log(M(50Hz)) - \log(M(40Hz)))}{50Hz - 40Hz} \tag{4}$$

Third, fourth and fifth objectives models the DC-functionality.

$$F3 = \left| 1.0 - \frac{\mathrm{R6}}{\mathrm{R5}} \cdot \frac{\mathrm{R_{TOT}}}{\mathrm{R4}} \right| \quad \mathrm{R_{TOT}} = \mathrm{R3} + \frac{\mathrm{R7} \cdot \mathrm{R8}}{\mathrm{R7} + \mathrm{R8}} \tag{5}$$

$$F4 = |4.6 - \Delta V1 - G \cdot 2.603| \quad G = \frac{1.0 + \mathrm{R4}/\mathrm{R_{TOT}}}{1.0 + \mathrm{R5}/\mathrm{R6}} \tag{6}$$

$$F5 = |0.15 + \Delta V2 - G \cdot 1.301| \tag{7}$$

The variables $\Delta V1 = 0.12$ and $\Delta V2 = 0.10$ in the above equations provide the necessary six-sigma margins for component variations.

The a priori analysis of the circuit design problem resulted in the following generalized preference values:

- $K_{NC} = 4$ – the project manager told us how many prototypes could be built.
- $K_{PD} = 0.0$ – the domain expert (electrical engineer) told us that all objectives could obtain a very low value simultaneously, but this was close to impossible with traditional design methods.
- $K_{DD} = 0.5$ – the domain expert told us that similar performance could be obtained with highly different solutions.

To compare the *convergence capability* of GCFDE and GDE3 on this problem, we use the same procedure as for the benchmark problems. However, the original GDE3 fails due to the absent selection pressure given only by Pareto-classification and a divergent secondary fitness. To enable comparison, we have exchanged the Crowding-distance measure of NSGA-II [3] used as secondary fitness in GDE3 with the convergent, preference based secondary fitness functions presented in Section 2.3.

In simulation, we used population size $N = 200$, $F = 0.35$, $CF = 0.2$, and the number of generations was 2000 for both algorithms. In scaling, $[0.0, -1.0, 0.0, 0.0, 0.0]$ was used as best point and $[1.0, 0.0, 1.0, 1.0, 1.0]$ was nadir point. $[1.0, 1.0, 3.0, 1.0, 1.0]$ was used as weight vector in WS.

In Table 3, we present a comparison between GCFDE variants and GDE3 variants, both using GCD as PSC. In the top row we denote the GCFDE SCC in the top and the GDE3 SSC below, while the left-most column denotes relations. As no equal solutions were produced in this many-objective setting, this relation is omitted. As simulating the circuit in question takes much more time than calculating the fitnesses of benchmark problems, only 10 runs were performed for the two algorithms, however on two different versions of both. With $K_{NC} = 4$, this

Table 3. GCFDE versus GDE3 on the Circuit Design for the Alpha Pro problem.

GCFDE GDE3	KNEE KNEE	KNEE WS	WS KNEE	WS WS
Dominates (%) Variance (%)	22.75 ± 20.64	17.25 ± 17.92	23.25 ± 20.08	19.25 ± 18.65
Dominated (%) Variance (%)	0.50 ± 3.50	0.50 ± 3.50	0.25 ± 2.49	0.50 ± 3.50
Incomparable (%) Variance (%)	76.75 ± 20.69	82.25 ± 17.78	76.50 ± 20.25	80.25 ± 18.47

amounts to 400 comparisons per column in Table 3. The high variance in this is due to the low cardinality of the GCFDE result sets.

Table 3 demonstrates that even using the same preference based utility functions as guide, the GCFDE algorithm clearly outperforms the GDE3 algorithm on this many-objective problem, with around 20 % GCFDE solutions dominating GDE3 solutions and below 1 % of GDE3 solutions dominating GCFDE solutions, for all algorithm variants. Thus, GCFDE performs well independently of the convergent SSC used.

The main reasons for this performance are most likely the more focused search using subpopulations with migration, as well as the application of both divergent and convergent SSCs in the GCFDE algorithm. The GDE3 algorithm using GCD ranking and a divergent SSC fails in a many-objective setting, because it relies to heavily on Pareto-classification, whereas it seems to converge prematurely using GCD ranking and a convergent SSC due to lack of solution diversity.

Overall, this shows that the approach of returning few, distinct solutions from the true Pareto-front of a many-objective problem is more feasible than trying to cover the full front, but also that maintaining diversity is necessary in achieving optimality. Thus, both diversity and optimality should be considered in many-objective optimization.

4. Conclusions

In this paper, we have introduced the Cluster-Forming Differential Evolution (GCFDE) algorithm, which is able to handle both multi- and many-objective constrained problems. Convergence was demonstrated on five well known constrained problems from mechanical engineering and a real-world many-objective problem from electrical engineering. GCFDE was found to perform equal or slightly better than GDE3 on

multi-objective problems, whereas for the many-objective problem, the GCFDE algorithm clearly outperformed the GDE3 algorithm.

In summary, the solutions produced by GCFDE are both distinct and more than competitive to solutions found with GDE3, even when using the same preference functions to guide the optimization, which shows the ability of GCFDE to converge to a low number of distinct optimal solutions – even when solving a many-objective problem. This further shows that diversity and optimality can be maintained in synergy.

References

[1] J. Branke, K. Deb, H. Dierolf, and M. Osswald. Finding knees in multi-objective optimization. *Lect. notes Comput. Sc.*, 3242:720–729, 2004.

[2] D. W. Corne and J. D. Knowles. Techniques for highly multiobjective optimisation: Some nondominated points are better than others. In *Proc. Conference on Genetic and Evolutionary Computation (GECCO)*, Vol. 1, pages 773–780, 2007.

[3] K. Deb. *Multi-Objective Optimization using Evolutionary Algorithms*. John Wiley & Sons, Chichester, 2001.

[4] K. Deb and A. Srinivasan. Innovization: innovating design principles through optimization. In *Proc. Conference on Genetic and evolutionary computation (GECCO)*, pages 1629–1636, 2006.

[5] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler. Scalable multi-objective optimization test problems. In *Proc. Congress on Evolutionary Computation (CEC)*, vol. 1, pages 825–830, 2002.

[6] P. D. Justesen and R. K. Ursem. Multiobjective distinct candidates optimization (modco): A cluster-forming differential evolution algorithm. *Lect. Notes Comput. Sc.*, 5467:525–539, 2009.

[7] S. Kukkonen and J. Lampinen. GDE3: the third evolution step of generalized differential evolution. In *Proc. IEEE Congress on Evolutionary Computation (CEC)*, Vol. 1, pages 443–450, 2005.

[8] T. Robič and B. Filipič. DEMO: Differential Evolution for Multiobjective Optimization. *Lect. Notes Comput. Sc.*, 3410:520–533, 2005.

[9] R. K. Ursem. A many-objective circuit design problem: The anti-alliasing filter. Grundfos technical report, 2009-01. Download: `www.daimi.au.dk/~ursem/publications/TR-09-01.pdf`.

[10] R. K. Ursem, P. D. Justesen. The multi-objective distinct candidates optimization approach. In *Proc. Fourth International Conference on Bioinspired Optimization Methods and their Applications (BIOMA)*, pages 55–66, 2010.

APPLICATIONS OF EVOLUTIONARY ALGORITHMS IN CHEMICAL ENGINEERING: A CASE STUDY ON FITTING SOVOVA'S MASS TRANSFER MODEL

Dejan Hrnčič, Marjan Mernik
Faculty of Electrical Engineering and Computer Science
University of Maribor, Slovenia
{dejan.hrncic; marjan.mernik}@uni-mb.si

Maša Knez Hrnčič, Željko Knez
Faculty of Chemistry and Chemical Engineering
University of Maribor, Slovenia
{masa.knez; zeljko.knez}@uni-mb.si

Abstract In chemical engineering reliable models are necessary to reduce the cost of process design. An evolutionary algorithm with resizable population was used to estimate coefficients of Sovova's mass transfer model and was compared with a global optimizer found in the literature. Comparison of the evolutionary algorithm to the global optimization technique proved that the evolutionary algorithm is more robust, efficient, and significantly better than the global optimizer in regards to the deviation of the model from experimental data.

Keywords: Evolutionary algorithms, Mass transfer, Parameter estimation, Sovova model

1. Introduction

Evolutionary Algorithms (EAs) [6] are useful for solving realistic chemical engineering problems because they are robust, give optimal or semi-optimal solutions, and can easily be adapted for different problems. They do not require any additional information about objective functions such as differentiability or continuity. In other words, they

do not make any assumptions about the underlying fitness landscape. Therefore, they are often used to solve hard, real world problems where other heuristic algorithms fail.

Several applications of EAs in chemical processes for parameter estimation can be found in literature. Authors in [7] estimated parameters of heterogeneous catalytic reaction with the combination of an EA and Levenberg-Marquardt method. In [1], an EA, with a local convergence method, was used to simultaneously estimate the kinetic and energetic parameters on a real and complex chemical system. The estimation of kinetic parameters of multi-component photocatalytic process was done in [9], also with the combination of an EA and Nelder-Mead simplex local optimization. All mentioned approaches use the combination of EAs with a local search technique. To the best of our knowledge, there are not any papers that use EAs in the field of extraction models.

In this paper, parameter estimation of Sovova's mass transfer model [8] with EA is presented. This model is used to describe extraction kinetics curve of vegetable oils and it is a commonly used model found in literature. The model parameters are mostly fitted with the use of local optimization tools. However, in [5] the parameter fitting was also studied using global optimization tool, that uses lexicographical grid method (LGM) with local variation algorithm (LVA) to obtain the near global optimum point on which the Nelder-Mead method was used to find the global optimum. This global optimization algorithm was compared with EA, presented in this paper, and the results are shown in Section 5.

In the following section, the supercritical fluid extraction (SFE) is briefly explained. Section 3 presents the Sovova's mass transfer model and Section 4 presents the EA for fitting the model to experimental data. The results and comparison between proposed EA and global optimization tool are presented in Section 5. Key findings and concluding remarks are summarized in Section 6.

2. Supercritical Fluid Extraction

The first industrial processes in chemical engineering for production of various substances were operated within a human living environment, at atmospheric conditions. But demands for new products, with special properties, in 20th century introduced new techniques in industrial processes, especially for obtaining high value products. Some of those techniques apply high pressure.

High pressure is used in different processes such as extractions, impregnation of wood and textile, small particle production in pharmacy, as solvent for chemical and biochemical reactions, etc [2].

One of the important processes that use high pressure are supercritical fluid extractions (SFEs). They are applied in food and flavoring industry (e.g., decaffeination of tea and coffee) and as a residual solvent removal. They are also used in petrochemical industry, pharmaceutical industry and lately also in environment protection processes for elimination of residual solvents from wastes and for purification of contaminated soil and water.

In SFE the feed material is contacted with a supercritical fluid (SCF) as solvent. SCFs are solvents at or above critical temperature and/or pressure. They are used because of their environmental, health, safety and chemical benefits [2]. As environmental benefits, the conventional organic solvents, which are replaced with SCFs, are environmentally far more damaging than most of the SCFs. Most important SCFs (SC CO_2 and SC H_2O) are non-cancerogenic, non-toxic, non-mutagenic, non-flammable and thermodynamically stable. This properties assure the health and safety benefits. At last one of the major process benefits is derived from the thermophysical properties of SCFs (high diffusivity, low viscosity, high density and varying dielectric constant), which can be fine-tuned by changes of operating pressure and/or temperature. The material extracted in SFE can be either in liquid or solid state.

This work takes a closer look into extractions solid-SCF, where the soluble components are separated from the solid material with SCF as a solute. For design of apparatus, for SFE, the data from phase equilibrium, energy balances and mass transfer is required. From the phase equilibrium data, the solubility of compounds in SCF at different process conditions (pressure and temperature) is obtained, therefore the operating conditions, the type and quantity of the solvent can be determined. Energy balances are the basis for determining the energy consumption of the process. From mass transfer data, which has an enormous influence on the economy of extraction process, the time used for high pressure process run is determined.

The flow sheet of the extraction process is shown in Figure 1. The CO_2 is compressed and heated to the operating conditions (supercritical state). The extraction step follows and here the soluble compounds are separated from the material. In this step the solubility of compound or mixture of compounds has to be the highest while in the next separation step the solubility of compound in SCF has to be the lowest. The picture of extractor is shown in Figure 2.

The phase equilibrium and mass transfer data are usually obtained with research and measurements, which takes a lot of time and money. Extended research on this field is in progress. One of the major activities

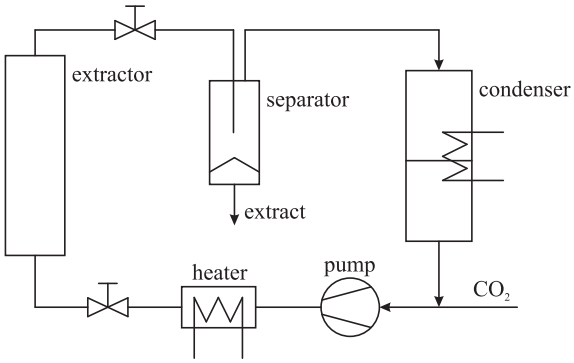


Figure 1. Simple flow sheet of extraction process.

Figure 2. High pressure extraction unit (Faculty of Chemistry and Chemical Engineering, University of Maribor, Slovenia).

is kinetic modeling, whose goal is to reduce the time used for measurements and to cut the experimental cost.

3. Sovova's Mass Transfer Model

Mass transfer data are obtained from the extraction curve that shows a plot of total mass of oil extracted vs. the time or total mass of solvent used. The Sovova's model is used to describe extraction kinetics curve for SFE of substances for plant materials [8]. It extends Lack's plug-flow model [3]. It separates the extraction curve into three periods (Figure 3). In the first period the easily accessible solute is linear with a slope close to the value of oil solubility in solvent. In the second period, rate

of extraction drops rapidly and continues with a third period where the extraction is almost linear but with much smaller extraction rate than in the first period.

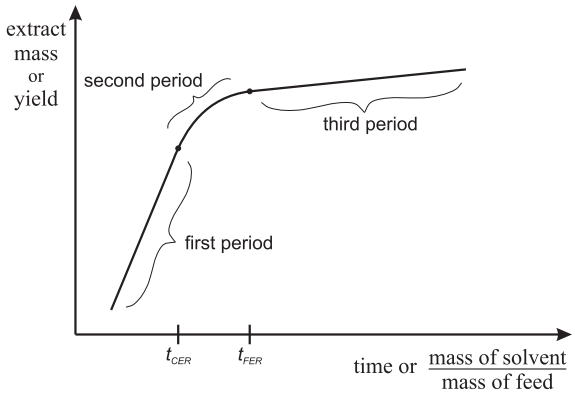


Figure 3. Extraction curve separated into three periods.

Sovova's model is described with the following equations, where extraction curve is defined as a function of time ($y(t)$).

$$y(t) = \begin{cases} Q_{CO_2} y_r [1 - exp(-Z)]t & \text{for } 0 \leq t \leq t_{CER} \\ Q_{CO_2} y_r [t - t_{CER} exp(z_w - Z)] & \text{for } t_{CER} < t \leq t_{FER} \\ m_{SI}[x_0 - \frac{y_r}{W} ln\{1 + [exp(\frac{W x_0}{y_r}) - 1] exp[\frac{W Q_{CO_2}}{m_{SI}}(t_{CER} - t)]\frac{x_k}{x_0}\}] & \\ & \text{for } t > t_{FER} \end{cases} \tag{1}$$

where Q_{CO_2} represents solvent flow rate, y_r solubility of compounds at operating conditions, m_{SI} mass of non-extractable material (extract-free), x_0 the initial mass of extractable material, relative to mass of non-extractable material and x_k initial mass of extractable material in intact cells, relative to mass of non-extractable material. The axial coordinate z_w, in the second extraction period, is calculated using equation (2).

$$z_w = \frac{Z y_r}{W x_0} ln \frac{x_0 exp[W Q_{CO_2}(t - t_{CER})/m_{SI}] - x_k}{x_0 - x_k} \tag{2}$$

Times t_{CER} and t_{FER} represent the transition between the fast- and slow-extraction periods and are calculated using equations (3) and (4).

$$t_{CER} = \frac{(x_0 - x_k) m_{SI}}{y_r Z Q_{CO_2}} \tag{3}$$

$$t_{FER} = t_{CER} + \frac{m_{SI}}{W Q_{CO_2}} ln \frac{x_k + (x_0 - x_k) exp(\frac{W x_0}{y_r})}{x_0} \tag{4}$$

Parameters Z and W are directly proportional to mass transfer coefficients by equation (5) and (6).

$$Z = \frac{k_f a_0 \rho}{Q_{CO_2}(1-\epsilon)\rho_s} = \frac{F}{Q_{CO_2}} \tag{5}$$

$$W = \frac{k_s a_0}{Q_{CO_2}(1-\epsilon)} = \frac{S}{Q_{CO_2}} \tag{6}$$

The k_f and k_s are mass transfer coefficients in CO_2 and solid phase, respectively. Value a_0 represents a specific interfacial area, ρ is density of the solvent phase, ρ_s is density of the material and ϵ represents void fraction in bed. In this paper, parameters Z, W and x_k were chosen to be optimized with EA, similar to parameters F, S and x_k fitted in [8].

4. The Evolutionary Algorithm for Sovova's Model

To solve the parameter fitting problem, the EA has been designed with properties borrowed from different EAs. The proposed EA could be classified as $(\mu + \lambda)$-ES (Evolutionary Strategies), where λ, number of offspring is not fixed, but variable parameter during the evolutionary cycle. Despite ES has no crossover operator, in this case the crossover operator was used. Moreover, the difference between the proposed EA and ES is also in the selection process. ES uses deterministic selection, while the proposed EA uses tournament selection ($k = 2$).

The individual of population is represented with a vector of float values [Z, W, x_k], which represent estimating parameters of the model. Exploration and exploitation of search space [4] is achieved with implemented crossover and mutation operators. The 1-point crossover was selected, because of small individual size. The probability of crossover p_c is fixed throughout algorithm run. For every value (gene) in the individual, the mutation operator determines if it will be mutated, based on the probability of mutation p_m. To calculate mutated value equation (7) or (8) are used.

$$v_n = v + r^3(v_{max} - v) \tag{7}$$

$$v_n = v - r^3(v - v_{min}) \tag{8}$$

where v_n represents value that replaces the old value v, r is the random value between 0 and 1, v_{max} is the maximum value and v_{min} the minimum value, both defined by user for every parameter separately. To achieve that the smaller value happened more often, the power of three on the random value was used. Individual fitness value is calculated using equation (9).

$$\phi(y, y_{obs}) = \sum_{i=1}^{m} \frac{|y(t_i) - y_{obs}(t_i)|}{y_{obs}(t_i)} \tag{9}$$

where $y(t_i)$ represents the calculated amount of extract at time t_i and is calculated using equation (1), while $y_{obs}(t_i)$ represents the experimentally obtained amount of extract.

In global optimizer [5] the upper and lower boundaries of the model parameters estimated (Z, t_{CER} and t_{FER}) were studied in detail and limited to very small intervals. In contrast the boundaries of parameters estimated in proposed EA were not dependent of each other. For parameters Z and W the minimum value was 0 and the maximum was 10, while the boundaries for parameter x_k were between 0 and x_0 (which fits with the description of x_k).

5. Results

To make the comparison between global optimizer [5] and EA, proposed in this paper, the same data for SFE with CO_2 of vetiver roots at 40°C and 200 bars, were used. Six experiments were made at different process conditions. The first three problems are small pilot scales and the next three are the large production scales. The values equal for all problems are initial mass of extractable material $x_0 = 0.0619$ and extract solubility $y_r = 0.06$. Other conditions are given in Table 1.

Firstly, the control parameter tuning was performed for parameters *pop_size* (1000, 5000, 10000), p_c (0.1, 0.2, 0.3) and p_m (0.05, 0.1, 0.2). From the results of control parameter tuning, the following parameter values were used on all six problems and for comparison with global optimizer: $pop\_size = 5000$; $max\_gen = 300$; $p_c = 0.1$; $p_m = 0.2$.

For each problem, 100 runs of the algorithm was performed. Commonly, the results obtained from EA runs are averaged and show the expected value, if the user makes a random run afterwards. In this case average value of results is not of much help, because the comparison of estimated model parameters values between proposed EA and global optimizer is also needed. Therefore, the best and the worst solutions with model parameters values are presented in Table 1. For the comparison of results, the percentage of average absolute relative deviation (AARD), calculated using equation (10), was used. It shows the deviation of the model from experimental data.

$$\phi(y, y_{obs}) = \frac{100}{m} \sum_{i=1}^{m} \frac{|y(t_i) - y_{obs}(t_i)|}{y_{obs}(t_i)} \tag{10}$$

Results from our EA, regarding AARD measure, have been statistically compared against the results from global optimizer. One sample

Table 1. Comparison of AARD and estimated model parameter values between global optimizer and proposed EA. *Calculated from Z, W and x_k using equations (3) and (4).

Conditions	**Global optimizer**	**Best run of EA**	**Worst run of EA**
Problem 1: $Q_{CO_2} = 0.85$ $m_{total} = 3.53$	AARD = 2.09499 $Z = 0.083382$ $W = 0.012494$ $t_{CER} = 24.372$ $t_{FER} = 26.411$	AARD = 2.09151 $Z = 0.0833816$ $W = 0.0124254$ $x_k = 0.0307196$ $t_{CER*} = 24.374312$ $t_{FER}{}^* = 26.413146$	AARD = 2.09178 $Z = 0.0833820$ $W = 0.0124398$ $x_k = 0.0307221$ $t_{CER}{}^* = 24.372241$ $t_{FER}{}^* = 26.410919$
Problem 2: $Q_{CO_2} = 0.85$ $m_{total} = 3.53$	AARD = 1.95147 $Z = 0.104925$ $W = 0.009490$ $t_{CER} = 18.766$ $t_{FER} = 20.740$	AARD = 1.9178 $Z = 0.0938798$ $W = 0.0074551$ $x_k = 0.0298954$ $t_{CER}{}^* = 22.220875$ $t_{FER}{}^* = 24.310840$	AARD = 1.91955 $Z = 0.0940318$ $W = 0.0074529$ $x_k = 0.0298973$ $t_{CER}{}^* = 22.183639$ $t_{FER}{}^* = 24.273479$
Problem 3: $Q_{CO_2} = 6.3$ $m_{total} = 26.2$	AARD = 1.62209 $Z = 0.228515$ $W = 0.056783$ $t_{CER} = 13.627$ $t_{FER} = 16.762$	AARD = 1.59072 $Z = 0.2285151$ $W = 0.0552940$ $x_k = 0.0141545$ $t_{CER}{}^* = 13.637772$ $t_{FER}{}^* = 16.774324$	AARD = 1.59073 $Z = 0.2285151$ $W = 0.0552929$ $x_k = 0.0141544$ $t_{CER}{}^* = 13.637800$ $t_{FER}{}^* = 16.774359$
Problem 4: $Q_{CO_2} = 17$ $m_{total} = 71.68$	AARD = 1.60608 $Z = 0.226704$ $W = 0.097600$ $t_{CER} = 14.269$ $t_{FER} = 17.537$	AARD = 1.56185 $Z = 0.2267038$ $W = 0.0943289$ $x_k = 0.0129419$ $t_{CER}{}^* = 14.291571$ $t_{FER}{}^* = 17.563863$	AARD = 1.56249 $Z = 0.2267038$ $W = 0.0939244$ $x_k = 0.0129218$ $t_{CER}{}^* = 14.297438$ $t_{FER}{}^* = 17.570886$
Problem 5: $Q_{CO_2} = 17$ $m_{total} = 72.26$	AARD = 1.947 $Z = 0.259076$ $W = 0.087644$ $t_{CER} = 12.182$ $t_{FER} = 15.371$	AARD = 1.84725 $Z = 0.2590756$ $W = 0.0816559$ $x_k = 0.0140226$ $t_{CER}{}^* = 12.328726$ $t_{FER}{}^* = 15.552807$	AARD = 1.84727 $Z = 0.2590756$ $W = 0.0816523$ $x_k = 0.0140224$ $t_{CER}{}^* = 12.328778$ $t_{FER}{}^* = 15.552870$
Problem 6: $Q_{CO_2} = 6.3$ $m_{total} = 73.1$	AARD = 3.10782 $Z = 0.283353$ $W = 0.220752$ $t_{CER} = 24.083$ $t_{FER} = 31.207$	AARD = 3.07182 $Z = 0.2833528$ $W = 0.2174639$ $x_k = 0.0243632$ $t_{CER}{}^* = 24.125243$ $t_{FER}{}^* = 31.257685$	AARD = 3.07291 $Z = 0.2833527$ $W = 0.2168423$ $x_k = 0.0243396$ $t_{CER}{}^* = 24.140419$ $t_{FER}{}^* = 31.276212$

t-test shows that our EA performs significantly better than global optimizer [5]. However, if the comparison between parameter values (Z, W, t_{CER} and t_{FER}) is done, it can be seen, that they are in most cases almost identical. The statistically significant difference is obtained because results from all EA runs were better than the result from global optimizer, although they are not much different. This means that both algorithms found the solution in almost the same search space location. It can be concluded that proposed EA found reasonable solutions with physical meaning without studying the connections between model parameters that was done in [5]. The upper and lower boundaries of parameter values were wider, although this did not affect the EA effectiveness. Therefore the proposed EA is more robust and with high confidence it can be concluded, that random run of proposed EA with given control parameter values, will give at least as good results than global optimizer [5]. Model parameter values obtained with both algorithms are almost equal, therefore the extraction curves are the same, except for the second problem, where our EA found better fitting with different model parameter values. Figure 4 shows both extraction curves and experimental points for the second problem.

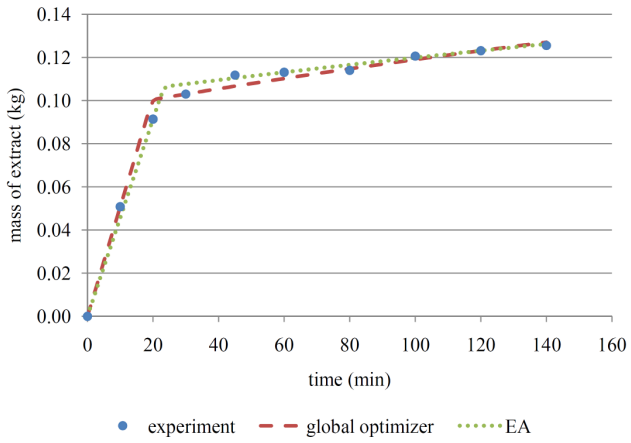


Figure 4. Difference between the extraction curve obtained with proposed EA and with global optimizer for the second problem.

The computational time used in global optimizer is described with equation $t = 10^{-6} n_1 n_2 n_3 n_{obs}$ (s), where n_1, n_2, n_3 represent the number of grid points for each model parameter and n_{obs} represents number of experimental points. In case, when n_1,n_2,n_3 are 100, this means about 10 seconds for data with 10 experimental points. The average run time for our EA of the same problem was 5 seconds. For testing, the

Intel processor with 2.66GHz was used and the algorithm was written in Java. Although the EA is population based technique, the time used for computation was better then time mentioned in [5] for global optimizer.

6. Conclusions

In this work, an evolutionary algorithm was proposed for fitting the SFE extraction data with Sovova's mass transfer model. Obtained results through the proposed evolutionary algorithm were compared with results from global optimizer. Our evolutionary algorithm proved successful on all tested problems and the results were significantly better than results obtained with global optimizer. The proposed evolutionary algorithm also proved to be faster, although the search space was larger because of parameter boundaries. The implemented evolutionary algorithm can easily be adapted to any other model, with minor changes in implementation (even with a higher number of model parameters and therefore larger search space).

References

[1] L. Balland, L. Estel, J.-M. Cosmao, and N. Mouhab. A genetic algorithm with decimal coding for the estimation of kinetic and energetic parameters. *Chemometr. Intell. Lab.*, 50(4):121–135, 2000.

[2] Ž. Knez, M. Škerget, and M. Knez Hrnčič. *Separation, extraction and concetration processes in the food, beverage and nutraceutical industries.* Chapter: Extraction using supercritical fluids. Woodhead Publishing Limited, Cambridge, In press, 2010.

[3] E. A. Lack. Kriterien zur Auslegung von Anlagen für die Hochdruckextraktion von Naturstoffen. Ph.D. thesis, TU Graz, 1985.

[4] S. H. Liu, M. Mernik, and B. Bryant. To explore or to exploit: An entropy-driven approach for evolutionary algorithms, *Int. J. Knowledge-Based Intel. Eng. Sy.*, 13(3-4):185–206, 2009.

[5] J. Martinez and J.M. Martinez. Fitting the Sovova's supercritical fluid extraction model by means of a global optimization tool. *Comput. Chem. Eng.*, 32:1735–1745, 2008.

[6] Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs.* Third Edition. Springer-Verlag, 1996.

[7] T.Y. Park and G.F. Froment. A Hybrid Genetic Algorithm for the Estimation of Parameters in Detailed Kinetic Models. *Comput. Chem. Eng.*, 22:S103-S110, 1998.

[8] H. Sovova. Rate of the vegetable oil extraction with supercritical CO2 - I. Modelling of extraction curves. *Chem. Eng. Sci.*, 49(3):409–414, 1994.

[9] L. Wang and C.N. Kim. On the Feasibility and Relibility of Nonlinear Kinetic Parameter Estimation for a Multi-Component Photocatalytic Process. *Korean J. Chem. Eng.*, 18(5):652–661, 2001.

PARALLEL DIFFERENTIAL EVOLUTION FOR SIMULATION-BASED MULTIOBJECTIVE OPTIMIZATION OF A PRODUCTION PROCESS

Matjaž Depolli
Department of Communication Systems
Jožef Stefan Institute, Ljubljana, Slovenia
matjaz.depolli@ijs.si

Erkki Laitinen
Department of Mathematical Sciences
University of Oulu, Finland
erkki.laitinen@oulu.fi

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

Abstract This paper presents a parallel Differential Evolution algorithm for solving numerical multiobjective optimization problems and demonstrates its performance on process parameter tuning in industrial continuous casting of steel. It briefly introduces multiobjective optimization, and presents the parallel algorithm designed for homogeneous computer architectures. It further describes the considered problem and the experimental setup, and reports the results in view of their effectiveness and efficiency. The results indicate the algorithm is a suitable platform for parallel multiobjective optimization in case of problems with time-consuming solution evaluation.

Keywords: Differential evolution, Multiobjective optimization, Pareto optimality, Parallelization, Production optimization

1. Introduction

Practical optimization often has to deal with several objectives, which, in addition, may be conflicting. In manufacturing, for example, permanent efforts are taken to increase productivity and product quality, while production costs and environmental degradation need to be reduced to the lowest possible amount. Similarly, customers seek maximum performance of products and services at a minimum price. Solving such problems is known as multiobjective optimization. It radically differs from traditional, single-objective optimization in that not only one, but a number of optimal solutions exist, each representing a particular compromise among the objectives. The appearance of empirical population-based search techniques, such as evolutionary algorithms [5], made it possible to search for multiple trade-off solutions among which a user can choose the most suitable one according to additional preferences [1, 2, 3, 10].

A prerequisite for optimizing the performance of a device or processes is its reliable numerical model which, linked with an optimization algorithm, serves as an evaluator of candidate solutions. Such models frequently rely on discretization of the application domain and involve complex iterative solving procedures. If the optimization algorithm is population-based and the problem involves multiple objectives, the number of evaluations needed to obtain acceptable results is usually high and results in computationally demanding numerical procedures. To alleviate this difficulty, parallel population-based search algorithms have been studied with an emphasis on multiobjective optimization in the last decade [11, 17].

This paper contributes to these efforts by presenting a parallel version of DEMO (Differential Evolution for Multiobjective Optimization), whose original serial implementation has proved successful in solving numerical multiobjective optimization problems [14, 16]. The parallelization approach was master–slave that relies on the inherent parallelism of evolutionary algorithms. The algorithm was primarily designed to run on homogeneous parallel computer architectures. It is intended to speedup time-consuming simulation-based optimization calculations. We demonstrate its applicability in multiobjective parameter tuning of industrial continuous casting of steel.

Further organization of the paper is as follows. Section 2 reviews the concepts of and algorithmic approaches to multiobjective optimization. Section 3 presents the proposed parallel algorithm, and Section 4 the industrial problem used to demonstrate the performance of the algorithm. Section 5 describes the experimental setup, and Section 6 reports the

results. In Section 7 the work is summarized and directions for further work are outlined.

2. Multiobjective Optimization: Concepts and Algorithms

In multiobjective optimization, the objective function $\mathbf{c}$ is a mapping from an n-dimensional variable space X, also called the decision space, to an m-dimensional objective space $Z \subseteq \mathbf{R}^m$, $m \geq 2$. In other words, variable vectors are evaluated by a vector function comprising several objectives, $\mathbf{c} : (x_1, \ldots, x_n) \mapsto (c_1(x_1, \ldots, x_n), \ldots, c_m(x_1, \ldots, x_n))$. The objective vectors in Z are partially ordered according to the *Pareto dominance* relation. Vector $\mathbf{z}^1$ *dominates* vector $\mathbf{z}^2$ ($\mathbf{z}^1 \prec \mathbf{z}^2$) if and only if $\mathbf{z}^1$ is better than $\mathbf{z}^2$ in at least one objective, and not worse than $\mathbf{z}^2$ in all other objectives. If no objective vector from Z dominates the objective vector $\mathbf{z}$, $\mathbf{z}$ is said to be *nondominated*. All nondominated vectors from Z form the set of optimal objective vectors, called the *Pareto optimal front*. Each vector from the Pareto optimal front represents a specific trade-off between the objectives, and without additional information on preferences among the objectives optimal vectors cannot be preferred to one another.

Traditionally, multiobjective optimization was carried out by first transforming the problems into a single-objective form and then solving them using techniques of single-objective optimization. The transformation was done either by applying the weighted sum of objectives, or identifying the key objective for optimization and considering others as constraints. Both approaches require additional information on relative importance of the objectives and return specific solutions. They are nowadays referred to as *preference-based* multiobjective optimization, while, on the other hand, recent population-based multiobjective optimizers perform *ideal* multiobjective optimization where an approximation of the Pareto optimal front, or the *approximation set*, is found in a single execution of the algorithm.

An appropriate algorithmic basis to support ideal multiobjective optimization is evolutionary computing. Many evolutionary algorithms for multiobjective optimization, including the well-known NSGA-II (Nondominated Sorting Genetic Algorithm) [4], employ genetic algorithms to explore the decision space. Another evolutionary algorithm that has been successfully adapted for multiobjective optimization is Differential Evolution (DE) [12, 13]. An example of such adaptation is a DE-based algorithm called DEMO [14].

Both DE and DEMO are intended for numerical optimization and encode solutions as n-dimensional real-valued vectors. New solutions (candidates) are generated from the existing ones through vector addition and scalar multiplication. After the creation of a candidate in DE, the candidate is compared to its parent and the best of them remains in the population, while the other is discarded. In DEMO, on the other hand, the candidate and its parent are compared using the Pareto dominance relation. If the candidate dominates the parent, it replaces the parent in the current population. If the parent dominates the candidate, the candidate is discarded. Otherwise, when the candidate and its parent are incomparable, the candidate is added to the population. After generating candidates for each parent individual in the population, the population size possibly exceeds the predefined value. In this case DEMO truncates the population to the original size using nondominated sorting and the crowding distance metric known from NSGA-II [4]. A detailed description of DEMO and its empirical evaluation are presented in [14, 16]. Besides its original serial version, a parallel variant of the algorithm has been implemented recently [7]. The next sections outlines its characteristics.

3. Parallel Differential Evolution for Multiobjective Optimization

The parallelization approach used in the parallel version of the differential evolution for multiobjective optimization is single-walk parallelization [11], which is aimed at speeding up the underlying sequential algorithm while its basic behavior is preserved. It implements the master–slave parallelization scheme, the most straightforward way of parallelizing the evolutionary algorithms that makes use of their inherent parallelism [17]. Although designed to run on homogeneous parallel computer architectures, it can also use heterogeneous architectures, but with lower utilization of faster processors.

In the parallel algorithm, creation of the initial population, variation of individuals and survivor selection are performed by the master, while candidate evaluation is run in parallel by all processors. Each generation starts with the master process holding a population of unevaluated individuals. These are then evaluated by the master and the slave processes in parallel, which requires interprocess communication. The interprocess communication is performed in two parts. The first part distributes the data on the individuals to the slave processes, and the second part returns the evaluation results to the master process. After the evaluation results for all individuals are known, the master process applies

the evolutionary algorithm operators and spawns the next generation. The slave processes are idle at this time, waiting to receive the data on individuals of the next generation. The master and the slave process are shown in Algorithms 1 and 2, respectively.

Clearly, the highest efficiency of this parallel algorithm is achieved on computers with homogeneous processors and in problem domains where the solution evaluation time is long and does not vary with solutions. The algorithm was designed for solving multiobjective optimization problems with the candidate evaluation time much longer than the time of executing the remaining algorithm steps. It is particularly suitable for real-world simulation-based optimization tasks, such as the production optimization problem discussed in the following section.

4. A Multiobjective Production Optimization Problem

We deal with multiobjective optimization of industrial continuous casting of steel. This process is used at steel plants worldwide to produce various steel semi-manufactures. It starts with pouring liquid steel into the mold which is cooled by internal water flow. The cooling water extracts heat from the molten steel and initiates the formation of a solid shell. Exiting the mold, the solidifying steel slab enters the secondary cooling area of the casting system where it is cooled by water sprays positioned at the center and the corner positions along the slab. The secondary cooling area consists of several cooling zones where coolant flows at all positions can be tuned individually.

The coolant flows should be set in such a way that the resulting slab surface temperatures approach the predefined target temperatures as closely as possible. In metallurgical practice, satisfying this criterion is known to increase the quality of the cast steel. For this purpose, a minimization objective is defined in the form of the sum of differences between the actual and target temperatures over the secondary cooling zones:

$$c_1 = \sum_{i=1}^{N_\mathrm{Z}} |T_i^{\mathrm{center}} - T_i^{\mathrm{center}*}| + \sum_{i=1}^{N_\mathrm{Z}} |T_i^{\mathrm{corner}} - T_i^{\mathrm{corner}*}|, \tag{1}$$

where N_Z is the number of zones, T_i^{center} and T_i^{corner} are the slab surface temperatures at the center and the corner positions in zone i, and $T_i^{\mathrm{center}*}$ and $T_i^{\mathrm{corner}*}$ the target temperatures in zone i.

Another empirical metallurgical criterion refers to the core length, l^{core}, i.e. the distance from the mold exit to the point of complete solidification of the slab. The target value for the core length, $l^{\mathrm{core}*}$, is

Algorithm 1 Parallel differential evolution for multiobjective optimization: The master process

```
1: create an empty initial population P
2: while stopping criterion not met do
3:   create an empty population P_new
4:   if P empty then
5:     fill P with random solutions
6:   else
7:     for each solution P_i, i = 1..pop_size from P do
8:       randomly select three different solutions I_1, I_2, I_3 from P
9:       create a candidate solution C := I_1+F·(I_2 − I_3)
10:      alter C by crossover with P_i
11:      add C to P_new
12:    end for
13:  end if
14:  repeat
15:    select unevaluated solutions C_1..C_n from P_new, where n =
       min(num_unevaluated, num_slaves + 1)
16:    send C_1..C_{n−1} to slave processes for evaluation
17:    evaluate C_n
18:    receive evaluation results for C_1..C_{n−1} from slave processes
19:    for solutions C_i from P_new and their parents P_i from P, i = 1..n
       do
20:      if C_i dominates P_i then
21:        leave C_i in P_new
22:      else if P_i dominates C_i then
23:        replace C_i with P_i in P_new
24:      else
25:        add P_i into P_new
26:      end if
27:    end for
28:  until all solutions from P_new evaluated
29:  if P_new contains more than pop_size solutions then
30:    truncate P_new
31:  end if
32:  copy P_new into P
33: end while
34: send termination request to all slave processes
```

prespecified, and the actual core length should be as close to it as possible. Shorter core length may result in unwanted deformations of the

Algorithm 2 Parallel differential evolution for multiobjective optimization: The slave process

```
1: while termination not requested do
2:   wait for a message from the master process
3:   if the message contains an individual C for evaluation then
4:     evaluate C
5:     send the evaluation result for C to the master process
6:   end if
7: end while
```

slab, while longer core length has to be avoided for safety reasons. This results in another objective to be minimized:

$$c_2 = |l^{\text{core}} - l^{\text{core}*}|. \tag{2}$$

Moreover, coolant flows cannot be set arbitrarily, but according to the technological constraints. For each zone, minimum and maximum values are prescribed for the center and the corner coolant flows.

As a result, we are faced with a constrained two-objective optimization problem with conflicting objectives. To solve it by means of parallel multiobjective optimization, we employ a numerical simulator of the casting process that, given the coolant flow values, calculates the temperature field in the slab and returns the values of objectives c_1 and c_2. For this purpose we use a numerical model of the process with Finite Element Method (FEM) discretization of the temperature field and the corresponding nonlinear equations solved with relaxation iterative methods. The model has previously been used in a single-objective optimization study of the casting process [6, 8], and in preliminary multiobjective optimization studies carried out using the serial variant of the DEMO algorithm [9].

5. Experimental Setup

Numerical experiments in optimizing the continuous casting process were performed to analyze both the effectiveness and efficiency of the parallel algorithm. The parallel computer architecture used in the experiments is a cluster of 16 dual-processor nodes The cluster consists of identical personal computers, each containing two AMD Opteron 244 processors, 1024 MB of RAM, a hard disk drive, and six Full-Duplex Gigabit Ethernet ports. On each computer, there is an independent installation of the Fedora Core 2 operating system and the MPICH library, an implementation of the Message Passing Interface (MPI) [15] supporting

communication between the nodes. During the experiments, all nodes were required to be running only the background system processes which leaves nearly all capabilities available for the parallel optimization algorithm.

The test optimization problem used in the experiments referred to a casting machine comprising nine secondary cooling zones. In each zone, cooling water is dispersed to the slab at the center and the corner positions, therefore 18 coolant flows need to be tuned. The considered target slab surface temperatures and coolant flow constraints are shown in Table 1.

Table 1. Target temperatures and water flow constraints for the casting machine considered in numerical experiments.

Slab position	Zone number	Target temp. [°C]	Water spray number	Min. flow [m^3/h]	Max. flow [m^3/h]
	1	1050	1	7.1	26.1
C	2	1040	2	22.8	57.5
e	3	980	3	13.3	39.9
n	4	970	4	1.5	7.9
t	5	960	5	2.7	10.0
e	6	950	6	0.8	6.5
r	7	940	7	0.7	5.9
	8	930	8	1.0	5.8
	9	920	9	1.2	6.2
	1	880	10	7.1	26.1
C	2	870	11	22.8	57.5
o	3	810	12	13.3	39.9
r	4	800	13	1.2	3.5
n	5	790	14	2.4	4.4
e	6	780	15	2.4	2.9
r	7	770	16	0.7	5.9
	8	760	17	1.0	5.8
	9	750	18	1.2	6.2

Optimization calculations were performed for a particular steel grade with the slab cross-section of 1.70 m × 0.21 m and the casting speeds of 1.6 m/min. The target core length, $l^{\mathrm{core}*}$, was 27 m.

Candidate solutions were encoded as real-valued vectors, representing coolant flow values at the center and the corner positions in the zones of the secondary cooling area. A single simulation-based evaluation of a candidate solution took about 30 seconds. The population size of 32 was chosen to perform load-balanced experiments on hardware configurations

comprising 1, 2, 4, 16, and 32 processors. Moreover, the number of generations was 300, resulting in 9600 evaluations per algorithm run, which was sufficient for the algorithm to converge. For each number of processors, the algorithm was run 25 times.

6. Results

The results produced by the parallel optimization procedure were highly repetitive both over the runs at each number of processors and over various numbers of processors used. These results also matched perfectly with those obtained previously with the original serial variant algorithm [9]. Since detailed results and their statistical evaluation cannot be shown due to space limitation, we provide an illustrative example of a nondominated front of solutions found by the algorithm (Fig. 1). It confirms the conflicting nature of the two objectives: improving the coolant flow settings with respect to the temperature differences makes them worse with respect to the core length difference. Moreover, a systematic analysis of the solutions shows that the actual slab surface temperatures are usually close to or higher than the target temperatures, while the core length is shorter than or equal to the target core length. For example, for a trade-off coolant setting from the center of the front shown in Fig. 2, the sum of temperature differences equals 315°C, the core length difference is 1.1 m, and the resulting temperature differences are shown in Fig. 3.

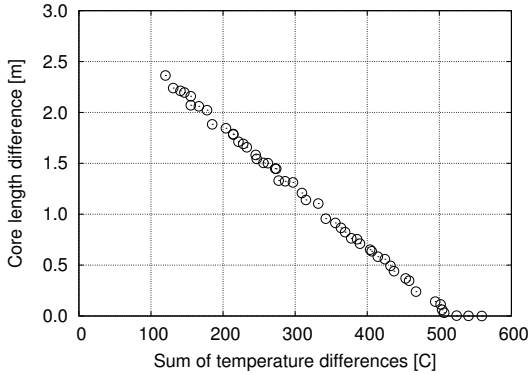


Figure 1. Nondominated solutions found for the casting speed of 1.6 m/min.

On top of that, the execution times (wall clock times) spent by the algorithm running on various numbers of processors (Fig. 4) show the expected speedup with the increasing number of processors. Of course, this is possible since the slave processors are load balanced, i. e. they are

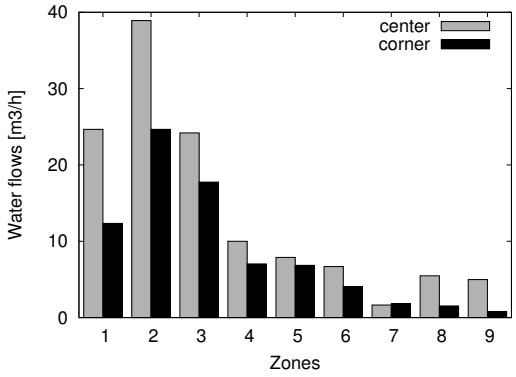


Figure 2. Optimized coolant flows in a trade-off solution.

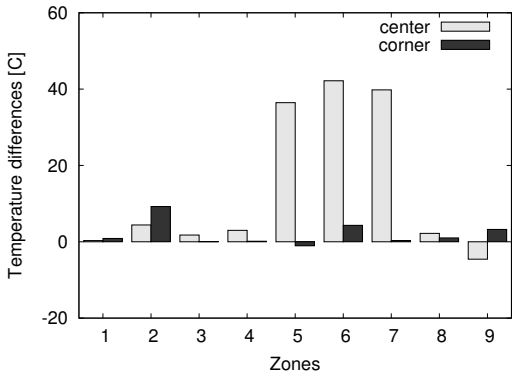


Figure 3. Temperature differences in a trade-off solution.

equally fast and performing the same number of evaluations per generation, and the evaluations take constant time. In practical simulation-based optimization, these conditions can be met to a high degree. Given a cluster of identical computers, and a problem with the solution evaluation time independent of individuals, the population size should be set equal to a multiple of the number of processors.

7. Conclusion

In evolutionary multiobjective optimization with time-consuming evaluation of solutions, parallelization is a valuable approach to reducing the time needed to get acceptable results. For this purpose we parallelized the Differential Evolution for multiobjective optimization and analyzed its performance in tuning the coolant flows in industrial continuous cast-

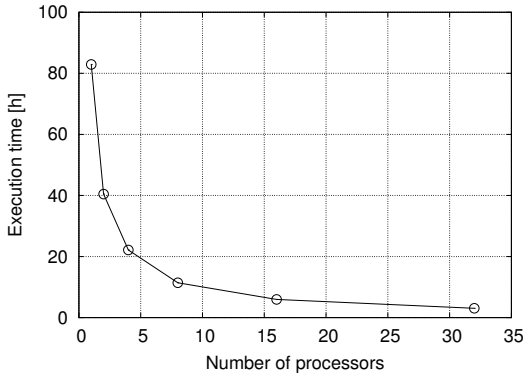


Figure 4. Execution time vs. the number of processors.

ing of steel. The objectives were defined empirically to ensure the highest possible product quality and process safety. The obtained fronts of nondominated solutions confirm the conflicting nature of the objectives. Allowing better understanding of the process behavior and providing a clear picture of trade-offs in process parameter setting, the results are useful both to the engineers operating such devices and the designers of new casting devices.

In the future, the parallel algorithm will be adjusted to achieve the best possible utilization of processors on heterogeneous computer architectures, or on homogeneous architectures with varying background load, or in solving problems with the solution evaluation time depending on individuals. From the problem domain point of view, we will study the effect of additional process characteristics, such as steel grade and slab geometry, on the optimization results.

Acknowledgment

This work was supported by the Slovenian Research Agency and the Academy of Finland under the Slovenian-Finnish project BI-FI/09-007 *Multiobjective Optimization of Technological Processes*, and by the Slovenian Research Agency under research programmes P2-0095 *Parallel and Distributed Systems*, and P2-0209 *Artificial Intelligence and Intelligent Systems*.

References

[1] J. Branke, K. Deb, K. Miettinen, and R. Slowinski (Eds.) *Multibjective Optimization – Interactive and Evolutionary Approaches*. Lecture Notes in Computer Science, Vol. 5252, Springer, Berlin, 2008.

[2] C. A. Coello Coello, D. A. Van Veldhuizen, and G. B. Lamont. *Evolutionary Algorithms for Solving Multi-Objective Problems*. Kluwer Academic/Plenum Publishers, New York, 2002.

[3] K. Deb. *Multi-Objective Optimization Using Evolutionary Algorithms*. John Wiley & Sons, Chichester, 2001.

[4] K. Deb, A. Pratap, S. Agrawala, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE T. Evol. Comput.*, 6(2):182–197, 2002.

[5] A. E. Eiben and J. E. Smith. *Introduction to Evolutionary Computing*. Springer, Berlin, 2003.

[6] B. Filipič. Efficient simulation-based optimization of process parameters in continuous casting of steel. In *Proc. First Invited Conference on Automatic Process Optimization in Materials Technology (COST 526)*, pages 193–198, 2005.

[7] B. Filipič and M. Depolli. Parallel evolutionary computation framework for single- and multiobjective optimization. In R. Trobec, M. Vajteršic, and P. Zinterhof (Eds). *Parallel Computing – Numerics, Applications, and Trends*, pages 217–240, Springer, Dordrecht, 2009.

[8] B. Filipič and E. Laitinen. Model-based tuning of process parameters for steady-state steel casting. *Informatica*, 29(4):491-496, 2005.

[9] B. Filipič, T. Tušar, and E. Laitinen. Preliminary numerical experiments in multiobjective optimization of a metallurgical production process. *Informatica*, 31(2): 233–240, 2007.

[10] J. Knowles, D. Corne, and K. Deb (Eds.). *Multibjective Problem Solving from Nature – From Concepts to Applications*. Springer, Berlin, 2008.

[11] A. J. Nebro, F. Luna, E.-G. Talbi, and E. Alba. Parallel Multiobjective Optimization. In E. Alba (Ed). *Parallel Metaheuristics – A New Class of Algorithms*, pages 371–394, John Wiley & Sons, Hoboken, NJ, 2005.

[12] K. V. Price and R. M. Storn. Differential evolution – A simple evolution strategy for fast optimization. *Dr Dobbs J.*, 22(4):18–24, 1997.

[13] K. V. Price, R. M. Storn, and J. A. Lampinen. *Differential Evolution: A Practical Approach to Global Optimization*. Springer, Berlin, 2005.

[14] T. Robič and B. Filipič. DEMO: Differential evolution for multiobjective optimization. *Lect. Notes Comput. Sc.*, 3410:520–533, 2005.

[15] M. Snir, S. Otto, S. Huss-Lederman, D. Walker, and J. Dongarra, *MPI – The Complete Reference*. The MIT Press, Cambridge, 1996.

[16] T. Tušar and B. Filipič. Differential evolution versus genetic algorithms in multiobjective optimization. *Lect. Notes Comput. Sc.*, 4403:257–271, 2007.

[17] D. A. van Veldhuizen, J. B. Zydallis, G. B. Lamont. Considerations in engineering parallel multiobjective evolutionary algorithms. *IEEE T. Evol. Comput.*, 7(2):144–173, 2003.

ANT COLONY OPTIMIZATION FOR BROADCASTING IN SENSOR NETWORKS UNDER A REALISTIC ANTENNA MODEL

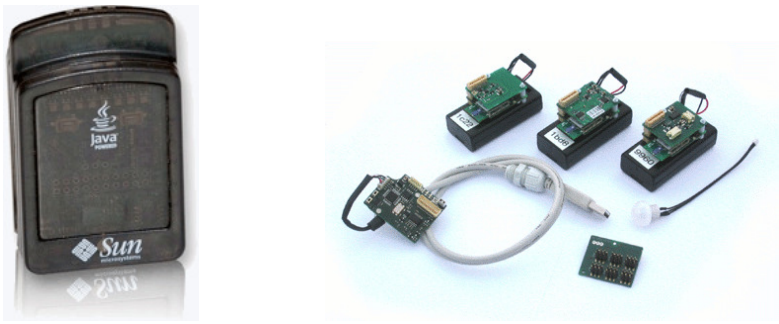
Hugo Hernández, Christian Blum
ALBCOM research group
Universitat Politècnica de Catalunya, Barcelona, Spain
{hhernandez; cblum}@lsi.upc.edu

Abstract Most, if not all, works from the literature dealing with minimum energy broadcasting in wireless ad-hoc networks such as sensor networks consider antenna models that allow the adjustment of the emission power to any desired real value from zero up to the maximum sensing range. However, looking at the currently available hardware shows that these antenna models are not very realistic. In this work we therefore adapt the currently best available algorithm for minimum energy broadcasting for a more realistic antenna model which only offers few different levels of emission power. The obtained results show that this ant colony optimization algorithm performs well in comparison to a standard heuristic known from the literature.

Keywords: Ant colony optimization, Broadcasting, Sensor networks

1. Introduction

Wireless ad-hoc networks—such as sensor networks—are being used in practical scenarios such as the monitoring of certain events [8]. Sensor nodes are generally equipped with omni-directional antennas for sending and receiving information. They have a packet-forwarding capability in order to communicate via shared and limited radio channels. In order to transmit information, a sender node must adjust its emission power in order to reach the desired receiver node. As network lifetime is limited by batteries, energy saving is critical. A fundamental problem in sensor networks arises when one node is required to transmit data to all other nodes of the network. This scenario is known as *broadcasting*. Obviously, for broadcasting to be energy-efficient, the emission powers of the sensor



(a) A SunSPOT, © Sun Microsystems (b) iSense sensor nodes, © Coalesenses GmbH

Figure 1. Examples of popular sensor hardware using antennas with a limited number of emission power levels.

nodes should be adjusted such that the sum of the energy spent by all nodes is minimized. This problem is know as the *minimum energy broadcast* (MEB) in the literature [9]. To our knowledge, most—if not all—works from the literature use an antenna model where emission powers can be adjusted to any real value between zero and the maximum transmission range. However, available hardware such as SunSPOTs (see `http://www.sunspotworld.com/`) or iSense sensor nodes (see `http://www.coalesenses.com/`) are equipped with antennas that offer a limited set of different emission powers; 200 in the case of SunSPOTs and 5 in the case of the iSense hardware. Note that SunSPOTs are among the most widely used sensor hardware, while iSense nodes are used by two of the currently largest European projects on sensor networks, FRONTS and WISEBED.

In this paper we will adapt the current state-of-the-art algorithm for the MEB problem—proposed in [7, 6]—to the antenna model of SunSPOTs and iSense nodes (see Section 3). Afterwards we will show an experimental evaluation of the obtained algorithm in Section 4. Finally, we will offer conclusions and an outlook to future work in Section 5.

2. Minimum Energy Broadcasting with Realistic Antennas

Given a set of sensor nodes V, each node $i \in V$ can choose an emission power p_i such that $p_i \in P = \{tp_1, \ldots, tp_m\}$, where P is a finite set of m different emission powers such that $tp_1 = 0$ and $tp_l < tp_{l+1}$ for all $l = 1, \ldots, m-1$. Signal power diminishes at a rate proportional to $r^{-\alpha}$, where r is the distance to the signal source, and α is a parameter that, depending on the environment, takes typically values between 2 and 4.

In our work we choose $\alpha = 2$, as in most other works (see, for example, [9]). A sender node i is able to successfully transmit a signal to a receiver node j if $p_i \geq k \cdot d(i,j)^\alpha$, where $d(i,j)$ is the Euclidean distance between i and j, and k is the receiving node's power threshold for signal detection which is usually normalized to 1.

The minimum energy broadcast problem with realistic antennas (MEBRA), as introduced in the following, is NP-hard. This is because it is a generalization of the standard MEB problem as defined in the literature [3]. It can be stated as follows. Given is a set V of nodes with fixed positions in a 2-dimensional area. Introducing a directed link (i,j) between all (ordered) pairs $i \neq j$ of nodes such that $d(i,j)^\alpha \leq tp_m$, where $d(i,j)$ is the Euclidean distance between i and j, induces a directed network $G = (V,E)$. Given a source node $s \in V$, one must find emission powers for all nodes such that a broadcast from s to all other nodes is possible, and such that the sum of all emission powers is minimal. This corresponds to finding a directed spanning tree $T = (V, E_T)$ with root node s in G such that function $f()$ is minimized:

$$f(T) := \sum_{i \in V} \max_{(i,j) \in E_T} p_{ij} \tag{1}$$

where $p_{ij} \in P$ is the emission power necessary to reach node j from node i. Note that for all $i \in V$, $p_{ij} \in P$ is defined such that:

- $p_{ij} \geq d(i,j)^\alpha$
- Exists an $l \in \{1, \ldots, m\}$ such that $p_{ij} = tp_l \in P$ and $tp_{l-1} < d(i,j)^\alpha$.

Note that the emission power setting that corresponds to a solution T is obtained by setting $p_i := 0$ for all leaf nodes of T, and $p_i := \max_{(i,j) \in E_T} p_{ij}$ otherwise.

3. The Algorithm

As mentioned before, we present here the adaptation of the ant colony optimization (ACO) algorithm proposed in [7, 6] to the MEBRA problem. This algorithm is a $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System (MMAS) in the Hyper-Cube Framework [2], which works roughly as follows. At each iteration each of $n_a = 10$ artificial ants construct a tree rooted at the source node s. Local search is applied to each of these trees. The pheromone model $\mathcal{T}$ used by our ACO algorithm contains a pheromone value τ_e for each link $e \in E$. After the initialization of the variables T^{bs} (i.e., the best-so-far solution), T^{rb} (i.e., the restart-best solution), and cf (i.e., the convergence factor), all the pheromone values are set to 0.5.

Algorithm 1 ACO for the MEBRA problem

```
1: INPUT: the network G = (V, E) and a source node s ∈ V
2: T^bs := NULL, T^rb := NULL, cf := 0, bs_update := FALSE
3: forall e ∈ E do τ_e := 0.5 end forall
4: while termination conditions not satisfied do
5:    for j = 1 to n_a do
6:       T^j := ConstructBroadcastTree(G,s)
7:       T^j := LocalSearch(T^j)
8:    end for
9:    T^ib := argmin{f(T^1), ..., f(T^{n_a})}
10:   Update(T^ib,T^rb,T^bs)
11:   ApplyPheromoneValueUpdate(cf,bs_update,T,T^ib,T^rb,T^bs)
12:   cf := ComputeConvergenceFactor(T, T^rb, T^bs)
13:   if cf ≥ 0.99 then
14:      if bs_update = TRUE then
15:         forall e ∈ E do τ_e := 0.5 end forall
16:         T^rb := NULL, bs_update := FALSE
17:      else
18:         bs_update := TRUE
19:      end if
20:   end if
21: end while
22: OUTPUT: T^bs
```

At each iteration, after the generation of solutions, some of them are used for updating the pheromone values. The details of the algorithmic framework shown in Algorithm 1 are explained in the following.

ConstructBroadcastTree(G,s): A solution construction starts with the partial solution $S = (V_S, E_S)$ where $V_S := \{s\}$ and $E_S := \emptyset$. Remember that s is the source node of the directed spanning tree to be constructed. Henceforth we denote by $\overline{V_S}$ the set of nodes which are not included in the current partial solution, that is, $\overline{V_S} := V \setminus V_S$. At each construction step, one link (and one node) is added to the current partial solution. The set E_{add} of potential links that can be added to S is defined as follows: $E_{\text{add}} := \{(i,j) \in E | i \in V_S, j \in \overline{V_S}\}$. In words, E_{add} consists of those links whose source node is in S and whose goal node is not in S. From these links, one link is chosen according to the following probabilities:

$$\mathbf{p}(e) := \frac{\tau_e \cdot \eta(e)}{\sum_{e' \in E_{\text{add}}} \tau_{e'} \cdot \eta(e')} \ , \tag{2}$$

Algorithm 2 Variable neighborhood descent (VND)

1: INPUT: the network $G = (V, E)$, a source node $s \in V$, a spanning tree $T = (V, E_T)$ of G rooted in s, a parameter $r_{\max}$
2: $r := 1$
3: **while** $r \leq r_{\max}$ **do**
4: $T' := r$-shrink(T)
5: **if** $f(T') < f(T)$ **then** $T := T'$ and $r := 1$ **else** $r := r + 1$
6: **end while**
7: OUTPUT: a (possibly) improved tree T

where $\eta(e)$ is the heuristic information of a link $e = (i, j)$ which is computed as follows: $\eta(e) := p_{ij}^{-1}$. In other words, the heuristic information accounts for the increase of emission power. After choosing a link $e = (i, j)$ for the expansion of the current partial solution S, all the other links of E_{add} (if any) that can be added to S without any further increase of emission powers are also added to S (in addition to e). This concerns all links $e' = (i, l) \in E_{\text{add}}$ with $d(i, l) \leq p_{ij}$. Note that the solution construction stops when $\overline{V_S} = \emptyset$.

LocalSearch(T^j): Note that solutions constructed by the ants may contain nodes whose emission powers can be reduced without destroying the broadcast property of the solution. Therefore, we first apply the so-called SWEEP procedure (see [9]) in order to detect and fix these cases. Afterwards that variable neighborhood descent (VND) algorithm [5] outlined in Algorithm 2 is applied to further improve the given solution. VND is based on the local search procedure r-shrink, which was originally developed by Das et al. in [4] for the MEB problem. The only difference between our implementation and the one by Das et al. is the re-definition of the emission power p_{ij} necessary to reach node j from node i. While for the MEB problem, p_{ij} is defined as $d(i, j)^\alpha$, for the MEBRA problem this emission power is defined as in Section 2. After tuning by hand, parameter $r_{\max}$ was set to $|V| - 1$.

Update(T^{ib},T^{rb},T^{bs}): In this procedure T^{rb} and T^{bs} are set to T^{ib} (i.e., the iteration-best solution), if $f(T^{ib}) < f(T^{rb})$ and $f(T^{ib}) < f(T^{bs})$.

ApplyPheromoneUpdate(*cf*,*bs_update*,$\mathcal{T}$,T^{ib},T^{rb},T^{bs}): As usual, the proposed ACO algorithm may use three different solutions for updating the pheromone values: (i) the iteration-best solution T^{ib}, (ii) the restart-best solution T^{rb} and, (iii) the best-so-far solution T^{bs}. Their influence depends on the convergence factor cf, which provides an esti-

Table 1. The schedule used for values κ_{ib}, κ_{rb} and κ_{bs} depending on cf (the convergence factor) and the Boolean control variable *bs_update*.

	bs_update = FALSE			*bs_update* = TRUE
	$cf < 0.7$	$cf \in [0.7, 0.9)$	$cf \geq 0.9$	
κ_{ib}	2/3	1/3	0	0
κ_{rb}	1/3	2/3	1	0
κ_{bs}	0	0	0	1

mate about the state of convergence of the system. To perform the update, first an update value ξ_e for each link $e \in E$ is computed: $\xi_e := \kappa_{ib} \cdot \delta(T^{ib}, e) + \kappa_{rb} \cdot \delta(T^{rb}, e) + \kappa_{bs} \cdot \delta(T^{bs}, e)$, where κ_{ib} is the weight of T^{ib}, κ_{rb} the weight of T^{rb}, and κ_{bs} the weight of T^{bs} such that $\kappa_{ib} + \kappa_{rb} + \kappa_{bs} = 1.0$. The δ-function is the characteristic function of the set of links in a tree T, that is, $\delta(T, e) = 1$ if $e \in E(T)$, and $\delta(T, e) = 0$ otherwise. Then, the following update rule is applied to all pheromone values τ_e:

$$\tau_e := \min\left\{\max\{\tau_{\min}, \tau_e + \rho \cdot (\xi_e - \tau_e)\}, \tau_{\max}\right\} \quad ,$$

where $\rho \in (0, 1]$ is the learning rate, set to 0.1. The upper and lower bounds $\tau_{\max} = 0.99$ and $\tau_{\min} = 0.01$ keep the pheromone values always in the range $(\tau_{\min}, \tau_{\max})$, thus preventing the algorithm from converging to a solution. After tuning, the values for κ_{ib}, κ_{rb} and κ_{bs} are chosen as shown in Table 1.

ComputeConvergenceFactor($\mathcal{T}, T^{rb}, T^{bs}$): This function computes, at each iteration, the convergence factor as

$$cf := \frac{\sum_{e \in E(T^{rb})} \tau_e}{(|E(T^{rb})| - 1) \cdot \tau_{\max}} \text{ , respectively } cf := \frac{\sum_{e \in E(T^{bs})} \tau_e}{(|E(T^{bs})| - 1) \cdot \tau_{\max}} \quad ,$$

if *bs_update* = FALSE, respectively if *bs_update* = TRUE. Here, $\tau_{\max}$ is the upper limit for the pheromone values. The convergence factor cf can therefore only assume values between 0 and 1. The closer cf is to 1, the higher is the probability to produce the solution T^{rb} (or T^{bs} analogously).

4. Experimental Evaluation

We implemented our algorithm in ANSI C++ using GCC 3.2.2 for compiling the software. Our experimental results were obtained on

a PC with an AMD64X2 4400 processor and 4 GB of memory. We used a set of 30 problem instances with 50 nodes that was introduced in [1, 10] for the MEB problem. The 50 nodes per instances are randomly scattered over a square of 1000×1000. Given these area restrictions we decided to solve the MEBRA problem for antennas where $P = \{0, 100, 200, 300, 400, 500\}$. This corresponds, for example, to the antennas of iSense sensor nodes. After a simple adaptation to the MEBRA problem, we used the popular *broadcast incremental power* (BIP) algorithm presented in [9] as a benchmark algorithm for comparison.

We applied the ACO algorithm 30 times with a computation time limit of 20 seconds to each of the 30 problem instances. The results are shown in Table 2, in the following way. The first column provides the instance name. Then the results of three algorithms are provided. For BIP, which is a deterministic constructive heuristic, we provide the result and the computation time. Note that a computation time of 0.00 means that the algorithm was faster than 0.01 seconds. The second algorithm is BIP+VND, which is BIP with the subsequent application of VND as outlined in Section 3. In the case of BIP+VND we also provide the result plus the computation time. Moreover, we show the percentage improvement (labelled deviation) over the results of BIP. Note that negative percentages indicate an improvement. Finally, the results of ACO are given, for each instance, by the best result obtained in 30 runs (see column **best**), the average result over 30 runs (see column **average**), and the average computation time for reaching the best solution of each run. As in the case of BIP+VND, we also show for ACO the percentage improvement over BIP.

The results clearly show that BIP+VND is much better than BIP, and that ACO is clearly better than BIP+VND. On average BIP+VND is 12.78 percent better than BIP, whereas ACO is on average 33.99 percent better than BIP. This comes at the cost of an increased computation time. However, the average computation time of ACO for all instances is 4.28 seconds, which is not much time at all. Moreover, sensor networks are rarely much larger than 50 or 100 nodes. Therefore, the increased computation time is not a problem for practical purposes. Finally, in order to give the interested reader a flavor of the nature of the solutions obtained by ACO, the best solution obtained by ACO for instance p50.00 is shown in Figure 2.

5. Conclusions

In this work we have adapted an ant colony optimization published earlier for the minimum energy broadcast problem in wireless ad-hoc

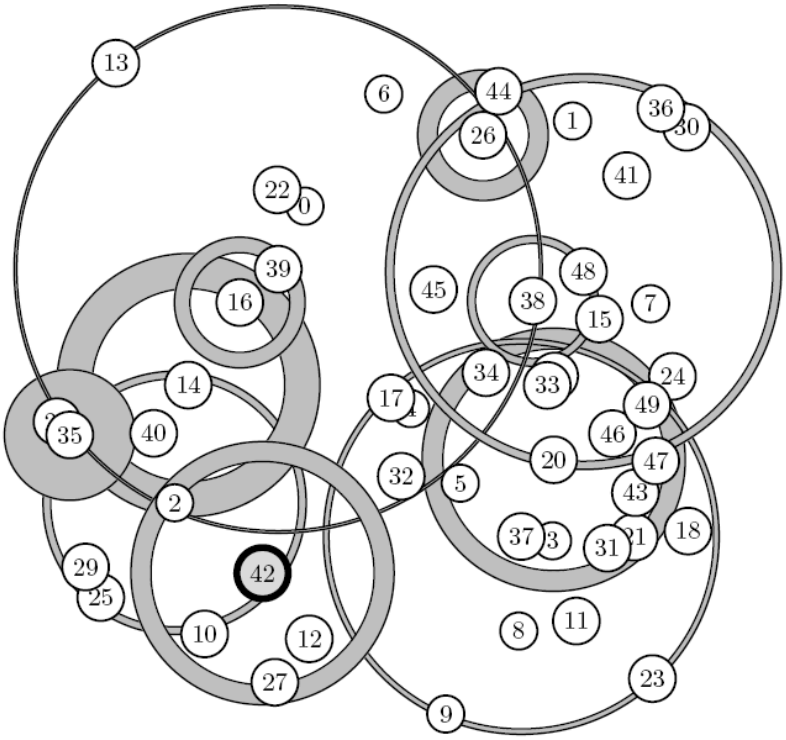


Figure 2. Instance: p50.00, required power: 540000.00. The circles show the emission powers of the individual sensor nodes. Gray shades rings show the "wasted energy" because the antenna model works with only 5 different power levels.

networks to the case of sensor networks with more realistic antennas. The results show that our algorithm is largely superior to an adapted popular constructive heuristic from the literature.

In the future we plan to do the following. It is important to realize that minimizing the sum of the emission powers of all sensor nodes for a single broadcast transmission does not necessarily imply a long network lifetime. Imagine, for example, a single node with a very high emission power. After repeated broadcast actions, this node will be the first one to run out of battery, which will make the whole network un-operational. Therefore, when network lifetime is concerned, it is important to balance the energy spending of the nodes. We are currently working on an extension of the algorithm in this direction.

Table 2. Results of BIP, BIP+VND, and ACO for all instances.

Instance	BIP		BIP+VND			ACO			
	result	time (s)	deviation	result	time (s)	best	deviation	average	time (s)
p50.00	780000.00	0.00	-17.95%	640000.00	0.08	540000.00	-30.68%	540666.67	4.33
p50.01	810000.00	0.00	-25.93%	600000.00	0.05	470000.00	-41.98%	470000.00	0.22
p50.02	820000.00	0.00	-3.66%	790000.00	0.05	490000.00	-40.24%	490000.00	2.16
p50.03	690000.00	0.00	-11.59%	610000.00	0.07	460000.00	-33.09%	461666.67	6.81
p50.04	640000.00	0.00	-7.81%	590000.00	0.06	420000.00	-34.06%	422000.00	7.39
p50.05	830000.00	0.00	-27.71%	600000.00	0.07	530000.00	-36.14%	530000.00	0.13
p50.06	740000.00	0.00	-8.11%	680000.00	0.07	470000.00	-36.22%	472000.00	8.54
p50.07	810000.00	0.00	-13.58%	700000.00	0.06	530000.00	-33.50%	538666.67	2.16
p50.08	740000.00	0.00	-12.16%	650000.00	0.06	420000.00	-40.99%	436666.67	5.99
p50.09	650000.00	0.00	-7.69%	600000.00	0.07	460000.00	-29.23%	460000.00	3.06
p50.10	800000.00	0.00	-8.75%	730000.00	0.08	550000.00	-30.75%	554000.00	5.84
p50.11	710000.00	0.00	-4.23%	680000.00	0.06	490000.00	-28.22%	509666.67	6.60
p50.12	820000.00	0.00	-23.17%	630000.00	0.08	540000.00	-34.15%	540000.00	3.52
p50.13	840000.00	0.00	-3.57%	810000.00	0.07	560000.00	-33.21%	561000.00	6.11
p50.14	970000.00	0.00	-11.34%	860000.00	0.05	540000.00	-44.30%	540333.33	1.46
p50.15	730000.00	0.00	-5.48%	690000.00	0.06	500000.00	-28.54%	521666.67	7.23
p50.16	810000.00	0.00	-12.35%	710000.00	0.07	560000.00	-30.86%	560000.00	4.05
p50.17	700000.00	0.00	-1.43%	690000.00	0.06	500000.00	-28.57%	500000.00	0.36
p50.18	870000.00	0.00	-27.59%	630000.00	0.07	500000.00	-42.49%	500333.33	6.47
p50.19	690000.00	0.00	-15.94%	580000.00	0.05	450000.00	-34.78%	450000.00	1.46
p50.20	840000.00	0.00	-20.24%	670000.00	0.09	580000.00	-29.80%	589666.67	0.63
p50.21	710000.00	0.00	-11.27%	630000.00	0.06	470000.00	-30.80%	491333.33	6.86
p50.22	660000.00	0.00	-9.09%	600000.00	0.07	440000.00	-30.96%	455666.67	8.00
p50.23	730000.00	0.00	0.00%	730000.00	0.06	560000.00	-21.05%	576333.33	5.92
p50.24	770000.00	0.00	-24.68%	580000.00	0.07	550000.00	-28.57%	550000.00	0.02
p50.25	770000.00	0.00	-7.79%	710000.00	0.06	430000.00	-44.16%	430000.00	2.66
p50.26	800000.00	0.00	-13.75%	690000.00	0.06	560000.00	-28.87%	569000.00	4.95
p50.27	890000.00	0.00	-33.71%	590000.00	0.08	540000.00	-39.10%	542000.00	3.60
p50.28	870000.00	0.00	-10.34%	780000.00	0.07	560000.00	-34.10%	573333.33	4.21
p50.29	820000.00	0.00	-2.44%	800000.00	0.04	480000.00	-40.33%	489333.33	7.72
	777000.00	0.00	-12.78%	675000.00	0.07	505000.00	-33.99%	510844.44	4.28

Acknowledgements

This work was supported by grant TIN2007-66523 (FORMALISM) of the Spanish government, and by the EU project FRONTS (FP7-ICT-2007-1). In addition, C. Blum acknowledges support from the *Ramón y Cajal* program of the Spanish Government, and H. Hernández acknowledges support from the *Comissionat per a Universitats i Recerca del Departament d'Innovació, Universitats i Empresa de la Generalitat de Catalunya* and from the *European Social Fund*.

References

[1] S. Al-Shihabi, P. Merz, and S. Wolf. Nested partitioning for the minimum energy broadcast. In *Proc. Learning and Intelligent Optimization (LION)*. Springer Verlag, Berlin, 2007.

[2] C. Blum and M. Dorigo. The hyper-cube framework for ant colony optimization. *IEEE T. Syst. Man Cy. B*, 34(2):1161–1172, 2004.

[3] M. Čagalj, J.-P. Hubaux, and C. Enz. Minimum-energy broadcast in all-wireless networks: NP-completeness and distribution issues. In *Proc. 8th Annual International Conference on Mobile Computing and Networking*, pages 172–182, 2002.

[4] A. K. Das, R. J. Marks, M. El-Sharkawi, P. Arabshahi, and A. Gray. r-shrink: A heuristic for improving minimum power broadcast trees in wireless networks. In *Proc. IEEE Globecom Telecommunication Conference*, pages 523–527, 2003.

[5] P. Hansen and N. Mladenović. Variable neighborhood search: Principles and applications. *Eur. J. Oper. Res.*, 130(3):449–467, 2001.

[6] H. Hernández and C. Blum. Ant colony optimization for multicasting in wireless ad hoc networks. *Swarm Intelligence*, 3(2):125–148, 2009.

[7] H. Hernández, C. Blum, and G. Francès. Ant colony optimization for energy-efficient broadcasting in ad-hoc networks. *Lect. Notes Comput. Sc.*, 5217:25–36, 2008.

[8] F. Schulz. Modeling sensor and ad hoc networks. *Lect. Notes Comput. Sc.*, 4621:21–36, 2007.

[9] J. E. Wieselthier, G. D. Nguyen, and A. Ephremides. On the construction of energy-efficient broadcast and multicast trees in wireless networks. *Proc. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, Vol. 2, pages 585–594, 2000.

[10] S. Wolf and P. Merz. Evolutionary local search for the minimum energy broadcast problem. *Lect. Notes Comput. Sc.*, 4972:61–72, 2008.

APPLICATION OF MEMETIC ALGORITHM IN PRODUCTION PLANNING

Peter Korošec, Gregor Papa, Vida Vukašinović
Computer Systems Department
Jožef Stefan Institute, Ljubljana, Slovenia
{peter.korosec; gregor.papa; vida.vukasinovic}@ijs.si

Abstract Today, many algorithms are developed, evaluated, and compared on test benchmark problems. Their drawback is that they can simulate real-world problems up to some degree. Often, it turns out that there are many specifics of the problem we are trying to solve. To make algorithms efficient, constraints need to be considered and included in the problem solving. In this paper a real-world production planning problem is addressed. A typical approach with genetic algorithm turned out to be insufficient due to complexity with many constraints. To successfully solve this problem, a memetic algorithm, which uses specialized local searches to improve solutions acquired by genetic algorithm, is proposed. It is shown, that the use of specialized local searches can significantly improve the convergence and efficiency of the algorithm.

Keywords: Application, Combinatorial optimization, Memetic algorithm, Scheduling problem

1. Introduction

Today's complex manufacturing processes have multiple types of products, each requiring many different steps, production processes, and product parts for completion. The job of the expert for the manufacturing plan is to find a way to successfully manage resources in order to produce products in the most efficient way possible. The expert must design a production schedule, primarily such that it satisfies on-time delivery and minimizes production costs in terms of time needed to complete the production process. There are also many specific constraints that need to be considered. The main problem is the exchange delay, caused by adapting production lines to different types of products and

supplying the appropriate parts. Such problem is a typical member of the family of job scheduling problems [3].

To solve the job scheduling problem there are many scheduling methods reported in the literature. They are all useful and efficient in some aspect. One of the approaches is the use of genetic algorithm (GA). A heuristic for the open job shop scheduling problem using GA to minimize makespan is developed in [16]. On the other hand a scheduling method based on GA is developed considering multiple criteria in [5]. Other implementations of GA for job scheduling can be found in [18]. Furthermore, memetic algorithms (MAs) [14] represent a synergy of evolutionary (or any population-based) approach with separate individual learning or local improvement procedures for problem search. Various MAs are developed [4, 9, 13] to obtain even better results than GA for various job scheduling applications. With the use of local search techniques the results are further improved. MAs do not only improve the quality of solutions but also reduce the overall computational time [9].

The rest of the paper is organized as follows: in Section 2 we formally define the problem; in Section 3 we describe the approach used for the search of the best schedule; in Section 4 we show the user interface; in Section 5 we describe the experimentation environment with results; and in Section 6 we list our conclusions.

2. Planning Problem

As shown earlier our planning problem can be represented as job scheduling problem. Job scheduling problem is NP-hard [3]. In general, a job scheduling problem can be described in a following manner. Let assume we have a finite set of n jobs, where each job consists of a chain of operations. Then we have a finite set of m machines, where each machine can handle at most one operation at a time. Here each operation needs to be processed during an uninterrupted period of a given length on a given machine. The purpose is to find a schedule, that is, an allocation of the operations to time intervals to machines, that has overall minimal length.

More formal definition would be as follows. Let assume job set $J = \{j_1, j_2, \ldots, j_n\}$ and a machine set $M = \{m_1, m_2, \ldots, m_m\}$ with operations $O = \{O_1, O_2, \ldots, O_n\}$, where $O_i = \{o_{i1}, o_{i2}, \ldots, o_{iq_i}\}$,and q_i is the number of operaitons in the chain O_i, $(i = 1, 2, \ldots, n)$. Here, each operation has its processing time $\{\tau_{i1}, \tau_{i2}, \ldots, \tau_{iq_i}\}$. The goal is to find an optimal feasible schedule with minimum length.

Scheduling problems involve search for the optimal schedule under various objectives, different machine environments and characteristics of the jobs.

Our problem has additional constraints, which need to be considered. Machines M (production lines) have different capabilities. Each machine has its own time schedule (when it is operational). Each job has its own deadline, which must be met. In addition it can be done only on some machines and on each of them the speed of manufacturing process is different. Between jobs there can be an exchange delay, which depends on previous job and machine used. There is also a stock of finished products. If a job consists of these products, than it does not need to be produced. We can use the stock.

Taking into account the above mentioned constraints the task is to find the schedule, that minimizes the number of delayed orders (produced after deadline), exchange delays and time to finish all the orders.

To tackle this problem we developed a adapted MA.

3. Memetic Algorithm

Our search approach bases on GA [1, 8], which is a population-based evolutionary approach. It was chosen due to algorithm's intrinsic parallelism that allows simultaneous search within a broad database of solutions in a search space. The risk of converging to a local optimum exists, but efficient results of various researches on different optimization problems [7, 10, 11, 15, 17] encouraged us to consider GA approach. We developed our own version of GA, to fully adapt it to the specific problem of production planning. It is further enhanced with local search procedures to improve the results of the search, which transforms GA to MA. Such MA [14] possesses the ability of GA to find a good (near-optimal) solution in a reasonable time, and the power of local search to move quickly from the near-optimal solution to the optimal one. The details of the MA, based on the directions in [2, 12] and the local search procedures, is presented below and described in the following sections.

3.1 Production Schedule Encoding

The production schedule is encoded into one chromosome with a tuples of values. Each tuple (gene) consists of the index of the enumerated order and the production line.

Based on the given list of all orders, which are sorted according to the deadlines, various orders of indexes that represent the given order are encoded in chromosome. A chromosome, which includes encoded

Algorithm 1 Memetic Algorithm

```
SetInitialPopulation(P)
Evaluate(P)
while not EndingCondition() do
   ReproduceBetterSolutions(P, r)
   Crossover(P, p_c)
   Mutation(P, p_m)
   if LSenable then
      LocalSearch()
   end if
   Evaluate(P)
end while
```

production schedule of n orders, looks like

$$C = i_{11}i_{12}i_{21}i_{22}\ldots i_{j1}i_{j2}\ldots i_{n1}i_{n2}, \tag{1}$$

where i_{j1} represents an index of the order, i_{j2} represents the production line for that order, and $j = 1, 2, \ldots, n$.

3.2 Population Initialization

The initial population P consists of N chromosomes. In each chromosome the orders are randomly distributed, and also the assigned production line is chosen randomly among possible lines for each order.

Since the numbers in the chromosome represent the indexes of orders their values can not be duplicated and no index can be missed; therefore both conditions must be considered during the initialization.

3.3 Genetic Operators

The elitism strategy is used in order to prevent losing the best found solution by memorizing it.

The substitution of the least-fit chromosomes with the equal number of the best-ranked chromosomes ensures better solutions to have more influence on the new generation. The ratio of all chromosomes in the population to be replaced is set by the ratio r.

The Crossover (P, p_c) is performed with the interchange of positions that store the ordered indices within the range (order-based crossover). The order-based crossover randomly takes a part of two parents, swaps genes of the parents in this part and orders the remaining genes in the first parent in accordance with its order in the second parent. During the optimization procedure four types of order-based crossover are used

and are switched every 10 generations. The implemented crossovers are order (OX), cycle (CX), partially-mapped (PMX) and PTL crossover. OX, CX and PMX are more precisely explained in [12], while PTL is explained in [6]. In OX, PMX and PTL two-point crossover scheme is used, where chromosome mates are chosen randomly.

In the mutation process Mutation(P, p_m) each value of the chromosome mutates with a mutation probability p_m. However, since a high mutation rate results in a random walk through search space, p_m has to be low. Three different types of mutation are applied: change of production line; switching of two genes in the chromosome; and shifting of a gene into some new position. Notice, that shifting of a gene into some new position has an effect on larger part of chromosome.

Each new population is generally better than the previous one. To overcome a possible disruptive effect of mutation at the later stages of the optimization and speed up the convergence to the optimal solution in the final optimization stages, crossover and mutation probability are being decreased with each new restart of the algorithm. Moreover, the lower number of mutated positions in the later stages presents some kind of a local search with minor movements around current solution, i.e. exploitation.

3.4 Local Search

Local search is enabled when new best solution is found. It locally optimizes the new best solution. Additionally, at every 300-th generation it randomly selects 10% of solutions from population P, and locally optimizes them. Local search is implemented with four different procedures, which run sequentially in the following order:

- Stock replacing. For each order filled from the stock it is checked, if the order for the same product is scheduled for the production. If the order, to be produced, is delayed, than this order is shifted in front of the order from the stock, which was checked. In this case some orders of the same product are possibly moved out of the stock and placed into the production. If the number of the delayed orders is increased, then the previously shifted order is moved back to its previous position.

- Deadline sorting. For each production line, all orders with delayed deadlines are checked if they can be moved before some other order. The delayed order is moved before each of the precedent order, so that it is not delayed anymore, while ensuring that the total number of delayed orders is not increased.

- Production line changing. For each production line, all orders are checked if they can be placed on any other feasible production line. If they can be placed on some other production line, then it is further checked if they can be merged with some other similar order on that new production line. The switch to another production line should not increase the exchange delay on the new production line.

- Similar product merging. For each production line, it is checked if several orders can be merged together. The merging is performed in four steps, according to different properties of the products. First the orders for the products with the same height are merged, then those with the same size are merged, after that the orders with the same connectors, and finally those with the same power characteristics. The idea of merging procedure is to decrease the production time on each line, as result of decreased exchange delay on the line.

3.5 Fitness Evaluation

After the variation operators modify the solutions, the modified part of new population is ready to be evaluated. Here the evaluator is used to evaluate solution according to number of delayed orders (n_{orders}), exchange delay times in minutes (t_{exchange}), overall production time in minutes (t_{overall}), and number of days of delayed orders (n_{days}). The cost function, which is calculated inside Evaluate(P) was proposed according to experts requirements and is set as follows:

$$f(P) = 10^7 \cdot n_{\text{orders}} + 10^4 \cdot t_{\text{exchange}} + t_{\text{overall}} + n_{\text{days}}^2.$$

From the cost function it is obvious that the most important item to minimize is n_{orders}, then t_{exchange} and lastly t_{overall} and n_{days}. The factors besides these items make sure that the first two digits of evaluation function value represent number of delayed orders, next three digits represent exchange delay times in minutes and the last digits represent the influence of t_{overall} and n_{days}.

3.6 Ending Condition

The EndingCondition() function checks if a certain number of generations are without improvement. When that occurs the system is assumed to be in a steady state, and the optimization ended. The currently best solution is chosen as a final result.

4. User Interface

When the algorithm is implemented for an industrial problem, we have to be aware of many other aspects/components besides the algorithm's performance. The program will be used by engineers and other staff on a daily basis. Therefore, it is important to make our optimization program usable and user friendly without having the background knowledge of the optimization algorithm. Since the end user requested that the application is accessible from a web browser, we have chosen Java as our implementation language.

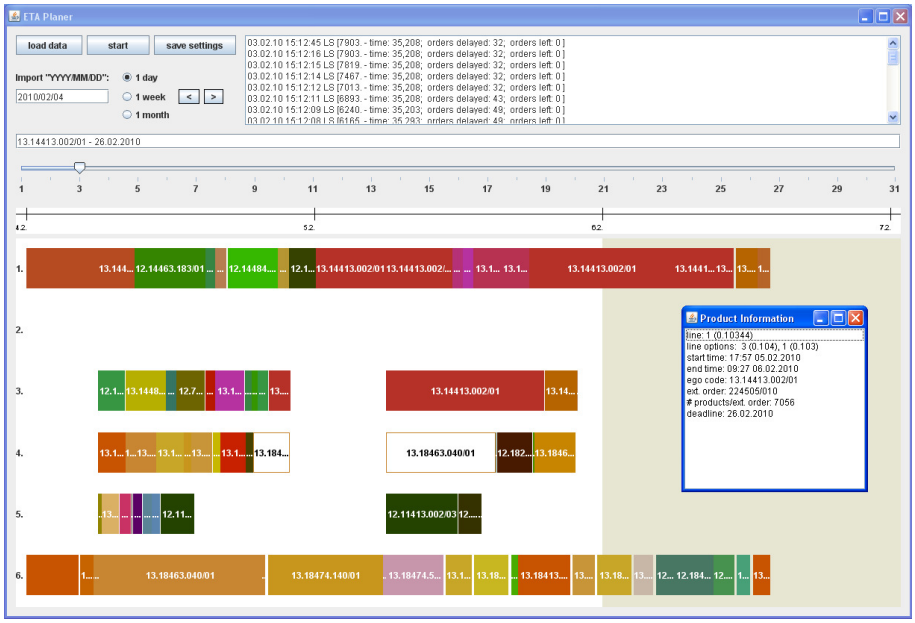


Figure 1. The graphical user interface in Java.

Graphical user interface (GUI) is done as a java application (see Figure 1), which can be easily modified to run from any web browser on any platform. The GUI provides a consistent appearance and intuitive controls like push buttons, text fields, sliders, and so forth. The GUI behaves in an understandable and predictable manner.

The algorithm optimizes the schedule of given list of orders. Every time a new batch of orders is added to the list, the algorithm is being rerun by the user. The GUI enables the user to visually observe the simulation of the current optimal production schedule. By pushing the start button the problem is loaded and the algorithm starts. The current best schedule is drawn on the panel in the lower part of the GUI. The

schedule refreshes every time the algorithm finds a better solution. It stops when the user interrupts the optimization process by pushing the stop button. During the optimization process, the information about the currently best solutions is displayed in a text list in the upper right corner of the GUI. The schedule is visualized by production lines with products shown as colored boxes. Each product box has its own color and the size corresponding to the duration of its production process. The product information window opens with a mouse click on the product box. It displays information on production line, production start and end time, deadline, and other production properties. The products, which exceed the deadline, are colored white with only borders in proper color. To emphasize the fact that the production line exchange delay downtime is minimized similar products are represented with different shades of the same basic color. The slider above the time scale enables the user to visualize up to 31 day schedule at once. The GUI also enables a moving forward or backward through the schedule.

5. Performance Evaluation

5.1 The Experimental Environment

The computer platform used to perform the experiments is based on AMD Athlon II ™ 2.9-GHz processor, 4 GB of RAM, and the Microsoft® Windows® 7 operating system. The MA is implemented in Sun Java 1.6.

5.2 The Test Cases

The GA and MA algorithms were tested on two different real order lists from a production company. The first task (Task 1) consists of $n = 711$ orders for 251 products, while the second task (Task 2) consists of $n = 737$ orders for 262 products. Number of orders n represent the problem dimension. In both tasks $m = 5$ production lines are available. Each product can only be put on some production lines (depending on the product characteristics).

We made 30 runs with each algorithm (GA and MA). The run time of optimization is approximately 10 minutes for $1,000,000$ number of evaluations.

5.3 Parameter Settings

We must note that during the experimentation we did not fine-tune the algorithms parameters, but only carried out a limited number of experiments to find satisfying settings.

The following parameters were used:

- the population size $N = 50$;
- the number of generations depends on the progress of the search. After there is no improvement for $1,000$ generations the search is restarted, and lasts until the number of evaluations limit is reached. With each restart the crossover and mutation probabilities were decreased by 50% of the current value;
- the replacement rate $r = 0.2$;
- the crossover probability $p_{\mathrm{c}} = 0.7$;
- the mutation probability $p_{\mathrm{m}} = 0.005$.

5.4 Results

In Tables 1 and 2 best, mean, worst, and standard deviations of solutions are presented. It is obvious that MA achieves consistently better solutions than GA in both tasks. From Table 1 it is seen that the mean solution obtained by GA consists of 22 delayed orders, while the worst solution obtained by MA consists of only 13 delayed orders for Task 1. The difference between the worst solution obtained by GA and MA are even larger and the standard deviation obtained by GA is much larger than standard deviation obtained by MA. The results of optimization for Task 2 are similar to the results of optimization for Task 1 according the difference between best, mean, worst and standard deviation of solutions of both algorithms.

Table 1. Results of optimization for Task 1

Algorithm	GA	MA
Best	1.312×10^8	1.308×10^8
Mean	2.253×10^8	1.340×10^8
Worst	5.216×10^8	1.610×10^8
StD	1.015×10^8	6.526×10^6

To show how each of the components of the cost function behaves during the search process, we present Figures 2 and 3. Here, only changes to the evaluation function variables (n_{orders}, t_{exchange}, t_{overall}, and n_{days}), when new better solution was found, are shown. It can be visually seen that the MA has better convergence with much more consistent and stable behavior than GA. Note that local search, used by MA, requires around 30% of all evaluations.

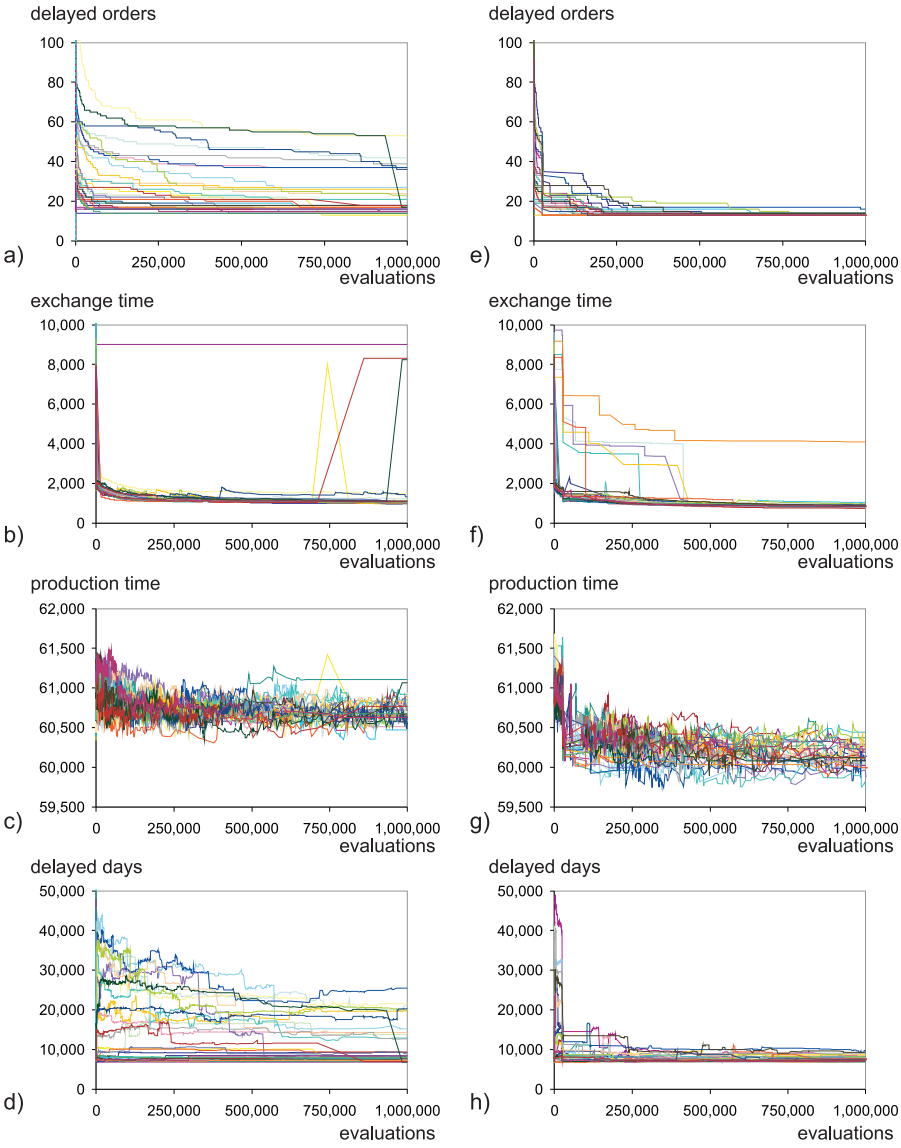


Figure 2. Task 1: a) n_{orders}, b) t_{exchange}, c) t_{overall}, and d) n_{days} for GA and e) n_{orders}, f) t_{exchange}, g) t_{overall}, and h) n_{days} for MA.

When comparing with the previous approach of production planning, the expert's manual plan for those two tasks had 53 delayed orders in Task 1 and 61 delayed orders in Task 2. This is significantly worse than results obtained by both of the algorithms. The details are presented in Table 3.

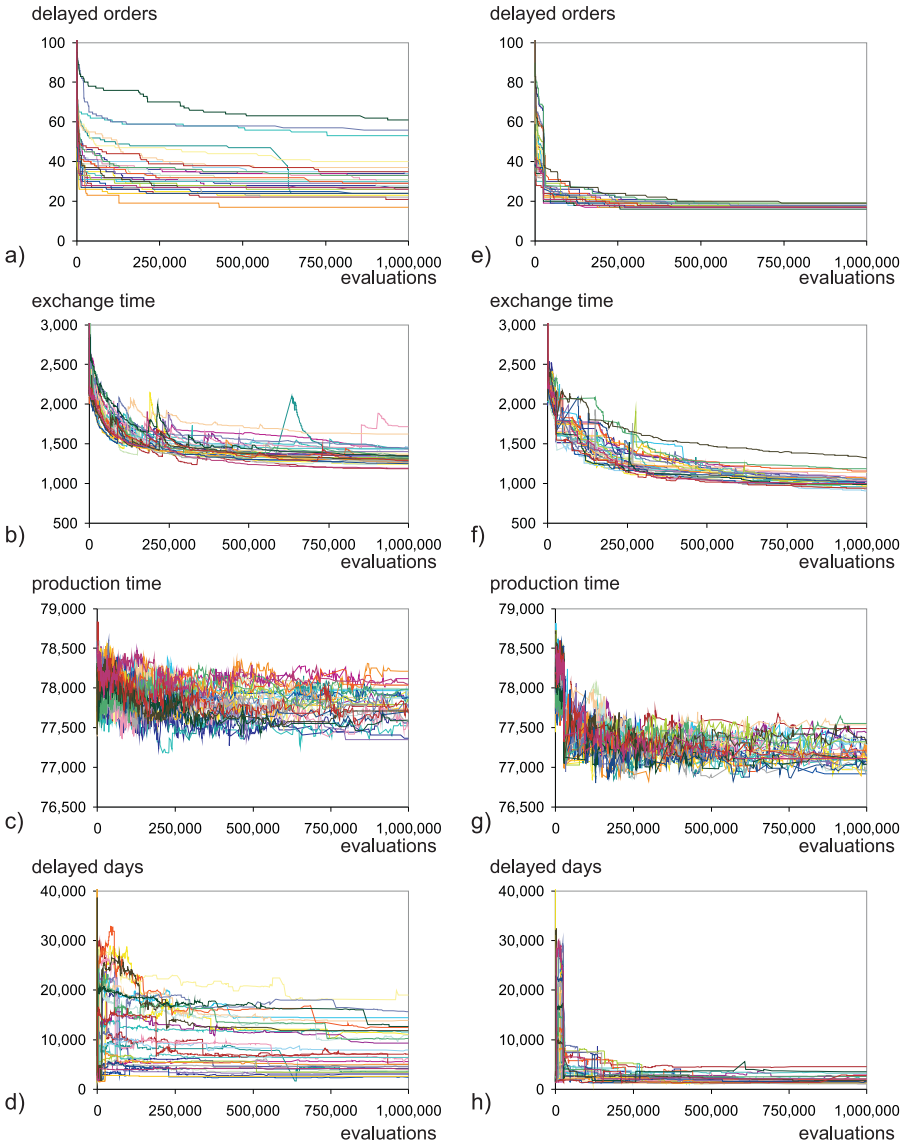


Figure 3. Task 2: a) n_{orders}, b) t_{exchange}, c) t_{overall}, and d) n_{days} for GA and e) n_{orders}, f) t_{exchange}, g) t_{overall}, and h) n_{days} for MA.

6. Conclusions and Future Work

In this paper, we have shown an application of specialized memetic algorithm on a real-world production planning problem. Since such a problem is a member of the family of job scheduling problems, which are

Table 2. Results of optimization for Task 2

Algorithm	GA	MA
Best	1.714×10^8	1.611×10^8
Mean	3.118×10^8	1.748×10^8
Worst	6.114×10^8	1.914×10^8
StD	1.009×10^8	7.235×10^6

Table 3. Comparison of delayed orders

	Task 1			Task 2		
	manual	GA	MA	manual	GA	MA
Best		13	13		17	16
Mean	53	22	13	61	31	17
Worst		42	16		56	19

known to be NP-hard, we have decided to use stochastic optimization approach called genetic algorithm (GA). Due to the problem complexity with many constraints it turned out that this is not sufficient. So we added some specialized local searches and therefore made the memetic algorithm (MA).

We have showed, that the use of both stochastic approaches greatly improved the quality of production plans in respect to expert's manual solution. Between the GA and MA was also a noticeable difference in favor of the MA. The convergence and efficiency of the algorithm has drastically improved with the use of specialized local searches.

For future work, we are planning to add some more constraints, like fixed order production days (the day when order has to be carried out), etc. Another aspect of improvement will be to make the application work on a client-server base. This means that the optimization algorithm will run on a host server and the results can be accessed from any local client, which is connected to the server.

References

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

[2] T. Bäck, D. B. Fogel, and Z. Michalewicz. *Evolutionary Computation 1: Basic Algorithms and Operators*. Taylor & Francis Group, 2000.

[3] P. Brucker. *Scheduling Algorithms, 2nd edition*. Springer, Heidelberg, 1998.

[4] A. Caumond, P. Lacomme, and N. Tchernev. A memetic algorithm for the job-shop with time-lags. *Comput. Oper. Res.*, 35(7):2331–2356, 2008.

[5] G. Chryssolouris and V. Subramaniam. Dynamic scheduling of manufacturing job shops using genetic algorithms. *J. Intell. Manuf.*, 12(3):281–293, 2001.

[6] J. Czogalla and A. Fink. On the Effectiveness of Particle Swarm Optimization and Variable Neighborhood Descent for the Continuous Flow-Shop Scheduling Problem. In *Studies in Computational Intelligence*, 128:61–89, 2008

[7] T. Garbolino and G. Papa. Genetic algorithm for test pattern generator design. *Appl. Intell.*, In press, doi:10.1007/s10489-010-0214-7.

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

[9] S. M. Kamrul Hasan, R. Sarker, D. Essam, and D. Cornforth. Memetic algorithms for solving job-shop scheduling problems. *Memetic Comp.*, 1(1):69–83, 2009

[10] P. Korošec, J. Šilc. Using stigmergy to solve numerical optimization problems. *Comput. Inform.*, 7 (3):377–402, 2008.

[11] B. Koroušić Seljak. Computer-based dietary menu planning. *J. Food Compos. Anal.*, 22(5):414–420, 2009.

[12] Z. Michalewicz and D. B. Fogel. *How to Solve It: Modern Heuristics.* Springer-Verlag, 2004.

[13] B. M. Ombuki, and M. Ventresca. Local search genetic algorithms for the job shop scheduling problem. *Appl. Intell.*, 21(1):99–109, 2004.

[14] Y. S. Ong and A. J. Keane. Meta-Lamarckian learning in memetic algorithms. *IEEE T. Evol. Comput.*, 8(2):99–110, 2004.

[15] G. Papa and B. Koroušić Seljak. An artificial intelligence approach to the efficiency improvement of a universal motor. *Eng. App. Artif. Intel.*, 18(1):47–55, 2005.

[16] P. Senthilkumar and P. Shahabudeen. GA based heuristic for the open job shop scheduling problem. *Int. J. Adv. Manuf. Techn.*, 30(3.4):297–301, 2006.

[17] T. Tušar, P. Korošec, G. Papa, B. Filipič, and J. Šilc. A comparative study of stochastic optimization methods in electric motor design. *Appl. Intell.*, 27(2):101–111, 2007.

[18] M. Vazquez and L. D. Whitley. A Comparison of Genetic Algorithms or the Static Job Shop Scheduling Problem', *Lect. Notes Comput. Sc.*, 1917:303–312, 2000.

DAYS-OFF SCHEDULING FOR A BUS TRANSPORTATION STAFF

Jari Kyngäs, Kimmo Nurmi
Satakunta University of Applied Sciences, Pori, Finland
{jari.kyngas; kimmo.nurmi}@samk.fi

Abstract Staff scheduling has become increasingly important for both public sector and private companies. Good rosters have many benefits for an organization, such as lower costs, more effective utilization of resources and fairer workloads and shifts. The construction of optimized days-off for the personnel is an important part of the process. This paper presents a successful way to schedule days-off for the staff of a Finnish bus transportation company. The algorithm is a variation of the cooperative local search method. The generated software is currently in use in the company.

Keywords: Metaheuristics, Real-world scheduling, Staff scheduling

1. Introduction

Many new timetabling problems and algorithms have been introduced in recent years. Still, most of the timetabling research concentrates on educational timetabling, staff scheduling and sports scheduling. Staff scheduling is the process of constructing optimized work timetables for the personnel. Different variations of the problem are NP-complete [2, 16, 13, 17] and thus extremely hard to solve. The first mathematical formulation of the problem based on a generalized set covering model was proposed by Dantzig [10]. A good overview of staff scheduling can be found in [12]. Nurse rostering [7] is by far the most studied application area of staff scheduling. Other successful application areas include airline crews [11], call centers [4], postal services [1] and transport companies [20]. Because of its economic scale, airline crew scheduling is probably the most celebrated application area of staff scheduling. Most of the cases in which academic researchers have announced that they have closed a contract with an organization concern nurse rostering (see e.g. [3] and [6]). This paper presents a new case: scheduling days-off for

the staff of a Finnish bus transportation company. To the best of our knowledge, this is one of the few papers focusing on days-off scheduling.

There are basically four reasons for the current interest in staff scheduling. First, public institutions and private companies around the world have become more aware of the possibilities of decision support technologies, and they no longer want to handle the schedules manually. Second, human resources are one of the most critical and most expensive resources for these organizations. Careful planning can lead to significant improvements in productivity. Third, good schedules are very important for the welfare of the staff. Besides increasing employee satisfaction, effective labor scheduling can also improve customer satisfaction. Finally, new algorithms have been developed to tackle previously intractable problem instances, and, at the same time, computer power has increased to such a level that researchers are able to solve real-world problems. One further significant benefit of automating the scheduling process is a considerable time-saving for the administrative staff involved.

The focus of this paper is to solve a constrained days-off scheduling problem. In Section 2 we define a staff scheduling problem and introduce the necessary terminology. Section 3 presents the requirements and the requests of the days-off scheduling problem and Section 4 details a days-off scheduling problem in one of the Finnish bus transportation companies. Section 5 presents our solution method. In Section 6 we describe the difficulty of the process of consulting with the problem owner. Finally, in Section 7, we propose a set of test instances that we hope the researchers of the days-off scheduling problem will adopt. It will be seen that our solutions to the real-world problem and for the test instances are competitive.

2. Staff Scheduling

The staff scheduling problem has a fairly broad definition. Most of the studies focus on assigning employees to shifts, determining working days and rest days or constructing flexible shifts and their starting times.

Staff scheduling consists of assigning *employees* to tasks over a period of time according to a given timetable. The *planning horizon* is the time interval over which the employees have to be scheduled. Each employee has qualifications and skills that enable her/him to carry out certain tasks. A *skill category* determines a group of employees who have a particular level of qualification. Days are divided into working days (*days-on*) and rest days (*days-off*). A sequence of working days is called a *work stretch*. Working days consist of *shifts*. A shift is a contiguous set of working hours and is defined by a starting period and day along with

a *shift length* in periods. Each shift is composed of a number of tasks. A specific sequence of shifts for an employee is called a *stint*. A work schedule for an employee over the planning horizon is called a *roster*. A roster is a combination of shifts and days-off assignments that covers a fixed period of time.

Cyclic schedules are such that all employees have the same basic schedule but start with a different day. In cyclic scheduling the goal is to find a schedule that is optimal for all employees. *Non-cyclic* schedules are individual. In non-cyclic scheduling the goal is to find rosters that fulfill the requests of most employees. *Continuous* schedules arise in organizations that operate 24 hours a day and seven days a week, otherwise a schedule is called discontinuous.

Table 1. An example of a staff scheduling problem

		M1	M2	M3	W1	W2	M4	M5
Mon	Day	1	2			3		
	Night			3			1	2
Tue	Day	1	2			3		
	Night			1	3		2	
Wed	Day	1				3	2	
	Night		1		3			2
Thu	Day			3		1	2	
	Night	1		2		3		
Fri	Day	1		3		2		
	Night	1		2				3
Sat	Day		1	3			2	
	Night		2			3	1	
Sun	Day	1		2	3			
	Night		2			3		1

Table 1 shows a solution for a one-week staff scheduling problem with seven employees (five men and two women), two shifts in a workday (day and night) and three tasks to be carried out within a shift. In addition, tasks one and two can only be carried out by men, and each employee should have one day-off.

A good classification of a staff scheduling process is given in [12]. Figure 1 shows a modified version of their presentation. The first thing is to determine how many employees are needed at different times over some planning horizon. Demand modeling is the process of determining

planning horizons, the shift structure, number of employees needed at different times, tasks to be carried out in particular shifts and the level of qualification needed in different shifts. The output is a mathematical model of the problem at hand.

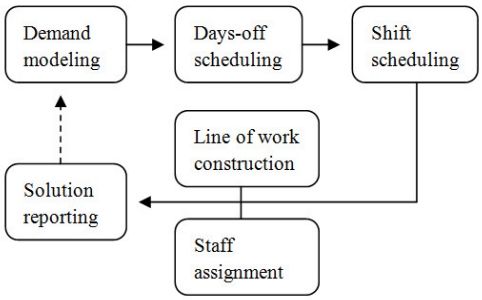


Figure 1. A staff scheduling process.

Days-off scheduling deals with the assignment of rest days between working days to employees over a given planning horizon. *Shift scheduling* deals with the assignment of employees to shifts. It can also specify the starting time and duration of shifts for a given day; that is, days-off scheduling deals with working days and shift scheduling with the working times of day. When days-off and shifts are scheduled simultaneously, the process is called *tour scheduling*. The name comes from the fact that we need to specify the hours of the day and days of the week through which each employee must travel. *Line of work construction* ensures the feasibility of each employee's roster. It also ensures that all the rosters together satisfy the work requirements at all times in the planning horizon. *Staff assignment* involves the assignment of individual employees to the rosters. In the case of cyclic schedules, this is usually performed manually. In the case of non-cyclic schedules, line of work construction and staff assignment are often done during shift scheduling. We adopt this convention in this paper. Finally, a *reporting* tool should display *solutions* and provide performance measures in such a way that a user can easily evaluate his or her level of satisfaction. When necessary, the demand modeling can be reprocessed and focused, and the whole staff scheduling process restarted. We will see an example of this in Section 6.

3. Requirements and Requests

In this section we will outline the typical constraints of a days-off scheduling problem. The hard and soft constraints of the problem vary somewhat depending on the problem instance at hand. However, in most

cases the hard constraints consist of coverage, regulatory and operational requirements and the soft constraints consist of operational and personal preferences (see e.g. [5]). The coverage requirement ensures that there are a sufficient number of employees on duty at all times. The regulatory requirements ensure that the employee's work contract and government regulations are respected. The personnel's requests are very important and should be met as far as possible; this leads to greater staff satisfaction and commitment, and reduces staff turnover. An organization can use a mixture of the following requirements and preferences as a framework for its days-off schedule generation:

Coverage requirement

(C1) A minimum number of employees must be guaranteed for each shift.

Regulatory requirements

(R1) The number of working days and days-off within a timeframe must be respected.

(R2) The number of holidays within a year must be respected.

(R3) The number of special days for particular employees within a timeframe must be respected.

(R4) Employees cannot work consecutively for more than k days (the maximum length of a workstretch).

(R5) Some employees cannot work at weekends.

Operational requirements

(O1) At least k working days must be assigned between two separate days-off.

(O2) A balanced assignment of weekdays must be guaranteed between employees.

(O3) A balanced number of surplus employees must be guaranteed for each working day.

Operational preferences

(E1) Single days-offs should be avoided.

(E2) The maximum length of consecutive days-off is k.

(E3) A balanced assignment of single days-off and single working days must be guaranteed between the employees.

Personal preferences

(P1) Try to assign given employees to the same shifts and try to avoid assigning another given employees to the same shifts.

(P2) Try to assign a requested day-on or avoid a requested day-off.

4. The Problem in a Finnish Bus Transportation Company

In our previous studies we have successfully scheduled the Finnish major ice hockey league [14] and the Finnish 1st division ice hockey league [15]. When Turku Transport Services Ltd. heard that we had scheduled these leagues, they contacted us. Turku Transport Services Ltd. is a bus transportation company in the City of Turku. They have currently 58 full-time, 8 part-time and 4 retired-but-still-active bus drivers. Two part-time drivers count as one full-time driver.

Prior to the year 2010, rosters for bus drivers were produced by a cyclic shift scheduling software which was quite out-of-date. The current number of employees and the current way of doing business had grown beyond the limits of the current system. The old system had led to an oversupply of bus drivers with too much idle time. Furthermore, the old system also required far too much manual work. For example, the days-off scheduling was done completely manually.

Staff scheduling is quite easy in many companies because the shifts are invariant and no work takes place on weekends. But when the amount of work varies from day to day and from week to week, and the company runs on every day, the scheduling process can be very complex and difficult. That is the case at Turku Transport Services Ltd. Their scheduling problem is non-cyclic and discontinuous, and can be divided into two separate sub-problems: days-off scheduling and shift scheduling. We concentrate on the days-off scheduling problem.

Table 2. The minimum number of employees needed on different days of week

	Mon	Tue	Wed	Thu	Fri	Sat	Sun
Employees	47	47	47	47	50	27	10

Suppose we have n (62) employees. Over the planning horizon of one year (13 timeframes with a timeframe of 4 weeks totaling 364 days), we

must find n sequences of days-on and days-off that satisfy the following hard constraints

(C1) A minimum number of employees must be guaranteed for each working day (see Table 2).

(R1) Each employee should have 9 days-off in every 4-week timeframe.

(R4) Employees cannot work consecutively for more than 6 days.

(R5) Six employees cannot work at weekends.

(O1) At least 2 working days must be assigned between two separate days-off.

(O2) The number of days-off per weekday between employees should not differ by more than 10%.

(O5) A balanced number of surplus employees must be guaranteed for each working day.

(P1) Exactly the same sequence for three employee groups with three employees in each group must be guaranteed.

and the following soft constraints

(E1) Single days-off should be minimized (one violation for each single day-off).

(E2) The maximum length of consecutive days-off is three (one violation for each day more than three).

(E3) The number of single days-off and single working days between employees should not differ by more than 25% (one violation for each unit of percentage over 25).

The company has more employees working than is needed to cover the minimum number of employees each working day. The surplus employees are used to cover the expected sick days. In addition, retired-but-still-active drivers can be used if necessary. The average number of surplus employees is calculated as follows. The number of man-days that is needed over the planning horizon is given as

$$man\_needed = pm \sum_{i=1}^{7} w_i,$$

where p is the number of timeframes, m is the number of weeks in the timeframe and w_i is the number of employees needed on weekday i

(see Table 2). Respectively, the number of man-days available over the planning horizon is given as

$$man\_available = np(7m - d),$$

where n is the number of employees and d is the number of days-off that must be respected within each timeframe (see **R1**). The average number of surplus employees is now given as

$$avg\_surplus = \frac{man\_available - man\_needed}{7pm}.$$

In this case, $man\_needed = 13 \times 4 \times 275 = 14300$, $man\_available = 62 \times 13 \times (7 \times 4 - 9) = 15314$ and $avg\_surplus = (15314 - 14300)/364 \approx 2.8$. The average number of surplus employees needed on different days of the week are calculated equally, giving values 3.33, 3.33, 3.33, 3.33, 3.55, 1.91, and 0.71. The constraint **O5** can now be rephrased as "On Mondays, Tuesdays, Wednesdays, Thursdays and Fridays either three or four, on Saturdays either one or two, and on Sundays either zero or one surplus employees must be guaranteed".

The objective is to find a solution that has no hard constraint violations and that minimizes the weighted sum of the soft constraint violations. We use the adaptive penalty method for multi-objective optimization (see Section 4). The importance of the soft constraints is handled by giving them different constant weights. Hard constraint weights are dynamically calculated according to the ADAGEN method. The values of the weights, given in Table 3, were decided based on the negotiations with the company. Note that the hard constraint **C1** is not listed in the table because the algorithm uses the exact number of employees given in Table 2.

Table 3. The constant weights for the soft constraints and the maximum values for the hard constraints (minimum is one)

E1	**E2**	**E3**	**R1**	**R4**	**R5**	**O1**	**O2**	**O5**	**P1**
1	10	5	50	50	50	20	30	50	70

We generated ten days-off schedules and selected the best one. The schedule had no hard constraint violations and 886 single days-off. The lengths of days-off sequences were between one and three, and the number of single days-off and single working days between employees did not differ by more than 25%. As a result, the weighted sum of the soft constraint violations was 886. The algorithm was run on an Intel Core 2

Extreme QX9775 PC with a 3.2GHz processor and 4GB of random access memory running 64bit Windows Vista Business Edition. The best solution was found in 16 hours of computer time. The time may first appear to be long. However, the point here is not to find a solution fast enough and with sufficient quality. Instead, the main point here is to find the solution of the best quality. The planning horizon is one year, so it is worth running the algorithm overnight.

5. Solution Method

We believe that metaheuristics are best suited to solving the problem at hand, because of the difficulty in finding even a feasible solution and because of the number of constraints. In fact, our studies have shown that in practical cases it is often possible to modify one's view of what is feasible and what is not.

The algorithm has features from many different optimization methods. It was first introduced as a hybrid genetic algorithm with one mutation operator and no recombination operators. Later, when the terminology evolved, it could have been presented as a genetic local search method or as a memetic algorithm. The algorithm uses features from tabu search, simulated annealing, variable neighborhood search and hyper-heuristics. Finally, the greedy hill-climbing mutation (GHCM) operator introduced in [18] is based on similar ideas to ejection chains. We believe the best way to describe our algorithm is to call it a cooperative local search [8]. Here, we describe the components of the algorithm. The details are further discussed in [19] and [14]. We believe the best way to describe our algorithm is to call it a cooperative local search [8]. The outline of the algorithm is given in Figure 2.

Marriage selection is used to select a schedule from the population of schedules for a single GHCM operation. The GHCM operator moves an object, *o1*, from its old position, *p1*, to a new position, *p2*, and then moves another object, *o2*, from position *p2* to a new position, *p3*, and so on, ending up with a sequence of moves. In days-off scheduling, an object is a day-off. A day-off can be moved between different employees. A tabu list prevents reverse order moves in the same sequence of moves. The simulated annealing feature uses the exponential cooling scheme. We stop the cooling at a predefined temperature. Therefore, after a certain number of iterations, we will continue to accept an increase in the cost function with some constant probability, p (equal to 0.0015). In addition, after a given number of iterations we shuffle the current solution. We use five simple shuffling operations.

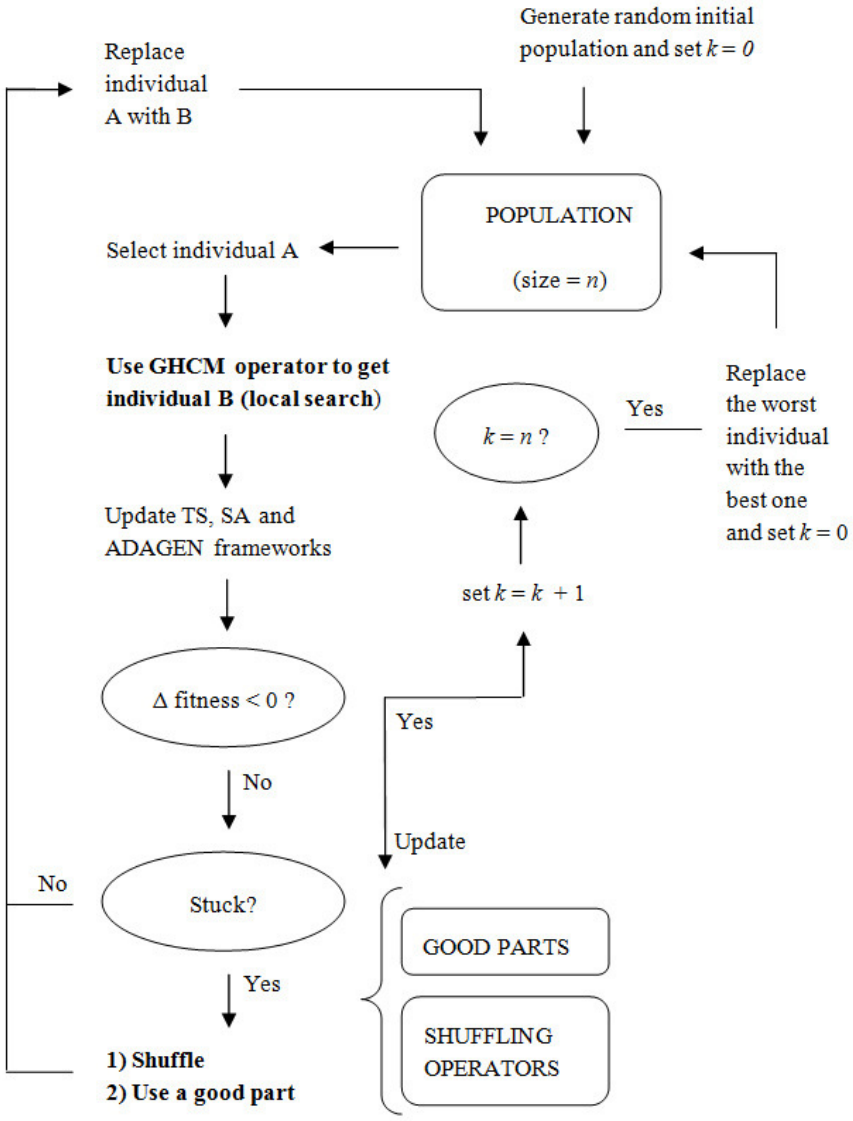


Figure 2. The outline of the staff scheduling algorithm.

The reproduction operation of the algorithm is based on the steady-state reproduction to a certain extent: the new schedule replaces the old one if it has a better or equal fitness. Furthermore, the least fit is replaced with the best one when n better schedules have been found, where n is the size of the population.

The ADAGEN method used is an adaptive penalty method for multi-objective optimization. The method assigns dynamic weights to the hard constraints instead of to the soft constraints. Finally, the current production version of the algorithm for the presented staff scheduling problems does not record and use the good parts of the previous solutions. The parameters of the algorithm are the same as were found to work best in [19]; that is, the population size equals 20, the maximum move sequence in the GHCM equals 10 and the size of the tabulist is 5.

6. The Difficulty of the Process

An essential part of solving any practical problem is the process of consulting with the various parties. We started by familiarizing ourselves with different staff scheduling problems. We had previously solved practical school timetabling and sports scheduling problems, but had no clear understanding of staff scheduling. The first stage of the consulting process was an interview with the CEO of the company. One important thing to note is that he wanted to forget all the historical burdens to start from scratch. After the interview we were quite pleased that the literature we had read had given us the correct background we needed. We were confident that we were being asked to solve a variation of the shift scheduling problem.

We returned to the company after two months of further reading and analyzing the given problem. This time we interviewed the transport coordinator, who is responsible for the practical scheduling of the buses and the bus drivers. She gave us new requirements and requests that we were not aware of, and corrected some old ones. We were somewhat surprised that she had a quite different vision of the future rostering system than the CEO had. But still we were quite confident of what we were doing, and decided to meet again within two months. Based on this negotiation we made some final corrections and adjustments and created the mathematical model of the given problem. We also had time to design and code the first version of the algorithm. We then returned to meet the CEO, the transport coordinator and her assistant to present our current work.

What happened then was a total surprise. They were astonished that we had not optimized the days-off. They claimed that it was the most

important part of the scheduling process. We had thought they would give the days-off to us, or to our algorithm as an input, and we would then solve the monthly shift scheduling problem. But that was not the case. We discovered that we should first solve the days-off scheduling problem for the whole year. So, we went back home and started to design another mathematical model for the new problem. Why we did not understand that in first place is still a total mystery for us. But it is worth noting that from the very beginning of the process they were enthusiastic and not at all skeptical of us trying to beat the rosters they had produced themselves.

It took another month before we could return to the customer. We had done nothing to the shift scheduling problem, and had only designed and coded an algorithm for the given days-off scheduling problem; just to be surprised again. We were earlier told very complicated rules of how to assign given employees to the same shifts. It had taken us a lot of effort to model and code the given rules. In that meeting we found out that it was not complicated at all: exactly the same sequence for three employee groups with three employees in each group must be guaranteed (see **P1** in Section 4). Once more we went back home, simplified our model and returned to the customer. We presented our software to the CEO and the transport coordinator, and they were very pleased with the results. What is the lesson learned? Researchers and customers speak a somewhat different language; and it is the researchers' job to learn a new one. Customers tend to learn what they want not until they have seen the outcome. That is why it pays off to first spend some time to illustrate simple, easy and explicit problem instances and their possible outcomes with customers. And even better if that could be done with "paper and pencil".

7. A Set of Test Instances

The generation of standard benchmark problems for staff scheduling has received only some attention. The best test instances for employee scheduling have been introduced in [9]. To the best of our knowledge, no set of standard test instances has been published for days-off scheduling problems. Researchers quite often only solve one real-world case. The strength of artificial test instances is the ability to produce many problems with many different properties. Still, they should be simple enough for each researcher to be able to use them in their test environment. The strength of practical cases is self-explanatory. However, an algorithm performing well on one practical problem may not perform

satisfactorily on another practical problem. That is why we present the first collection of artificial test instances for the days-off scheduling.

Table 4. Nineteen days-off scheduling test instances (n = the total number of employees, m = the exact number of employees needed each working day, t = the number of timeframes, w = the length of a timeframe in weeks, d = the number of days-off in a timeframe, s_{max} = the maximum length of a work stretch, c_{max} = the maximum length of consecutive days-off, eq = the number of identical days-off sequences between the employees, no = the number of employees who cannot work at weekends). The solutions (sol) are the best of three runs.

#	n	m	t	w	d	s_{max}	c_{max}	eq	no	sol
1	7	4	1	2	6	5	2			0*
2	14	9	1	2	5	5	3			1
3	14	11	1	2	3	6	2			14*
4	14	8	1	3	9	4	3			0*
5	15	10	1	3	7	5	3			0*
6	16	8	1	2	7	3	3			0*
7	28	19	1	4	9	5	3			6
8	28	20	2	2	4	5	2			32
9	42	27	2	4	10	6	4			0*
10	56	36	2	4	10	6	4			0*
11	28	14	4	4	14	4	4			0*
12	28	19	4	4	9	5	3			49
13	14	6	6	2	8	3	3			20
14	14	9	8	4	10	6	3			0*
15	32	16	8	2	7	3	3			0*
16	14	9	10	2	5	5	3			0*
17	14	9	5	2	5	5	3	3		10
18	14	9	5	2	5	5	3		2	6
19	14	9	5	2	5	5	3	3	2	15

Table 4 shows 19 test instances. The total number of employees (n) varies between 7 and 56, and the number of total weeks ($t \times w$) between 2 and 36. The exact number of employees (m) must be guaranteed for each day. The maximum lengths of work stretches and consecutive days-off are given. Their minimum lengths are two. Two instances require a number of identical days-off sequences between the employees, and in two instances a given number of employees cannot work at weekends. The challenge is to find a feasible solution that minimizes the number of single days-offs and single working days. The penalty for a single days-off is one and the penalty for a single working day is two.

We were able to find the optimum solution (*) for eleven of the test instances. For the other instances the optimum is not yet known. We hope

these test instances will lay the foundation for the standard benchmark instances for days-off scheduling problem.

8. Conclusions and Future Work

We scheduled days-off for the staff at a Finnish bus transportation company. Our algorithm found a feasible and acceptable schedule for their days-off scheduling problem. The generated days-off are currently in use. We believe that the model and the algorithm presented in this paper can be quite easily modified and transferred to solve other staff scheduling problems. We also proposed a set of test instances that we hope the researchers of the days-off scheduling problems will adopt.

We are currently solving the shift scheduling problem in the same company. Our direction for future research will be to solve nurse rostering problems occurring in major Finnish hospitals.

References

[1] J. F. Bard, C. Binici, and A. H. Desilva. Staff scheduling at the United States Postal Service. *Comput. Oper. Res.*, 30(5):745–771, 2003.

[2] J. J. Bartholdi III. A Guaranteed-Accuracy Round-off Algorithm for cyclic scheduling and set covering. *Oper. Res.*, 29(3):501–510, 1981.

[3] G. R. Beddoe, S. Petrovic, and J. Li. A hybrid metaheuristic case-based reasoning system for nurse rostering. *J. Sched.*, 12(2):99–119, 2009.

[4] A. Beer, J. Gaertner, N. Musliu, W. Schafhauser, and W. Slany. Scheduling breaks in shift plans for call centers. In *Proc. 7th International Conference on the Practice and Theory of Automated Timetabling*, Montréal, Canada, 2008.

[5] F. Bellanti, G. Carello, F. Della Croce, and R. Tadei. A greedy-based neighborhood search approach to a nurse rostering problem. *Eur. J. Oper. Res.*, 153(1):28–40, 2004.

[6] B. Bilgin, P. De Causmaecker, B. Rossie, and G. Vanden Berghe. Local search neighbourhoods to deal with a novel nurse rostering model. In *Proc. 7th International Conference on the Practice and Theory of Automated Timetabling*, Montréal, Canada, 2008.

[7] E. K. Burke, P. De Causmaecker, G. Vanden Berghe, and H. Van Landeghem. The State of the Art of Nurse Rostering. *J. Sched.*, 7(6):441–499, 2004.

[8] E. K. Burke and J. D. Landa Silva. The Design of Memetic Algorithms for Scheduling and Timetabling Problems. In N. Krasnogor, W. E. Hart, and J. E. Smith, editors, *Recent Advances in Memetic Algorithms, Studies in Fuzziness and Soft Computing*, volume 166, pages 289–312. Springer, 2004.

[9] T. Curtois. Employee Scheduling Data Sets. Available: `http://www.cs.nott.ac.uk/~tec/NRP/`. (Last update 13.10.2009).

[10] G. B. Dantzig. A comment on Edie's traffic delays at toll booths. *Oper. Res.*, 2(3):339–341, 1954.

[11] D. Dowling, M. Krishnamoorthy, H. Mackenzie, and D. Sier. Staff rostering at a large international airport. *Ann. Oper. Res.*, 72(1):125–147, 1997.

[12] A. T. Ernst, H. Jiang, M. Krishnamoorthy, and D. Sier. Staff scheduling and rostering: A review of applications, methods and models. *Eur. J. Oper. Res.*, 153(1):3–27, 2004.

[13] A. Fukunaga, E. Hamilton, J. Fama, D. Andre, O. Matan, and I. Nourbakhsh. Staff scheduling for inbound call and customer contact centers. *AI Mag.*, 23(4):30–40, 2002.

[14] J. Kyngäs and K. Nurmi. Scheduling the Finnish Major Ice Hockey League. In *Proc. IEEE Symposium on Computational Intelligence in Scheduling*, Nashville, USA, 2009.

[15] J. Kyngäs and K. Nurmi. Scheduling the Finnish 1st Division Ice Hockey League. In *Proc. 22nd Florida Artificial Intelligence Research Society Conference*, Florida, USA, 2009.

[16] H. C. Lau. On the complexity of manpower shift scheduling. *Comput. Oper. Res.*, 23(1):93–102, 1996.

[17] D. Marx. Graph coloring problems and their applications in scheduling. *Period. Polytech. Ele.*, 48(1):5–10, 2004.

[18] K. Nurmi. Genetic Algorithms for Timetabling and Traveling Salesman Problems. Ph.D. dissertation, Department of Applied Mathematics, University of Turku, Finland, 1998.

[19] K. Nurmi and J. Kyngäs. A framework for school timetabling problem. In *Proc. 3rd Multidisciplinary International Scheduling Conference: Theory and Applications*, Paris, France, 2007.

[20] D. Pedrosa and M. Constantino. Days-off scheduling in public transport companies. *Lect. Notes Econ. Math.*, 505:215–232, 2001.

BIOINSPIRED ONLINE MATHEMATICS LEARNING

Barbara Koroušić Seljak, Gregor Papa
Computer Systems Department
Jožef Stefan Institute, Ljubljana, Slovenia
{barbara.korousic; gregor.papa}@ijs.si

Abstract Learning mathematics is like studying a foreign language. At first it is hard, but eventually, it gets progressively easier. A lot of concepts in mathematics are inter-related, so knowing one helps understand many others. However, it has been known for a long time, through many descriptive studies that have been undertaken since the 1970s, that mathematics has been unpopular and disliked. Namely, steps required for learning mathematics, such as "*Make sure you have at least an hour a day to dedicate to learning mathematics. Progress through the levels of mathematics. Practice with many problems.*" are demanding for many students. In this paper, we present a methodology for learning elementary-school mathematics online. It is supported by a decision-making system, based on evolutionary computation, which leads a student in selecting an optimal subset of math items to effectively upgrade the knowledge.

Keywords: e-learning, Evolutionary optimization, Web-based application

1. Introduction

The modern generation of students have grown up with technology as a commodity for playing, social networking, obtaining many kind of information and even for learning. Many of them act within virtual environments and have developed virtual identities. They live in human and technical networks that provide new opportunities for the presentation of various experiments and knowledge. Learning within such networks is based on concepts of aggregation, externalization, collective knowledge creation and immersion [14].

In order to investigate the possibility of learning elementary-school mathematics online, we developed a web-based application, called MatPort (`sinica.ijs.si/matport`; Fig.e 1).[1]

Figure 1. The MatPort system.

Although it is based on information and communication technologies, it has a social dimension. Being part of a group and working as a collective enables students to share and discuss the mathematical knowledge. On the other hand, online learning

- is self-paced and gives students a chance to speed up or slow down as necessary;
- is self-directed, allowing students to choose content and tools appropriate to their differing interests, needs, and skill levels;
- is designed around a student.

It eliminates geographical barriers, opening up broader education options, and enhances other (e.g., computer) skills.

The rest of the paper is organized as follows: Section 2 describes the content and the technology aspects of the web application. Section 3 outlines the concept of the decision-making system used to provide an automatic search facility. The experimental work is presented in Section 4, while conclusions and directions for future work are given in Section 5.

2. Web-Based System

In Slovenia, there already exist web-based applications supporting online learning of elementary school mathematics, e.g.,:

- *Učiteljska.net* (`uciteljska.net`), which is intended to support exchange of teacher experience and information;

- *e-um* (`www.e-um.si`) that provides interactive online courses for students.

However, there has been no system providing online courses on solving verified mathematics problems.

2.1 MatPort Characteristics

There are two main characteristics that distinguish the web-based application MatPort:

1 The system is based on the Slovenian curriculum for elementary school mathematics;

2 It provides a set of verified mathematical problems for children under 15 years of age [4].

In the pilot project, we have used a collection of verified sets of grades 6 to 9 (age group 12 to 15 years) mathematics problems published in the form of flash cards. These problem-solving items are valuable because they have been used for many years in the Slovenian schools, and have already passed through many steps of evaluation. Of course, some of the real-life items needed to be updated for the present time.

With the help of experienced teachers we classified the items into subgroups with respect to the knowledge required for problem solving. Items were classified by the content into three difficulty levels.

2.2 MatPort Modules

The web-based application provides modules for:

- Entering math items, their solutions, and teaching instructions;

- Solving math items;

- Preparing the paper-and-pencil form of a test;

- Providing other information relevant for teaching the elementary-school mathematics.

While the first and the third module are aimed for teachers, the second one is intended for students, and the last one for teachers and parents.

2.3 MatPort Technology Aspect

The application was designed using state-of-the-art technologies. We applied a UML (*Unified Modelling Language*) based model-driven methodology (`www.uml.org`) to cover the life-cycle of the Web application development. The MatPort's data model was designed as a rational database, consisting of tables that store data on user profiles, math items, knowledge required for solving items, relations between the content areas, and history of items' solving. In this stage we involved experienced teachers to design the application sympathetically with the way students, teachers, and parents actually use the Web - not how we think they should.

The application consists of the following modules:

- modules for providing and solving math items;
- a test-generator module;
- an informer module; and
- other modules, such as a forum and a download center.

The informer module has a static content that is managed by a content management system, while the others may change their content in a dynamic way. The application's structure is shown in Fig. 2.

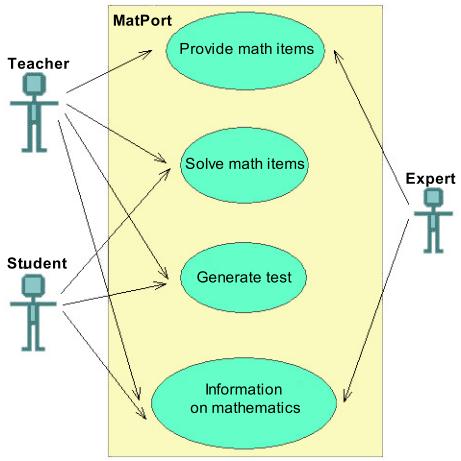


Figure 2. The MatPort's structure.

As the project's budget was low and we needed to minimize the cost, we decided to make good use of open-source and freely available software applying:

- Apache (`www.apache.org`) as a Web server that replies to Web clients' requests via HTTP (*HyperText Transfer Protocol*);
- MySQL (`www.mysql.com`) as a database management system that is based on the relational model;
- PHP5 (*HyperText Preprocessor*) (`www.php.net`) as a server-side technology;
- JavaScript [3] as a client-side technology.

To support mathematical symbols, we integrated TinyMCE (`tinymce.moxiecode.com`) as a platform independent JavaScript WYSIWYG editor. It has the ability to convert HTML TEXTAREA fields or other HTML elements to editor instances. Because TinyMCE supports only standard HTML math symbols, we extended its library with a module for handling mathematical expressions. Once entered, these are rendered into images using the LatexRender scripts (`www.mayer.dial.pipex.com/tex.htm`). All MatPort printouts have the standard PDF (*Portable Document Format*) format. The MatPort forum is based upon the PHPBB™ (`www.phpbb.com`) open-source forum solution. Last but not least, we set formatting MatPort visual options in a centralized document that is referenced from PHP files by using CSS (*Cascading Style Sheets*).

3. Decision-Making System

We are aware of the fact that only providing a dataset of math items is not enough. We need to incorporate an extrinsic motivation system to bribe 6^{th} to 9^{th} grade students to practice mathematics. There are rare children who are intrinsically motivated to do repetitive, boring tasks.

3.1 How to Motivate Children?

A student who solves a MatPort math item receives information on the progress through graphical symbols and stimulative words. Each correctly solved math item, regardless of difficulty level, contributes to the final score.

In addition, the application provides information on math items that need to be further solved to receive a higher score. These can be supplementary or additional items to help strengthen or increase knowledge,

respectively. The information is provided by the MatPort decision support system, when the automatic search facility is used.

3.2 Evolutionary Computation

The MatPort decision-making system, aimed to motivate students to continue solving exercises, is based upon an evolutionary computation method, i.e., the genetic algorithm (GA) [1, 5]. The GA is based on a heuristic method, which requires little information to search effectively in a large search space. The algorithm employs an initial population of chromosomes, which evolve to the next generation by probabilistic transition rules (randomized genetic operators), such as selection, crossover and mutation. The objective function evaluates the quality (fitness) of solutions coded as chromosomes. This information is used to perform an effective search for better solutions. There is no need for other auxiliary knowledge. The GA tends to take advantage of the fittest solutions by giving them greater weight, and by concentrating the search in the regions of the search space with likely improvement.

The GA is a population-based evolutionary approach that allows searching within a broad set of solutions from the search space simultaneously. Namely, because there are

- many math items (few hundreds or even more than thousand math items per a grade), and
- many interrelated content areas (more than 100 content areas per a grade),

the student may continue solving items in many possible ways that may or may not lead to a higher score. Moreover, math items are dynamically generated by teachers (i.e., the item dataset expands with time) and the student may start solving them anywhere in the dataset. In the GA, there is a risk of converging to a local optimum, but good results of various research work obtained in other optimization problem areas [5, 8, 9, 11, 13] encouraged us to consider the GA as a promising approach to the decision-making problem.

We developed our own version of the GA to fully adapt to the specific problem. The details of the implemented GA, which takes into consideration the directions introduced in [2, 10], are described in the subsections below.

The main idea is to find a set of math items within different content areas that, when solved correctly, improve the user's knowledge and increase his/her score as much as possible. The set of items should consist of math problems from all poorly-scored content areas and the

areas that precede these areas. Therefore, before starting a search, the system identifies all feasible items, i.e., math problems from the poorly-scored content areas. These items form some kind of a pool of relevant items P for the current-score improvement.

Encoding The suggested list of math items needed to improve the score is encoded into a chromosome, where each gene represents the identification (ID) number of the item in the MatPort database. The chromosome length has been fixed to 15, while this number represents a reasonable number of items to perform, in order to significantly improve the score.

Population Initialization The initial population consists of n chromosomes. The initial chromosomes are set with randomly chosen items from the pool of relevant math items. In case of identical initial chromosomes some chromosomes are repaired by permutation to ensure versatility.

The only requirement in this initialization stage is that no item is repeated within each set (i.e., chromosome).

Genetic Operators The elitism approach is used - to store the best found solution.

The substitution of the least-fit chromosomes with the equal number of the best-ranked chromosomes ensures better solutions to have more influence on the new generation. The ratio of all chromosomes in the population to be replaced is set by the ratio r.

With the one-point crossover scheme, chromosome mates are chosen randomly and with a probability p_c all items after randomly chosen position are swapped, which leads to two new solutions that replace their original sources.

In the mutation process each item of the chromosome mutates with a probability p_m. If the item of the chromosome needs to be changed, than some new item from the pool of relevant math items is chosen and placed onto the mutated position of the chromosome.

There is no special repair function implemented to be used when two identical items appear in the chromosome, either during the crossover or mutation process. If this occurs then it leads to lower score of the evaluation function, since not enough different items would be solved. Consequently, such unrepaired chromosomes are removed from the population on the bases of evolution. Also, the items are not sorted within the chromosome, therefore there is no need to check the position of some item within the chromosome. The final order of the items is set at the

end, after the GA finishes its search, and is set according to the levels and dependencies of different items.

When enabled, the variable mutation probability p_m is decreasing linearly with each new population. Since each new population generally fits better, we overcome a possible disruptive effect of mutation at the late stages of the optimization, and speed up the convergence of the GA in the final optimization stages. Moreover, the lower number of mutated positions in the later stages presents some kind of a local search with minor movements around current solution, i.e. fine-tuning.

Fitness Evaluation After the variation operators modify the solutions, the whole new population of chromosomes is ready to be evaluated. In the evaluation process the set of math items is assumed to be solved correctly and the score improvement is calculated. The calculated score improvement is used as a fitness value of each chromosome. Here all the items are weighted with respect to

- difficulty levels,
- content areas, and
- relations between content areas

to increase the diversity; the order of items is relevant when the problems belong to different content areas that derive from each other.

Parameter Settings In order to ensure optimal solutions in a reasonable response time robust parameter settings need to be found for the population size, number of generations, selection criteria and genetic operator probabilities:

- If the population size and the number of generations are too small, the GA converges too quickly to a local optimal solution and may not find the best solution. On the other hand, a large population and too much iteration require long time to converge to a region of the search space with significant improvement. In our case, we have used the population size $n = 15$ and number of generations $n_g = 30$;
- With applying the elitism strategy, fitter solutions have greater chance to be reproduced. But when the number of worse solutions to be exchanged with better ones (the selection criteria) is too high, the GA is trapped too quickly in a local optimum solution. Our replacement rate r has been 20%;

- Too low crossover probability preserves solutions to be interchanged and longer time is required to converge. This probability should be large enough to crossover almost all mated solutions. In our case, efficient setting for p_c has been 70%;
- Too high mutation probability may introduce too much diversity and takes longer time to reach an optimal solution. Too low mutation probability tends to miss some near-optimal solutions. Again, the efficient setting for p_m has been 5%.
- When variable mutation was enabled the p_m decreased linearly to 1%, from the first till the last generation.

Termination After a certain number of populations are generated and evaluated, the system is assumed to be in a non-converging state. A chromosome with the highest score improvement is chosen as a final result.

On average, the GA finds an optimal selection of math items within all the content areas that need to be further solved by the user to receive a higher score (i.e., obtain a sufficient knowledge) in order of few seconds. As it is implemented as a background process, it does not slow down the application.

4. Experimental Work

We ran the GA to collect sufficient amount of results, which were then subjectively judged by three experienced teachers. They estimated the decision-making system as a reliable tool:

1 Math items were selected from content areas that really needed to be further investigated;

2 Content areas were ordered in a meaningful way;

3 Selected math items could improve the student's knowledge;

4 The results were generated in real-time, in order of seconds.

However, we still need to prove this by using statistics. We should involve a group of students and their teachers, and statistically evaluate the effectiveness of the MatPort system in different aspects, measuring the success in terms of knowledge and joy of learning.

5. Conclusions

The main aim of the methodology presented in this paper is to provide a modern tool that would support education, which is more than

acquiring information and knowledge, but is also about whether and how these are memorized and used.

First, the instructional design of the MatPort Web application, which is aimed for elementary mathematics online learning in the active way, was presented. We upgraded this design by incorporating a high-performance evolutionary computation method to support automatic search of relevant math items. In this way, the MatPort may lead its users toward higher scores in a thoughtful way. The main reason for selecting a heuristic method has been that we wished to provide a non-deterministic behavior of the decision-making system to overcome the problem of cheating. Finally, we described the method for evaluation of the application's effectiveness.

After the pilot stage of the project, we are planning to expand the dataset of math items to other elementary school grades. We will increase its efficiency through additional motivation tools, such as winner lists or computer games, which will be activated as soon as a student will gain a certain score.

Much work needs to be done to find an adequate level of human intervention. In cooperation with teachers, we will try to improve the way of providing teaching instructions and intermediate solutions. Last but not least, we will discuss the problem of cheating.

In addition, we will do some experimental work on the application of the efficient parameter-less evolutionary search method [12] as a substitution for the currently implemented genetic algorithm.

Notes

1. It has been known for a long time, through many descriptive studies that have been undertaken since the 1970s, that mathematics has been unpopular and disliked [7].

References

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

[2] T. Bäck, D. B. Fogel, and Z. Michalewicz. *Evolutionary Computation 1: Basic Algorithms and Operators.* Taylor & Francis Group, 2008.

[3] C. Darie, B. Brinzarea, F. Chereches-Tosa, M. Bucica. *AJAX and PHP*, 1st edition. PACKT Publishing, Birmingham, 2006.

[4] N. Doganoc, M. Lenarčič, A. Mihelič, Š. Pirnat. *Računski listki za 6. razred, naloge in rešitve.* ZPS Ljubljana.

[5] T. Garbolino and G. Papa. Genetic algorithm for test pattern generator design. *Appl. Intell.*, In press, doi:10.1007/s10489-010-0214-7.

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

[7] P. Grootenboer and R. Jorgensen (Zevenberger). Towards a theory of identity and agency in coming to learn mathematics. *Eurasia J. Math. Sc. Tech. Educat.*, 5(3):255–266, 2009.

[8] P. Korošec, J. Šilc. Using stigmergy to solve numerical optimization problems. *Comput. Inform.*, 7 (3):377–402, 2008.

[9] B. Koroušić Seljak. Computer-based dietary menu planning. *J. Food Compos. Anal.*, 22(5):414–420, 2009.

[10] Z. Michalewicz and D. B. Fogel. *How to Solve It: Modern Heuristics*. Springer-Verlag, 2004.

[11] G. Papa and B. Koroušić Seljak. An artificial intelligence approach to the efficiency improvement of a universal motor. *Eng. App. Artif. Intel.*, 18(1):47–55, 2005.

[12] G. Papa. Parameter-less evolutionary search. In *Genetic and Evolutionary Computation Conference (GECCO)*, pages 1133–1134, 2008.

[13] T. Tušar, P. Korošec, G. Papa, B. Filipič, and J. Šilc. A comparative study of stochastic optimization methods in electric motor design. *Appl. Intell.*, 27(2):101–111, 2007.

[14] W. Veen. Homo Zappiens, Learning in a Digital Age. In *Proc. First International Conference on Computer Supported Education (CSEDU)*, pages 13, Lisboa, Portugal, 2009.

WOODY PLANTS MODEL RECOGNITION BY DIFFERENTIAL EVOLUTION

Aleš Zamuda, Janez Brest, Borko Bošković, Viljem Žumer
Faculty of Electrical Engineering and Computer Science
University of Maribor, Slovenia
{ales.zamuda; janez.brest; borko.boskovic; viljem.zumer}@uni-mb.si

Abstract This paper presents an approach for recognition of procedural three-dimensional models of woody plants (trees). The used procedural tree model operates by building a three-dimensional structure of a tree by applying a fixed procedure on a given set of numerically-coded input parameters. The parameterized procedural model can later be used for computer animation. Recognition of a parameterized procedural model, from the photographic images, is done by differential evolution algorithm which evolves this model by fitting a set of its rendered images to a set of given photographic images. The comparison is done on a pixel level of the images through the integration of distances to the nearest similar pixels. The obtained results show that the presented approach is viable for modeling of woody plants for computer animation by evolution of the numerically-coded procedural model.

Keywords: Differential evolution, Numerical encoding, Procedural model, Self-adaptation, Structure recognition, Woody plant

1. Introduction

In this paper we present a new approach to design three-dimensional geometrical models for woody plants (trees). The geometrical models are expressed indirectly with the use of procedural models to reduce the enormous data storage space needed for their representation. The procedural models can also be easily animated and are suitable in computer graphics and animation. Our new approach to design of woody plant models is based on recognition of their procedural models [21], from images using evolutionary algorithms [22]. The paper [2] presents an approach for recognition of procedural models. However, the procedural models obtained in [2] were not as complex to express woody plants. Also the recognized procedural models were two-dimensional.

Therefore, we extend this approach to the domain of three-dimensional procedural models suitable to model woody plants.

In the next section, the related work is presented. In the Section 3, the proposed approach for procedural models recognition using differential evolution is described. In the Section 4, experimental results and their discussion is given, which show that the given approach is suitable for design of woody plant models. The Section 5 concludes with final remarks and propositions for future work.

2. Related Work

In this section, we present the differential evolution algorithm and one of its improvements, the jDE algorithm [4, 6]. Then, we list some of the procedural models for modeling of trees and outline the numerically-coded procedural model of the EcoMod framework [21, 23, 25].

2.1 Differential Evolution

Differential Evolution (DE) [18] is a floating-point encoding evolutionary algorithm for global optimization over continuous spaces, which can also work with discrete variables. Its main performance advantages over other evolutionary algorithms [4, 11] lie in floating-point encoding and a good combination of evolutionary operators, the mutation step size adaptation and elitistic selection. The DE algorithm has a main evolution loop in which a population of vectors is computed for each generation of the evolution loop. During one generation G, for each vector $\mathbf{x}_i$, $\forall i \in \{0, NP\}$ in the current population, DE employs evolutionary operators, namely mutation, crossover, and selection, to produce a trial vector (offspring) and to select one of the vectors with best fitness value. NP denotes population size and G the current generation step.

Mutation creates a mutant vector $\mathbf{v}_{i,G+1}$ for each corresponding population vector. One of the most popular DE mutation strategies is *'rand/1/bin'* [14, 18]:

$$\mathbf{v}_{i,G+1} = \mathbf{x}_{r_1,G} + F(\mathbf{x}_{r_2,G} - \mathbf{x}_{r_3,G}),$$

where the indexes r_1, r_2, and r_3 represent the random and mutually different integers generated within the range $\{1, NP\}$ and also different from index i. F is an amplification factor of the difference vector within the range $[0, 2]$, but usually less than 1. Vector at index r_1 is a base vector. The term $\mathbf{x}_{r_2,G} - \mathbf{x}_{r_3,G}$ denotes a difference vector which after multiplication with F, is named amplified difference vector.

After mutation the mutant vector $\mathbf{v}_{i,G+1}$ is taken into recombination process with the target vector $\mathbf{x}_{i,G}$ to create a trial vector $u_{i,j,G+1}$. The

binary crossover operates as follows:

$$u_{i,j,G+1} = \begin{cases} v_{i,j,G+1} & \text{if } rand(0,1) \leq CR \text{ or } j = j_{rand} \\ x_{i,j,G} & \text{otherwise} \end{cases},$$

where $j \in \{1, D\}$ denotes the j-th search parameter of D-dimensional search space, $rand(0,1) \in [0,1]$ denotes a uniformly distributed random number, and j_{rand} denotes a uniform randomly chosen index of the search parameter, which is always exchanged to prevent cloning of target vectors. CR denotes the crossover rate.

Finally, the selection operator chooses one of the vectors with a better fitness value (for minimization problem):

$$\mathbf{x}_{i,G+1} = \begin{cases} \mathbf{u}_{i,G+1} & \text{if } f(\mathbf{u}_{i,G+1}) < f(\mathbf{x}_{i,G}) \\ \mathbf{x}_{i,G} & \text{otherwise} \end{cases}.$$

DE was proposed by Storn and Price [18] and since then, it has been modified and extended several times with new versions proposed [14, 9]. We have used the jDE algorithm [4], which adds to the original DE, a self-adaptation mechanism of F and CR control parameters. In this work, only the original jDE algorithm [4] was used, although the algorithm also has some extensions that have not been used in this work [5, 6, 7].

2.2 Woody Plants Procedural Models

The procedural modeling of trees has a thirty year tradition in computer graphics. Manual editing of a tree structure and its leaves is a tedious task, since each branch and leaf position, rotation, size, and texture must be appointed. Therefore, procedural tree models are used instead, and several techniques for procedural models are available today. Different procedural models are based on various types of branching structure construction [15]. These techniques differ in the level of detail [1, 3, 16], the flexibility, and pretentiousness of modeling [10, 19], space [13], and time complexity [13] in addition to the animation ability and representation of the built three-dimensional model. The majority of these models try to determine some visible properties of the final three-dimensional model, such as the rotation of branches around their central axes. These properties are usually biologically inspired by *phyllotaxis*, i.e. the main influence on the tree's architecture [17].

Holton [10] created trees with the use of biologically inspired strand model. An upside of this model is that, thickness of branches and proportions between branching angles are determined directly with internal rules in the model. Strands flow along branches and are divided with-

out splitting a single strand. Branches with single strands are carrying leaves. Strand distribution determines branch thickness and their lengths. User enters the number of strands along the tree, proportions between branch lengths and branching angles to parametrize the procedural model. Certain attractors influence the branching structure, e.g. central trunk uprightness, gravimorphism, phototropism, planartropism, and phyllotaxis. A downside of the model is that user still has to enter a huge amount of numerical data, which diminishes the flexibility of the model.

Weber and Penn [19] represented the tree model with the use of simple geometry without a development of branching topology. For all branches at the same levels, they entered branching angle, branch length proportions and thickness for branches. They presented wind sway animation, branch cutting to predetermined volume, and progressive level of detail rendering.

The EcoMod framework incorporates a procedural model for woody plants, based on the Holton and Webber-Penn models. The procedural model and its modeler with woody plant models library was first introduced in [25] and is in greater detail described in [21, 23, 24]. The procedural model also helps to design the tree from a minimized set of parameters that the user must set by automatically determined positions, rotations, sizes and textures for several thousand branch segments and several thousand leaves. An individual tree species model is created by parametrizing the procedural model. It generates a three-dimensional structure [20] of a tree by recursively executing a fixed procedure over a given set of numerically coded input parameters, such as branch thickness, relative branch length and branching structure proportions. Each step of the procedure adds a building block of a tree to the geometrical model. The trees designed with this model can be foliage or coniferous trees with very different branching structures. Each branch and each leaf can be animated in real time to show the growth of a tree or its sway in the wind. By slightly modifying the parameters of procedural models, we can achieve computer animation of these models [24], thereby creating several geometrical models from a single procedural model.

Parameters of EcoMod woody plant procedural model are distinguished as vectors (local) and scalars (global). Global parameters are constant for all branch segments although local parameters vary along Gravelius (g) and Weibull (w) branch order. Vector parameters design the strand distribution, branching angles, branch segment proportions, and gravity impact to tree geometry. Scalar parameters of the model are height and thickness of base trunk, wind impact, and density and size of leaves. Using listed vector and scalar parameters, geometrical model is

Algorithm 1 Calculation of geometrical structure of the procedural model tree. Recursive procedure is called using **branchsegment**(0, 0, S, 1, $l_0^{0,0}$, **I**, **I**, **I**), **I** denoting an identity matrix.

procedure branchsegment(g, w, S_0, L_0, l_0, $\mathbf{M}_0$, $\mathbf{M}_{m;0}^{-1}$, $\mathbf{M}_{w;0}^{-1}$)

Require: g, w - Gravelius and Weibull index of base branch; S_0 - number of strands in base branch; L_0, l_0 - base branch relative and actual length; $\mathbf{M}_0$ - base branch coordinate system; $\mathbf{M}_{m;0}^{-1}$ - inverse matrix of rotations for gravimorphism in coordinate system for base branch; $\mathbf{M}_{w;0}^{-1}$ - inverse matrix of rotations for directed wind in coordinate system for base branch; *global* k_d, k_c, l_{type}, $k_s^{g,w}$, $M^{g,w}$, $m^{g,w}$, $k_l^{g,w}$, $\alpha_m^{g,w}$, $\alpha^{g,w}$, t, k_f, w_s, w_g

Ensure: rendered tree image

$d := k_d\sqrt{S_0}$; {thickness calculation from Mandelbrot}

render base branch($\mathbf{M}_0$, l_0, d);

if $S_0 = 1$ **then**

  render leaves(l_{type}); **return**;

end if

$S_1 := \lceil 1 + k_s^{g,w}(S_0 - 2) \rceil$, $S_2 = S_0 - S_1$; {number of strands in major and minor subbranches}

$r_1 := \max\left\{\min\left\{\sqrt{\frac{S_1}{S_0}}, M^{g,w}\right\}, m^{g,w}\right\}$ {branch length proportions dependant on strands}

$r_2 := \max\left\{\min\left\{\sqrt{\frac{S_2}{S_0}}, M^{g,w}\right\}, m^{g,w}\right\}$;

$L_1 := r_1 L_0$, $L_2 := r_2 L_0$; {relative length of subbranches}

$l_1 := k_l^{g,w} L_1$, $l_2 := k_l^{g,w} L_2$; {active subbranch length}

$\alpha_1 := k_c\sqrt{\frac{S_2}{S_0}}\alpha^{g,w}$, $\alpha_2 := \alpha^{g,w} - \alpha_1$; {branching angles}

$\alpha_x(t) := \sin(t + R_x) w_s (1 - k_f) l_0$; {animation of un-directed wind impact}

$\alpha_z(t) := \sin(t + R_z) w_s (1 - k_f) l_0$;

$\alpha_w := \frac{S_0}{S} w_g$; {animation of directed wind impact}

$\mathbf{M}_1 := \mathbf{R}_{\mathbf{w}_0}(\alpha_w)\mathbf{R}_z(\alpha_1 + \alpha_z(t))\mathbf{R}_x(\alpha_x(t))\mathbf{R}_y(\alpha_p)\mathbf{R}_{\mathbf{y}\times\mathbf{y}_m}(\alpha_m^{g,w})\mathbf{T}_y(l_0)\mathbf{M}_0$;

$\mathbf{M}_2 := \mathbf{R}_{\mathbf{w}_0}(\alpha_w)\mathbf{R}_z(\alpha_2 + \alpha_z(t))\mathbf{R}_x(\alpha_x(t))\mathbf{R}_y(\alpha_p)\mathbf{R}_{\mathbf{y}\times\mathbf{y}_m}(\alpha_m^{g,w})\mathbf{T}_y(l_0)\mathbf{M}_0$;

$\mathbf{M}_{m;1}^{-1} := \mathbf{R}_{\mathbf{y}\times\mathbf{y}_m}(-\alpha_m^{g,w})\mathbf{R}_y(-\alpha_p)\mathbf{R}_x(-\alpha_x(t))\mathbf{R}_z(-\alpha_1 - \alpha_z(t))\mathbf{M}_{m;0}^{-1}$; {refreshing inverse matrix for construction of gravimorphism vector, without considering wind impact}

$\mathbf{M}_{m;2}^{-1} := \mathbf{R}_{\mathbf{y}\times\mathbf{y}_m}(-\alpha_m^{g,w})\mathbf{R}_y(-\alpha_p)\mathbf{R}_x(-\alpha_x(t))\mathbf{R}_z(-\alpha_2 - \alpha_z(t))\mathbf{M}_{m;0}^{-1}$;

$\mathbf{M}_{w;1}^{-1} := \mathbf{R}_{\mathbf{y}\times\mathbf{y}_m}(-\alpha_m^{g,w})\mathbf{R}_y(-\alpha_p)\mathbf{R}_x(-\alpha_x(t))\mathbf{R}_z(-\alpha_1 - \alpha_z(t))\mathbf{R}_{\mathbf{w}_0}(-\alpha_w)\mathbf{M}_{w;0}^{-1}$; {refreshing inverse matrix for construction of directed wind vector}

$\mathbf{M}_{w;2}^{-1} := \mathbf{R}_{\mathbf{y}\times\mathbf{y}_m}(-\alpha_m^{g,w})\mathbf{R}_y(-\alpha_p)\mathbf{R}_x(-\alpha_x(t))\mathbf{R}_z(-\alpha_2 - \alpha_z(t))\mathbf{R}_{\mathbf{w}_0}(-\alpha_w)\mathbf{M}_{w;0}^{-1}$;

branchsegment($g+1$, $w+1$, S_2, L_2, l_2, $\mathbf{M}_2$, $\mathbf{M}_{m;2}^{-1}$, $\mathbf{M}_{w;2}^{-1}$); {minor branch development}

branchsegment(g, $w+1$, S_1, L_1, l_1, $\mathbf{M}_1$, $\mathbf{M}_{m;1}^{-1}$, $\mathbf{M}_{w;1}^{-1}$); {major branch development}

built recursively. From a procedural model for a tree, a geometry model is calculated using the briefly denoted Algorithm 1. Geometrical model is rendered using photo textures for final look of a tree. This model differs from many other models [1, 3, 10, 12, 16] since all of its parameters are fully numerically encoded and are fixed dimensionality. It is therefore especially suitable for parameter estimation using differential evolution.

2.3 Image-based Approaches to Modeling

Image-based approaches have the best potential to produce realistically looking plants, since they rely on images of real plants [8]. Also, little work has been done to design trees with the use of a general recognition from images without user interaction. In [2] an approach for recognition of procedural models is presented. However, the procedural models used in [2] were two-dimensional. Therefore, we extended their approach to the domain of three-dimensional procedural models suitable to model woody plants without user interaction.

3. Woody Plants Recognition by Differential Evolution

We have combined the jDE algorithm [4] and the numerically coded procedural model of woody plants from EcoMod framework [21, 23, 25]. Thereby, we recognize woody plant models from images by evolving the parameters of the procedural model. The fitness computation is based on the comparison of two-dimensionally rendered images. The fitness is better (i.e. takes smaller values) for images with greater similarity. The recognition method operates by encoding the parameters of the procedural model in genotype of the individual vector of jDE population. In the following, parts of the optimization procedure are described, i.e. the genotype encoding, genotype-phenotype mapping, and its fitness evaluation.

3.1 Genotype Encoding

An individual genotype vector $\mathbf{x}$ of jDE population represents the set of procedural model parameters, used in Algorithm 1. The dimensionality of evolved floating-point encoded parameters is $D = 4509$. Each parameter $x_{i,j} \in [0, 1]$ for all $i \in \{1..NP\}$ and $j \in \{1..D\}$ encodes the following parameters (for more explicit formulation of the parameters see [21]):

- number of strands of a tree
  $S = 400x_{i,0} + 10$ ($S \in [10, 410]$),
- height of base trunk
  $l_0^{0,0} = x_{i,1}10$ m ($l_0^{0,0} \in [0 \text{ m}, 10 \text{ m}]$),
- coefficient of branch thickness
  $k_d = 0.05x_{i,2}$ ($k_d \in [0, 0.05]$),
- phyllotaxis angle
  $\alpha_p = 360^\circ x_{i,3}$ ($\alpha_p \in [0^\circ, 360^\circ]$),
- branching ratio of subbranch strands distribution
  $k_s^{g,w} = 0.5x_{i,j} + 0.5$, $\forall j \in \{4, 753\}$ ($k_s^{g,w} \in [0.5, 1]$),
- branching angle between dividing subbranches
  $\alpha^{g,w} = 180^\circ x_{i,j}$, $\forall j \in \{754, 1503\}$ ($\alpha^{g,w} \in [0^\circ, 180^\circ]$),
- maximum relative subbranch to base branch length
  $M^{g,w} = 20x_{i,j}$, $\forall j \in \{1504, 2253\}$ ($M^{g,w} \in [0, 20]$),
- minimum relative subbranch to base branch length
  $m^{g,w} = 20x_{i,j}$, $\forall j \in \{2254, 3003\}$ ($m^{g,w} \in [0, 20]$),
- branch length scaling factor
  $k_l^{g,w} = 20x_{i,j}$, $\forall j \in \{3004, 3753\}$ ($k_l^{g,w} \in [0, 20]$),
- gravicentralism impact
  $k_c = x_{i,3754}$ ($k_c \in [0, 1]$),
- gravimorphism impact (i.e. gravitational bending of branches)
  $\alpha_m^{g,w} = 360^\circ x_{i,j} - 180^\circ$, $\forall j \in \{3755, 4504\}$ ($\alpha_m^{g,w} \in [-180^\circ, 180^\circ]$),
- enabling leaves display on a tree
  $B_l = \lfloor x_{i,4505} + 0.5 \rfloor$ ($B_l \in \{0, 1\}$),
- density of leaves
  $\rho_l = 30x_{i,4507}$ ($\rho_l \in \{0, 30\}$),
- size of leaves
  $l_l = 0.3x_{i,4506}$ ($l_l \in [0, 0.3]$), and
- leaf distribution type
  $l_{type} = 5x_{i,4508}$ ($l_{type} \in \{Spiral, Stacked, Staggered, Bunched, Conifer ous\}$),

where $g \in \{0, 15\}$, $w \in \{0, 50\}$, and each 750 real-coded parameters encode one matrix of a Gravelius and Weibull ordered parameter.

3.2 Genotype-phenotype Mapping

Our recognition method is based on recognition of two-dimensional images of woody plants $\mathbf{z}^*$, possibly taken by a digital camera. To compare the three-dimensional tree evolved with the use of genotype $\mathbf{x}$ to the reference image $\mathbf{z}^*$, the encoded D-dimensional genotype $\mathbf{x}$ must be transformed to its phenotype first. Phenotype is a rendered two-dimensional image $\mathbf{z}$ of a genotype $\mathbf{x}$ with the use of Algorithm 1. Images $\mathbf{z}^*$ and $\mathbf{z}$ are all of dimensionality $X \times Y$ pixels, where the reference image is scaled to the given resolution, if necessary. Both images are converted to black and white, where white (0) pixels mark background and black (1) pixels mark the material, e.g. wood. With the use of the conversion, the evolved procedural model is compared twice to the reference images, differing by camera view angle of $\beta = 90°$ along the trunk base. The latter is done to favor three-dimensional procedural models generation. If we denote the Algorithm 1 as function $\mathbf{g}$ then $\mathbf{z} = \mathbf{g}(\mathbf{x}, \beta)$.

3.3 Phenotype and Reference Image Comparison

The recognition success is measured by similarity of the reference original images and the generated rendered images of evolved parametrized procedural models. To measure similarity of these images we chose to compare the images pixel-wise as follows. For each pixel rendered as non-background in the evolved image, we compute the Manhattan distance to the nearest non-background pixel in the reference image, and vice-versa [2]. The sum of these distances accounts for fitness evaluation of each phenotype:

$$f(\mathbf{x}) = f(\mathbf{g}(\mathbf{x}, 0°), \mathbf{g}(\mathbf{x}, 90°)) = h(\mathbf{z}_1) + h(\mathbf{z}_2),$$

$$h(\mathbf{z}) = \sum_{x,y} m_1(z_{x,y}, z^*_{x,y}) + \sum_{x,y} m_1(z^*_{x,y}, z_{x,y}),$$

where m_1 denotes a function which computes a Manhattan distance to the nearest pixel in an image $\mathbf{z}^*$, being set to 1 (i.e. black, wood material).

4. Experimental Results

We have assessed the algorithm for tree recognition on an example tree, seen in Fig. 1 on the far right. The sampling rate dimension of the rendered parametrized procedural model was set to 250x250, the maximal number of strands in the tree to $S = 410$, and the maximal number of fitness evaluations (FEs) for jDE algorithm to $FEs = 10000$. The remaining parameters were kept as defaults in original algorithms from their literature.

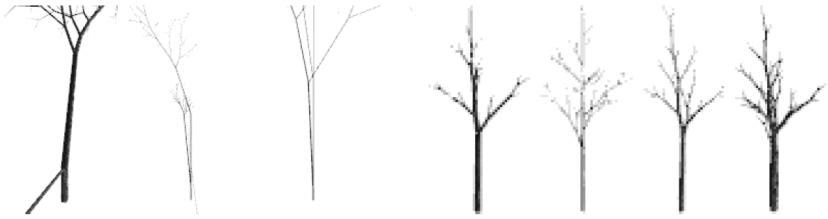


Figure 1. Rendered evolved parameterized procedural models at FEs ={1, 8, 18, 1992, 2727, 3230} ($NP = 100$, seed 1) and fifth, the reference image.

Final best evaluations obtained over 30 runs for different settings of NP in the evolutionary algorithm are seen in Fig. 2. The best average final best was obtained using $NP = 100$ with fitness of 1828.3. For population size of $NP = 100$, the jDE algorithm in 30 runs obtained the best fitness value of 1806, the worst being 1870, and the average of 1828.3 with standard deviation of 84.4. The sampled procedural models for run 1, with seed 1, for this test are seen in Fig. 1. The tree on the image is 2.5 m tall, 1 m for the first branch segment, therefore it only extends to a part of the image's canvas which is 25 m tall in total. Therefore, the images in Fig. 1 were zoomed to fit. Since we can design woody plants with a reliability, seen in Fig. 2 and obtain such models as seen in Fig. 1, we can conclude that the presented approach is viable for modeling of woody plants for computer animation by evolution of the numerically-coded procedural model.

5. Conclusions

We presented an approach to design woody plant geometrical models. To construct a geometrical model, we have used a parameterized procedural model. The parameters of the procedural model were evolved through the jDE differential evolution algorithm. The sampled procedural models were rendered with the use of the EcoMod framework. Rendered images were then compared to the reference source images, for recognition, to guide the optimization process. After the description of proposed approach, we demonstrated its experimental results by recognition of a sample woody plant model and statistical analysis of the obtained results.

In the future research, we would like to improve metrics for comparison of rendered and reference images. Multiple metrics could be used

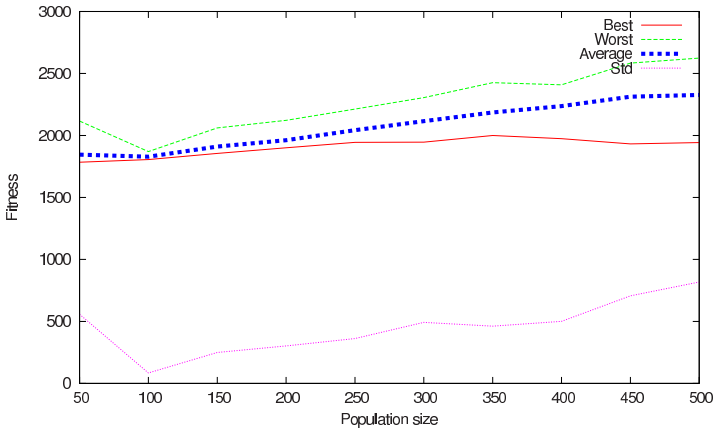


Figure 2. Algorithm performance, dependent on population size.

and combined with the use of multi-objective search [22], and possibly combined with interactive methods for optimization.

References

[1] M. Aono and T. Kunii. Botanical tree image generation. *IEEE Comput. Graph.*, 4(5):10–34, 1984.

[2] D. Beaumont and S. Stepney. Grammatical Evolution of L-systems. In *Proc. IEEE Congress on Evolutionary Computation*, pages 2446–2453, 2009.

[3] J. Bloomenthal. Modeling the mighty maple. In *Proc. 12st Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, pages 305–311, 1985.

[4] J. Brest, S. Greiner, B. Bošković, M. Mernik, and V. Žumer. Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems. *IEEE T. Evol. Comput.*, 10(6):646–657, 2006.

[5] J. Brest and M. Sepesy Maučec. Population size reduction for the differential evolution algorithm. *Appl. Intell.*, 29(3):228–247, 2008.

[6] J. Brest, A. Zamuda, B. Bošković, M. S. Maučec, and V. Žumer. Dynamic optimization using self-adaptive differential evolution. In *Proc. IEEE Congress on Evolutionary Computation*, pages 415–422, 2009.

[7] J. Brest, A. Zamuda, B. Bošković, and V. Žumer. An analysis of the control parameters adaptation in DE. In *Advances in Differential Evolution, Studies in Computational Intelligence*, 143:89–110, 2008.

[8] X. Chen, B. Neubert, Y. Q. Xu, O. Deussen, and S. B. Chang. Sketch-based Tree Modeling Using Markov Random Field. ACM T. Graphic., 27(5):109, 2008.

[9] V. Feoktistov. *Differential Evolution: In Search of Solutions*. Springer, 2006.

[10] M. Holton. Strands, gravity, and botanical tree imagery. *Comput. Graph. Forum*, 13(1):57–67, 1994.

[11] E. Mezura-Montes and B. C. Lopez-Ramirez. Comparing bio-inspired algorithms in constrained optimization problems. In *Proc. IEEE Congress on Evolutionary Computation (CEC)*, pages 662–669, 2007.

[12] B. Neubert, T. Franken, and O. Deussen. Approximate image-based tree-modeling using particle flows. *ACM T. Graphic.*, 26(3):88, 2007.

[13] P. E. Oppenheimer. Real time design and animation of fractal plants and trees. *Comp. Graph.*, 20(4):55–64, 1986.

[14] K. V. Price, R. M. Storn, and J. A. Lampinen. *Differential Evolution: A Practical Approach to Global Optimization*. Natural Computing Series. Springer, 2005.

[15] P. Prusinkiewicz and A. Lindenmayer. *The Algorithmic Beauty of Plants*. Springer, 1990.

[16] W. Reeves. Approximate and probabilistic algorithms for shading and rendering structured particle systems. *Proc. 12st Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, pages 313–322, 1985.

[17] D. Reinhardt, E.-R. Pesce, P. Stieger, T. Mandel, K. Baltensperger, M. Bennett, J. Traas, J. Friml, and C. Kuhlemeier. Regulation of phyllotaxis by polar auxin transport. *Nature*, 426:255–260, 2003.

[18] R. Storn and K. Price. Differential evolution – A simple and efficient heuristic for global optimization over continuous spaces. *J. Global Optim.*, 11:341–359, 1997.

[19] J. Weber and J. Penn. Creation and rendering of realistic trees. In *Proc. 22nd Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, pages 119–128, 1995.

[20] O. Yogev, A. A. Shapiro, and E. K. Antonsson. Computational evolutionary embryogeny. *IEEE T. Evol. Comput.*, 14(2):301–325, 2010.

[21] A. Zamuda. Modeling, Simulation and Rendering of Tree Ecosystems. Diploma thesis, Faculty of Electrical Engineering and Computer Science, University of Maribor, 2006.

[22] A. Zamuda. Self-adaptation of Control Parameters in Differential Evolution for Multiobjective Optimization Algorithm. MSc Thesis, Faculty of Electrical Engineering and Computer Science, University of Maribor, 2008.

[23] A. Zamuda, J. Brest, N. Guid, and V. Žumer. Construction of virtual trees within ecosystems with EcoMod tool. In *Proc. International Conference on Advances in the Internet, Processing, Systems, and Interdisciplinary Research (IPSI)*, 2006.

[24] A. Zamuda, J. Brest, N. Guid, and V. Žumer. Modelling, simulation, and visualization of forest ecosystems. In *Proc. IEEE Region 8 EUROCON 2007: International Conference on "Computer as a Tool"*, pages 2600–2606, 2007.

[25] A. Zamuda and D. Strnad. Interactive tool for modelling realistic animated trees. In *Proc. Thirteenth International Electrotechnical and Computer Science Conference (ERK)*, pages 11–14, 2004.

NEW EVIDENCES ON VARIABLE SELECTION WITH STOCHASTIC OPTIMIZATION ALGORITHM

Antonio Gargano
Department of Finnance
Università Bocconi, Milan, Italy
antonio.gargano@unibocconi.it

Abstract This paper adopts four stochastic optimization algorithms to perform model selection in linear regression models when the number of candidate variables is such that full enumeration of all possible models is impossible. Algorithms performance in maximizing several fitness functions are compared using different measures. The main conclusion is that performance differentials depend on the problem complexity, in terms of the number of local solutions, and on the measure used. In absolute terms, however, Genetic Algorithms and Simulated Annealing give the best results.

Keywords: Genetic algorithms, Optimization heuristics, Return predictability, Simulated annealing, Variable selection

1. Introduction

Variable selection is one of the core steps before fitting any econometric model, especially when dealing with multivariate ones. Given its central role, many different methodologies have been proposed.

The first ones being developed have been sequential and iterated procedures. Forward Selection and Backward Elimination [3] rely on sequential hypothesis testing of the coefficients significance. Lasso [19] and garrote [9], are shrinkage based techniques that contemporaneously estimate coefficients and select variables, by applying progressively stronger constraints on the estimates. Finally, the two most used bayesian algorithms in this framework are Stochastic Search Variable Selection (SSVS) [4] and Markov Chain Monte Carlo Model Comparison (MC^3) [13]. Finally, Principal Variables [12] and Diffusion Indexes [18], popular in a

forecasting framework, select variables by using data dimensionality reduction techniques such as principal components and factor analysis.

Despite such abundance and variety of approaches, as the number of candidate variables increases some of those procedures become infeasible. This is due to the well known dimensionality curse.

A possible remedy might come from Stochastic Optimization Algorithms (SOA). A natural way to link them to the variable selection problem is in couple with Information Criteria.

Despite their great potential, not much is known about their real performance in solving the variable selection problem in an econometric framework. On the contrary, in data mining this problem is known as feature selection and extensive research use SOA [7]. The aim of this paper is to fill this gap by investigating how those algorithms work, both in absolute and in relative terms.

The paper proceeds as follows. Section 2 briefly describes SOA and their implementation, Section 3 introduces the adopted methodology and illustrates the results, Section 4 concludes and proposes future improvements.

2. Theory

In presenting the algorithms, we follow the taxonomy in [20], whose first distinction is between local and global search algorithms. The classification is based on the way the algorithms organize their walks through the solutions space.

2.1 Local Search algorithms

The two local search algorithms adopted are Random Search and Multiple Random Search. As typically implemented, Random Search entails the following steps:

Algorithm 1 Random Search

1: Generate initial solution m^c. Calculate $f(m^c)$.
2: **while** stopping criteria not met **do**
3: Generate $m^n \in \Upsilon(m^c)$
4: **if** $f(m^n) < f(m^c)$ **then**
5: $m^c = m^n$
6: **end if**
7: **end while**

In case there are no local solutions this algorithm is expected to perform as well as global search ones with fewer iterations. It stops when

there is no m^n in the neighborhood of m^c that represents a better solution. Similarly, the Multiple Random Search algorithm might be formalized in the following way:

Algorithm 2 Multiple Random Search

```
1: Initialize Ψ
2: for ψ = 1 to Ψ do
3:    Generate initial solution m^c_ψ
4:    while stopping criteria not met do
5:       Generate m^n_ψ ∈ ϒ(m^c_ψ)
6:       if f(m^n_Ψ) < f(m^c_ψ) then
7:          m^c_ψ = m^n_ψ
8:       end if
9:    end while
10:   store m^*_ψ
11: end for
12: Select M^*_ψ s.t. Φ = [m^*_ψ | m^*_ψ > M^*_ψ] = ⊘
```

In the case of the variable selection problem, a solution m represents a model, a $1 \times k$ row vector of zeros and ones $[\ 1 \quad 0 \quad \ldots \quad 0 \quad 1_k\ ]$ where a 1 in the j_{th} position indicates that the j_{th} variable belongs to the subset of variables used to compute the objective function $f(\circ)$. M^*_ψ represents the final solution that the algorithm returns. In the following analysis, as in [4], the starting solution m^c is randomly generated according to the following density $p(m^c) = \prod_{j=1}^{k} \pi_j^{m_j^c}(1-\pi_j)^{m_j^c}$ where $\pi = 0.5$. A neighborhood of a current solution, $\Upsilon(m^c)$, is the set of all $1 \times k$ vectors with at least one and at most $\kappa < k$ elements different from m^c. For the empirical analysis κ has been set equal 1. Given a standard linear regression framework where y is the vector of the dependent variable, X the regressors matrix, $\widehat{\beta} = (X'X)^{-1}X'y$ the ordinary least square estimates, $\widehat{y} = X\widehat{\beta}$ the fitted values, $\widehat{e}$ the vector of residuals, and the log-likelihood $\widehat{\lambda} = -\frac{T}{2}(1+\ln(2\pi)+\ln(\frac{\widehat{e}'\widehat{e}}{T}))$, the fitness functions $f(\circ)$ to be maximized are:

1 Akaike Information Criterion (AIC): $\widehat{\lambda} - k$

2 Bayesian Information Criterion (BIC): $\widehat{\lambda} - \frac{\ln(T)k}{2}$

3 Adjusted R^2: $1 - \frac{\frac{\widehat{e}'\widehat{e}}{T-k-1}}{\frac{(y-\overline{y})'(y-\overline{y})}{T-1}}$

4 Mean Square Error: $-[\frac{1}{T}\sum_{t=1}^{T}(y_t - \widehat{y}_t)^2]$

5 Sign Criterion: $\frac{1}{T}\sum_{t=1}^{T}\left\{I_{y_t>0}I_{\widehat{y}_t>0} + (1 - I_{y_t>0)})(1 - I_{\widehat{y}_t>0})\right\}$

Those functions have been chosen because they are most frequently used for in-sample and out-of-sample model validation and because they entail different degrees of nonlinearity.

2.2 Global Search algorithms

2.2.1 Genetic Algorithms. Genetic Algorithms (GAs) by [10] and [5] are a subfamily of a wider random-guided search techniques called evolutionary algorithms [16]. As typically implemented, GAs involve the following steps:

Algorithm 3 Genetic Algorithm

```
Generate initial population P, initialize p_mut and p_cross
while stopping criteria not met do
  Select P' ⊂ P (mating pool) set P'' = ∅ (set of child)
  for i = 1 to n (population size) do
    Select individuals m^a and m^b at random from P'
    if u(0,1) < p_cross then
      cross-over and to produce m^c
    end if
    if u(0,1) < p_mut then
      mutate produced child m^c
    end if
    P'' = P'' ∪ m^c
  end for
  P = P''
end while
```

P and P'' are $n \times k$ matrices, where each row is a binary vector representing a solution. Similarly, P' is a submatrix of P whose row size depends on the parameters discussed below. When the solutions are taken individually as $1 \times k$ vectors, they are labeled with m. GA parameters are often chosen according to some general guidelines presented in literature and by calibrating, mainly by means of Monte Carlo simulations, the algorithm to the problem [1]. Following [2] the value for p_{cross} and p_{mut} have been set to 0.7 and 0.1 and as suggested by [16], they decrease over time. Uniform crossover and flipping mutation

are adopted. To improve the algorithm performance, elitism operator is used by setting the row size of P' 0.5 times the one of P; it implies that only the better half of the current population is used to breed the next generation. Finally, the stopping criterion is determined by a maximum number of generations to be performed, and is a decreasing function of the population size. The row size of P is around $2k$ where k is the number of regressors, and number of generation has been set to 50, with those value increasing according to problem difficulty.

2.2.2 Simulated Annealing. It is based on ideas from statistical mechanics and motivated by an analogy to the behavior of physical systems in the presence of a heat bath [11]. This approach avoids entrapment in poor local optima by allowing an occasional uphill move. It is done under the influence of a random number $x \sim u(0,1)$ and a control parameter T, called the temperature, that affect the probability of uphill move $e^{-\frac{\Delta}{T}}$. As typically implemented, the simulated annealing approach involves the following steps:

Algorithm 4 Simulated Annealing

```
1: Generate a current solution m^c, initialize L, r < 1 and T
2: for l = 1 to L do
3:   while stopping criteria not me do
4:     Select a new candidate solution m^n ∈ ϒ(m^c)
5:     Compute Δ = f(m^n) − f(m^c)
6:     if Δ ≤ 0 then
7:       m^c = m^n
8:     else
9:       if u(0,1) < e^{−Δ/T} then
10:        m^c = m^n
11:      end if
12:    end if
13:  end while
14:  Set T = rT
15: end for
```

While $r = 0.005$ and $L = 20$ have been found to fit for most of functions to be maximized, initial T has been found strongly depending on the function characteristics, in particular on its scale. It must be carefully chosen because if it is too high, the algorithm converges too late. If it is too low, it does not explore a sufficiently broad area of the solution space, in both cases it returns a local solution far from the global one.

3. Results

As common practice in this kind of studies [14], two analysis are performed: with real and randomly generated data.

3.1 Monte Carlo Simulations

3.1.1 Absolute performance evaluation. The following simulations are aimed to investigate how three factors affect algorithms' performances. They are N, the number of candidate variables, t, the number of observations, and Σ the variance covariance matrix of the regressors. The two values that t can assume are aimed to mimic a small and a large sample, while the values of Σ are aimed to simulate a situation of no and one of medium multicollinearity (i.e. the degree of pairwise correlation between variables). Some more details are required for N. In order to evaluate the algorithms in absolute terms, the global solution must be known. This requires tabulating the empirical cumulated distribution function, $\Phi(\circ)$, of all the possible 2^N models and $\Phi^{-1}(1)$, its maximum. If N is too high, the analysis becomes infeasible, due to the excessive computational burden. If it is too low, it becomes useless since there would be no real need to use SOA. Therefore the two values that N can assume, 14 and 17, have been chosen in such a way to balance this trade-off between feasibility and significance.

A setting is defined by a vector $\Theta = [N_{i\in 1,2}, t_{j\in 1,2}, \sum_{l\in 1,2}]$, where $N \in [14, 17]$, $t \in [35, 150]$ and $\sum \in [I_k, M]$; I is the identity matrix while M is such that it induces a pairwise correlation among variables of about 0.5. Given those six values, a total of eight settings are simulated, from $\Theta = [14, 35, I_k]$ to $\Theta = [17, 150, M]$. For each setting, the full regressors matrix is generated such that $X_{t_j \times N_i} \sim N(\mu, \Sigma_l)$, from it the dependent variable is constructed according to $y_t = X_{t_j \times n < N_i}\beta + \varepsilon_t$, finally all $f(m_i)$ for $i = 1$ to 2^{N_i} with $m_i \subseteq M_{N_i}$ are computed. Once the full empirical distribution of the values assumed by the function is obtained, the algorithms are run 50 times and summary statistics $\mu_a = \frac{1}{50}\sum_{i=1}^{50} c(f(m_{a,i}))$ are computed, where a is the subscript to identify the algorithm and c are criteria described below.

In order to judge the algorithms performance, eight criteria have been designed, with the intent of catching different aspects.

1 **Fitness:** It is the value of the maximized function $f(m_a^*)$, where m_a^* is the model found by the a_{th} algorithm.

2 **Rank:** It is the rank of $f(m_a^*)$ out of all the possible 2^{N_i}.

3 **Percentile:** It is the percentile $\Phi(f(m_a^*))$ of $f(m_a^*)$ where $\Phi(\circ)$ represents the empirical cumulative distribution function.

4 **Best:** It is an indicator variable that takes value 1 if $f(m_a^*) = f(M^*)$, where M^* is the model at which the function reaches the global max: $\Phi^{-1}(1)$.

5 **Percentile 99.5:** It is an indicator variable that takes value 1 if $99.5 \leq \Phi(f(m_a^*))$.

6 **Distance:** $\Delta = f(M^*) - f(m_a^*)$ where M^* is the model at which the function reaches the global max: $\Phi^{-1}(1)$.

7 **Efficiency:** $\frac{\nu_a}{2^N}$ where ν_a is the number of times the cost function is evaluated by the a_{th} algorithm and 2^N represents the maximum number of model combinations given N variables.

8 **Time:** It is the time needed to perform ν_a.

The following results are based on the maximization of AIC and BIC criteria as defined in Section 2.

Table 1. BIC: Rank

N	*Sigma*	*t*	*RandomSearch*	*MRandomSearch*	*GA*	*SA*
14	I	35	3.9	1	1	1
14	I	150	4.8	1.1	1	1
14	M	35	4.3	1	1	1.1
14	M	150	6	1.3	1	1
17	I	35	4.6	2	1.1	1.9
17	I	150	34.8	2.2	1.6	1.7
17	M	35	30.6	2.1	1	1
17	M	150	28.4	1.9	1.1	1.3
		Mean	14.68	1.58	1.10	1.25

Tables 1 and 2 display the average rank for each setting. Genetic algorithms, Simulated Annealing and Multiple Random Search outperform a simple Random Search whose average rank, across all experiments, is around 13. In particular, Genetic Algorithms are always able to find the global solution when there are 14 variables and $2^{14} = 16{,}384$ possible solutions.

Tables 3 and 4 display the success rate, i.e. the percentage of times the algorithm converges to the global solution. It is calculated as the mean of Criterion 4 illustrated before. From this point of view, Genetic

Table 2. AIC: Rank

N	*Sigma*	*t*	*RandomSearch*	*MRandomSearch*	*GA*	*SA*
14	I	35	3	1	1	1.2
14	I	150	4.7	1.2	1.2	1
14	M	35	7.2	1.1	1	1.3
14	M	150	5	1.3	1	1
17	I	35	23.6	1.9	1	1.6
17	I	150	25.2	3.9	1.6	1.7
17	M	35	17.2	1.5	1.2	1.3
17	M	150	21	1.9	1.1	1.4
		Mean	13.36	1.73	1.14	1.31

Table 3. BIC: Success rate

N	*Sigma*	*t*	*RandomSearch*	*MRandomSearch*	*GA*	*SA*
14	I	35	0.6	1	1	1
14	I	150	0.3	0.9	1	1
14	M	35	0.3	1	1	0.9
14	M	150	0	0.8	1	1
17	I	35	0.2	0.3	0.9	0.6
17	I	150	0	0.4	0.6	0.2
17	M	35	0	0.2	1	1
17	M	150	0	0.7	0.9	0.7
		Mean	0.15	0.66	0.93	0.80

Table 4. AIC: Success rate

N	*Sigma*	*t*	*RandomSearch*	*MRandomSearch*	*GA*	*SA*
14	I	35	0.6	1	1	0.8
14	I	150	0.3	0.8	0.8	1
14	M	35	0	0.9	1	0.7
14	M	150	0.3	0.8	1	1
17	I	35	0	0.4	1	0.6
17	I	150	0.2	0	0.7	0.4
17	M	35	0	0.5	0.8	0.7
17	M	150	0.3	0.7	0.9	0.7
		Mean	0.14	0.61	0.90	0.74

Algorithms and Simulated Annealing have a quite good performance with a success rate around 0.9 and 0.8 respectively, Multiple Random Search slightly below with 0.65, with Random Search being the worst with just 0.14.

Table 5. BIC: Distance

N	*Sigma*	*t*	*RandomSearch*	*MRandomSearch*	*GA*	*SA*
14	I	35	2.73	0	0	0
14	I	150	65.12	6.21	0	0
14	M	35	4.70	0	0	0.13
14	M	150	31.04	3.26	0	0
17	I	35	7.58	5.52	0.78	3.47
17	I	150	31.16	3.4	1.49	0.92
17	M	35	24.12	9.97	0	0
17	M	150	43.44	4.96	0.33	0.98
		Mean	26.24	4.17	0.32	0.69

Tables 5 and 6 display the average distance from the global solution in terms of the function to be maximized, it is based on the Criterion 6 discussed above. The tables show that when Random Search and Multiple Random Search fail in finding the best solution, the magnitude of their error is bigger than Genetic Algorithms and Simulated Annealing. Random Search has the worst performance, with an average distance of 25. The difference in the other three algorithms is more evident looking at Table 5, rather than Table 6; nevertheless Multiple Random Search preforms worse in both cases.

Table 6. AIC: Distance

N	*Sigma*	*t*	*RandomSearch*	*MRandomSearch*	*GA*	*SA*
14	I	35	2	0	0	0.03
14	I	150	65.69	12.72	12.72	0
14	M	35	6.34	0.1	0	0.3
14	M	150	25.86	3.71	0	0
17	I	35	13.13	5.20	0	8.85
17	I	150	24.48	7.34	1.64	0.9
17	M	35	22.95	6.23	2.49	3.74
17	M	150	31.93	3.86	0.33	2.16
		Mean	24.05	4.90	2.15	2.03

In the light of these results, it seems that GAs and SA clearly outperform the other two algorithms. However, this gap tends to shrink if one looks at the average percentile in Tables 7 and 8.

Table 7. BIC: Percentile

N	*Sigma*	*t*	*RandomSearch*	*MRandomSearch*	*GA*	*SA*
14	I	35	0.999822998	1	1	1
14	I	150	0.999768066	0.9999939	1	1
14	M	35	0.999798584	1	1	0.9999939
14	M	150	0.999694824	0.9999817	1	1
17	I	35	0,999972534	0.9999924	0.999999	0.9999931
17	I	150	0.999742126	0.9999908	0.999995	0.9999924
17	M	35	0.999774170	0.9999916	1	1
17	M	150	0.999790955	0.9999931	0.999999	0.9999977
		Mean	0.999796	0.999993	0.999999	0.999997

Although Random Search and Multiple Random Search have a lower success rate, their solutions still lie well above the 95^{th} percentile. Therefore, if the best 0.05×2^{N_i} solutions are considered being a satisfactory result, these two algorithms represent a good alternative given that they require less iterations as shown in the following tables.

They display the efficiency measured according to Criterion 7. This result show how the better performances of Genetic Algorithms and Simulated Annealing are based on a larger number of iterations. As the problem complexity increases the number of iterations increases too, but with a lower rate; therefore with 14 variables they perform 0.22 and 0.35 out of the total 2^{N_i}, but just 0.027 and 0.053 with $N = 17$.

Table 8. AIC: Percentile

N	*Sigma*	*t*	*RandomSearch*	*MRandomSearch*	*GA*	*SA*
14	I	35	0.99987793	1	1	0.9999878
14	I	150	0.99977417	0.9999873	0.999988	1
14	M	35	0.999621582	0.9999933	1	0.9999817
14	M	150	0.999755859	0.9999817	1	1
17	I	35	0.999827576	0.9999931	1	0.9999763
17	I	150	0.999815369	0.9999779	0.999995	0.9999947
17	M	35	0.999876404	0.9999962	0.999998	0.9999977
17	M	150	0.999847412	0.9999931	0.999999	0.9999969
		Mean	0.999800	0.999990	0.999998	0.999992

Table 9. BIC: Efficiency

N	*Sigma*	*t*	*RandomSearch*	*MRandomSearch*	*GA*	*SA*
14	I	35	0.006	0.060	0.220	0.350
14	I	150	0.006	0.052	0.220	0.350
14	M	35	0.006	0.054	0.220	0.350
14	M	150	0.006	0.053	0.220	0.350
17	I	35	0.001	0.010	0.027	0.053
17	I	150	0.001	0.009	0.027	0.053
17	M	35	0.001	0.009	0.027	0.053
17	M	150	0.001	0.009	0.027	0.053
		Mean	0.003	0.03	0.12	0.20

Table 10. AIC: Efficiency

N	*Sigma*	*t*	*RandomSearch*	*MRandomSearch*	*GA*	*SA*
14	I	35	0.006	0.059	0.220	0.350
14	I	150	0.006	0.052	0.220	0.350
14	M	35	0.006	0.056	0.220	0.350
14	M	150	0.006	0.053	0.220	0.350
17	I	35	0.001	0.009	0.027	0.053
17	I	150	0.001	0.008	0.027	0.053
17	M	35	0.001	0.009	0.027	0.053
17	M	150	0.001	0.009	0.027	0.053
		Mean	0.003	0.03	0.12	0.20

3.1.2 Relative performance evaluation. The procedure for relative performance is similar to the one described in previous subsection. The main difference is that the complexity of the problem has been increased by setting $N = 35$ for a total of $2^{35} = 34{,}359{,}738{,}368$ possible models. Since it is not feasible to evaluate the full cumulative distribution function, only a comparison in relative terms is possible. This requires also a slight change in the steps of the Monte Carlo simulation. A setting is still defined by a vector $\Theta = [N, t_{j\in 1,2}, \sum_{l\in 1,2}]$ with $N = 35$, $t \in [70, 250]$ and $\sum \in [I_k, M]$. However, instead of generating the data once for each setting, they are generated 50 times and algorithms are run once for each generation, in order to still have 50 total values to compute the summary statistics. Consequently, the criteria are slightly different. Percentile and Percentile 99.5 can not be computed, the rank is not out of 2^{N_i} but out of a, the number of algorithms, finally M^* is

no longer the model at which the function reaches the global max but simply the highest among the ones found by the algorithms.

Table 11 displays results for AIC while Table 12 for adjusted R^2. Results from the first table are in line with the one from previous subsection, in the sense that Genetic Algorithms and Simulated Annealing perform better than the other two algorithms with the GAs being the best most of the times.

Table 11. AIC

N	*Sigma*	*t*	*RandomSearch*	*MRandomSearch*	*GA*	*SA*	
35	I	70	2156.2	2164.7	2171.1	2169.2	Fit
35	I	70	15.3	6.8	0.4	2.3	Dist
35	I	70	3.9	2.8	1.2	1.7	Rank
35	I	70	0	0.1	0.8	0.5	Best
35	I	70	0.000000014	0.000000067	0.000000314	0.000000877	Eff
35	I	250	7738	7755.3	7786.2	7781	Fit
35	I	250	52	34.7	3.8	9	Dist
35	I	250	3.8	3.2	1.3	1.7	Rank
35	I	250	0	0	0.7	0.3	Best
35	I	250	0.000000012	0.000000067	0.000000314	0.000000877	Eff
35	M	70	2035.9	2050.1	2069.7	2067.1	Fit
35	M	70	34.8	20.6	1.1	3.6	Dist
35	M	70	3.7	3.2	1.2	1.7	Rank
35	M	70	0	0	0.8	0.4	Best
35	M	70	0.000000014	0.000000070	0.000000314	0.000000877	Eff
35	M	250	7422.5	7470.9	7469.1	7490	Fit
35	M	250	78.6	30.2	5.1	11.1	Dist
35	M	250	4	2.6	1.4	1.9	Rank
35	M	250	0	0	0.8	0.3	Best
35	M	250	0.000000024	0.000000066	0.000000314	0.000000877	Eff

Table 12 shows one important point. When the function to maximize has just one or very few local maxima, no matter how the problem is complex in terms of number of possible solutions, all the algorithms will have very similar results.

The main conclusions from both absolute and relative evaluations are the followings. In absolute terms, SOA have quite satisfactory performances. In relative terms GAs and SA seem to outperform the others, with GAs having the best performance. Local search algorithms are faster and might still represent a good alternative when having larger set of satisfactory solutions. Nevertheless, they have a big pitfall that make them less preferable – they cannot be controlled. GAs and SA instead can be improved and better tuned (increasing the population or the number of generations in GAs, and lowering the decaying factor for SA), in case they do not provide satisfactory results. The last observation is that the harder the problem, in terms of local solutions, the clearer the difference in performance between local and global algorithms.

Table 12. Adjusted R^2

N	*Sigma*	*t*	*RandomSearch*	*MRandomSearch*	*GA*	*SA*	
35	I	70	1.0	1.0	1.0	1.0	Fit
35	I	70	0	0	0	0	Dist
35	I	70	1	1	1	1	Rank
35	I	70	1	1	1	1	Best
35	I	70	0.000000011	0.000000059	0.000000314	0.000000877	Eff
35	I	250	1.0	1.0	1.0	1.0	Fit
35	I	250	0	0	0	0	Dist
35	I	250	1	1	1	1	Rank
35	I	250	1	1	1	1	Best
35	I	250	0.000000011	0.000000059	0.000000314	0.000000877	Eff
35	M	70	1.0	1.0	1.0	1.0	Fit
35	M	70	0	0	0	0	Dist
35	M	70	1	1	1	1	Rank
35	M	70	1	1	1	1	Best
35	M	70	0.000000012	0.000000060	0.000000314	0.000000877	Eff
35	M	250	1.0	1.0	1.0	1.0	Fit
35	M	250	0	0	0	0	Dist
35	M	250	1	1	1	1	Rank
35	M	250	1	1	1	1	Best
35	M	250	0.000000011	0.000000060	0.000000314	0.000000877	Eff

3.2 Real Data Analysis

Real data analysis is based on a popular dataset in return predictability literature. The variables are the ones used in [6] plus some others added by the author that represent other less known predictors available in literature. The final dataset has 19 time series and contains monthly observations of macro, financial and accounting variables that span from January 1961 to December 2008. The total number of possible models is 524,288. Results for the functions described in Section 2 are displayed. They are grouped into three categories according to the difficulty of the problem. Table 13 and Table 14 show the result for BIC and AIC. They confirm that Genetic Algorithms and Simulated Annealing perform better and that performance differentials shrink towards zero when looking at Percentile and Percentile 99.5

Table 15 show results for Mean Square Error and Table 16 for Adjusted R^2, they represent easy maximization problem because all the four algorithms are able to find the best solution out of the 524,288 available.

Table 17 displays results for Sign Criterion. It is apparently the hardest function to be maximized. Only Genetic Algorithms are able to systematically find the global solutions against a success rate of 0.72, 0.14 and 0.08 of the other three algorithms.

Table 13. BIC

	RandomSearch	*MRandomSearch*	*GA*	*SA*
Fitness	1165.95	1166.33	1166.39	1166.39
Rank	7	2.6	1	1
Percentile	1	1	1	1
Best	0.6	0.8	1	1
Percentile 99.5	1	1	1	1
Distance	0.44	0.06	0	0
Efficiency	0.0004	0.0014	0.0114	0.0109
Time	0.1	0.3	3.1	2.7

Table 14. AIC

	RandomSearch	*MRandomSearch*	*GA*	*SA*
Fitness	1142.82	1143.85	1143.85	1143.85
Rank	61.5	1	1	1
Percentile	0.9999	1	1	1
Best	0.86	1	1	1
Percentile 99.5	1	1	1	1
Distance	1.3	0	0	0
Efficiency	0.0004	0.0015	0.0114	0.0019
Time	0.001	0.009	0.027	0.053

Table 15. Mean Square Error

	RandomSearch	*MRandomSearch*	*GA*	*SA*
Fitness	-0.0010	-0.0010	-0.0010	-0.0010
Rank	1	1	1	1.1
Percentile	1	1	1	1
Best	1	1	1	0.88
Percentile 99.5	1	1	1	1
Distance	0	0	0	0
Efficiency	0.0004	0.0015	0.0114	0.0143
Time	0.1	0.4	3.6	4.7

4. Conclusions

This paper deals with variable selection in linear regression models using stochastic algorithms. Four algorithms (Random Search, Multiple Random Search, Genetic Algorithms and Simulated Annealing) are

Table 16. Adjusted R^2

	RandomSearch	*MRandomSearch*	*GA*	*SA*
Fitness	0.4283	0.4283	0.4283	0.4283
Rank	1	1	1	1
Percentile	1	1	1	1
Best	1	1	1	1
Percentile 99.5	1	1	1	1
Distance	0	0	0	0
Efficiency	0.0004	0.0016	0.0114	0.0064
Time	0.1	0.3	3.2	2.9

Table 17. Sign Criterion

	RandomSearch	*MRandomSearch*	*GA*	*SA*
Fitness	0.756	0.765	0.770	0.769
Rank	12328.1	173.9	1	4.6
Percentile	0.9765	0.9997	1	1
Best	0.08	0.14	1	0.72
Percentile 99.5	0.6	1	1	1
Distance	0.01	0.01	0	0
Efficiency	0.0002	0.0007	0.0114	0.0095
Time	0.008	0.1	3.2	3.4

tested with respect to eight criteria on simulated and real data. Both Monte Carlo and Real data analysis confirm the same results. In accordance with the literature, Genetic Algorithms and Simulated Annealing perform well in absolute terms with the former being the best in relative terms. Contrary to previous evidence, performance differentials strongly depend on the function to be maximized and on the measure used to compare the algorithms. One limit of this study is that it restricts the analysis to linear regression. It would be interesting in future studies to investigate the algorithms' behavior in more complicated problems involving VAR or even nonlinear models.

References

[1] S. H. Chen, ed. *Genetic Algorithms and Genetic Programming in Computational Finance*. Spring-Verlag, 2002.

[2] J. De Jong. An analysis of the beavior of a class of genetic adaptive systems. Doctoral Dissertation, University of Michigan, 1975.

[3] M. A. Efroymson. *Multiple Regression Analysis*. Wiley, 1960.

[4] E. I. George and R. E. McCulloch. Variable selection via Gibbs sampling. *J. Am. Stat. Assoc.*, 88(423):450–472, 1993.

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

[6] A. Goyal and I. Welch. A comprehensive look at the empirical performance of equity premium prediction. *Rev. Financ. Stud.*, 21(4):1455-1508, 2008.

[7] I. Guyon and A. Elisseff. An introduction to variable and feature selection. *J. Mach. Learn.*, 3:1157-1182, 2003.

[8] R. Hocking. The analysis and selection of variables in linear regression. *Biometrics*, 32:1–49, 1976.

[9] A. Hoerl and R. Kennard. Ridge regression: biased estimation for nonorthogonal problems. *Technometrics*, 42(1):80–86, 2000.

[10] J. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan press, 1974.

[11] S. Kirkpatrick. Optimization by simulated annealing: quantitative studies. *J. Stat. Phys.*, 34(5):975–986, 1984.

[12] G. P. McCabe. Principal variables. *Technometrics*, 26(2):34–67, 1984.

[13] A. Raftery, D. Madigan, and J. A. Hoeting. Bayesian model averaging for linear regression models. *J. Am. Stat. Assoc.*, 92(437):230–247, 1997.

[14] D. E. Rapach and W. E. Wohar. Forecasting the recent behavior of US business fixed investment spending: an analysis of competing models. *J. Forecasting*, 26(1):33–51, 2007.

[15] G. E. Schwarz. Estimating the dimension of a model. *Ann. Stat.*, 6(2):461–464, 1978.

[16] S.N Sivanandam and S. Deepa, *Introduction to Genetic Algorithms*. Springer, 2008.

[17] J. C. Spall. *Introduction to Stochastic Search and Optimization*. Wiley, 2003.

[18] J. H. Stock and M. W. Watson. Macroeconomic forecasting using diffusion indexes. *J. Bus. Econ. Stat.*, 20(2):147–162, 2002.

[19] R. Tibshirani. Regression shrinkage and selection via the lasso. *J. Royal Stat. Soc. B*, 58(1):65–83, 1996.

[20] P. Winker and M. Gilli. Applications of optimization heuristics to estimation and modelling problems. *Comput. Stat. Data An.*, 52(15):19–31, 2007.

[21] D. H. Wolpert and W. G. Macready. No free lunch theorems for optimization. *IEEE T. Evol. Comput.*, 1 (1):67–82, 1997.