# BIOINSPIRED OPTIMIZATION METHODS AND THEIR APPLICATIONS

# BIOINSPIRED OPTIMIZATION METHODS AND THEIR APPLICATIONS

## Proceedings of the Student Workshop on Bioinspired Optimization Methods and their Applications, BIOMA 2014

13 September 2014, Ljubljana, Slovenia

Edited by
JURIJ ŠILC
ALEŠ ZAMUDA

Jožef Stefan Institute, Ljubljana

**Jožef Stefan Institute**
Ljubljana, Slovenia

Editors: Jurij Šilc, Aleš Zamuda

Cover design by Studio Design Demšar, Škofja Loka
Logo design by Gregor Papa
Printed by Tiskarna Artelj, Ljubljana
Published by Jožef Stefan Institute, Ljubljana, Slovenia, September 2014
Circulation: 50 copies

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)

STUDENT Workshop on Bioinspired Optimization Methods and their Applications (2014 ; Ljubljana)
Bioinspired optimization methods and their applications : proceedings of the Student Workshop on Bioinspired Optimization Methods and their Applications - BIOMA 2014, 13 September 2014, Ljubljana, Slovenia / edited by Jurij Šilc, Aleš Zamuda. - Ljubljana : Jožef Stefan Institute, 2014

ISBN 978-961-264-068-2
1. Gl. stv. nasl. 2. Šilc, Jurij

275097088

# Contents

# Preface

The possibly changing and uncertain environment attracts and retains the fittest members of biological populations, which accumulate experience and improve, from adapting and competing among themselves. Their material of experience is exchanged and propagated from iteration to iteration according to the laws of nature. Relying on elementary activities of individuals, societies of these biological populations exhibit complex emergent behaviors. Assemblies of genes, insects, bird flocks, and many other fascinating natural phenomena have been a rich source of inspiration in computer algorithms design for decades. Specifically, optimization is an area where these techniques are studied and exercised with particular practical success.

As a result, the family of bioinspired algorithms under the Bioinspired Optimization Methods and its Applications (BIOMA) includes the evolutionary algorithms, genetic algorithms, evolution strategies, evolutionary programming, genetic programming, ant colony optimization, particle swarm optimization, artificial immune systems, and related bioinspired methods and their applications in science, engineering, and business. They were designed to overcome the drawbacks of traditional algorithms in demanding application scenarios including those where little, if any, information is available to assist problem solving. The emerging challenges inspire new methods to be delivered and existing ones being introduced for specific tasks.

This volume contains recent theoretical and practical contributions to the field of bioinspired optimization presented at the Student Workshop on Bioinspired Optimization Methods and their Applications (BIOMA 2014), held at the Ljubljana Exhibition and Convention Centre, Slovenia, on 13 September 2014.

Encouraged by the success of the previous BIOMA conferences organized in 2004, 2006, and 2008 as part of the Information Society Multiconference and 2010 and 2012 when BIOMA continued its own way, this time BIOMA conference takes part of the 13th International Conference on Parallel Problem Solving from Nature (PPSN 2014). BIOMA con-

tinues 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. This year, basis are the student papers, and therefore BIOMA is organized as a workshop to thrive new contributors in the field.

Each paper submitted to the conference was reviewed by two members of the international program committee. In the reviewing procedure, 11 papers were selected for presentation at the conference and publication in the proceedings. They were contributed by 31 (co)authors coming from Russian Federation, Slovenia, Turkey, and United Kingdom.

Theoretical and algorithmic studies presented at the conference address a variety of issues in bioinspired optimization: constraint optimization, multi-objective optimization, uncertain variables, parameter control, self-adaptation, population structuring and sizing, multiple populations, stagnation re-initialization, local search, memetic operators, parallel algorithms, distributed optimization architecture, application problem encoding, evolutionary operators definition, optimization algorithm proposition, evaluation function formalization, sub-function aggregation, and problem dimension reduction.

The applied work reports come from a number of interesting domains: multi-objective min-max optimization of design budget, optimal stock portfolio selection, production planning, spread spectrum radar polyphase code design, tersoff radar function minimization, automated slogan production, facial composite construction, spatial light distribution model, CUDA implementation, triangular brushstrokes composed image evolution, finite-state machine inference, and power generating units commitment.

The proposed theoretical mechanisms and realized challenging applications are balanced into four workshop sessions. The BIOMA 2014 workshop concludes with a general discussion, which aims to spring debates on interesting emerging topics from bioinspired optimization methods and their applications, with especial emphasis on future challenges in energy, finance, computational creativity and multimedia, computer vision, machine inference, and other.

Technical sponsor of the BIOMA conference is the Jožef Stefan Institute. Organized under the PPSN 2014 as a workshop, the following sponsors are acknowledged: B2, d.o.o., Flaška, d.d., and Kolektor Group, d.o.o. The partners participating in the conference organization in various ways are the City of Ljubljana, Ljubljana Exhibition and Convention Centre, Slovenian Artificial Intelligence Society, and Toleranca marketing, d.o.o.

Our thanks go to the conference sponsors and partners, members of the program and organizing committees, session chairs, 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, 1 September 2014*

Jurij Šilc and Aleš Zamuda

## Program Committee

Jurij Šilc, *Chair*, Jožef Stefan Institute, Ljubljana, Slovenia
Aleš Zamuda, *Chair*, University of Maribor, Slovenia

Janez Brest, University of Maribor, Slovenia
Dirk Büche, MAN Diesel & Turbo Schweiz AG, Switzerland
Rolf Drechsler, DFKI - Cyber-Physical Systems, Bremen, Germany
Bogdan Filipič, Jožef Stefan Institute, Ljubljana, Slovenia
Shih-Hsi "Alex" Liu, California State University, Fresno, USA
Nalini N, Nitte Meenakshi Institute of Technology, Bangalore, India
Gregor Papa, Jožef Stefan Institute, Ljubljana, Slovenia
Franciszek Seredynski, Cardinal Stefan Wyszynski University in Warsaw, Poland
Jim Tørresen, University of Oslo, Norway
Xin-She Yang, Middlesex University, London, UK

## Organizing Committee

Gregor Papa, *Chair*, Jožef Stefan Institute, Ljubljana, Slovenia

Jurij Šilc, Jožef Stefan Institute, Ljubljana, Slovenia
Aleš Zamuda, University of Maribor, Slovenia

# Contributing Authors

**Eren Akca** received his B.Sc. degree in Electronics and Communication Engineering from Çankaya University of Ankara, Turkey, in 2011. He has also a Major degree in Computer Engineering from Çankaya University, Ankara. He is currently a M.Sc. student at Ankara University, Department of Electrical-Electronics Engineering. His research interests include software technologies and GPGPU.

**Simone Alicino** received his M.Sc. degree in Aerospace Engineering from the University of Pisa, Italy, in 2009. He is currently a Ph.D. student at the University of Strathclyde, Glasgow, United Kingdom, Faculty of Engineering. His research interests include space systems, optimization, and uncertainty quantification.

**Ökkes Tolga Altınöz** received his M.Sc. degree in Electrical-Electronics Engineering from Hacettepe University, Ankara, Turkey, in 2010. He is currently a Ph.D. student at Ankara University, Department of Electrical-Electronics Engineering. His research interests include optimization and control theory. He is a member of IEEE.

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

**Miran Brezočnik** received the Ph.D. degree from the Faculty of Mechanical Engineering at the University of Maribor, Slovenia, in 1998. He is a Full Professor at the Production Engineering Institute. His main scientific interests include: intelligent manufacturing and assembly systems, intelligent machines and systems, advanced production tech-

nologies, machine learning (evolutionary computation methods, neural networks, swarm intelligence, gravitational search algorithm), modeling and optimization of different systems and processes in manufacturing using intelligent systems. He is also Editor-in-Chief of the international scientific journal Advances in Production Engineering & Management.

**Domen Butala** received his M.Sc. in Financial Mathematics from University of Ljubljana, Slovenia, in 2014 where he studied the optimization methods for a power systems. He is currently working as a quantitative power analyst in a power trading company ElectroRoute in Dublin, Ireland, where we develops different mathematical methods including stochastic optimization and stochastic processes.

**Daniil Chivilikhin** received his M.Sc. degree in Applied mathematics and informatics from the ITMO University, Saint-Petersburg, Russian Federation, in 2013. He is currently a Ph.D. student at ITMO University, Faculty of Information technologies and programming. His research interests include evolutionary computation, ant colony algorithms, and searh-based software engineering.

**Nejc Cvörnjek** received his B.Sc. degree in Industrial engineering from the University of Maribor, Slovenia, in 2010. He is a parallel master degree student at the University of Maribor at Faculty of Mechanical engineering-Faculty of Economics and Business and Faculty of Health Science. His research interests include intelligent systems in finance, biology, medicine and health science.

**Marilena Di Carlo** received her M.Sc. degree in Aerospace Engineering from the University of Pisa, Italy, in 2012. She is currently a Ph.D. student at the University of Strathclyde (Glasgow, UK) in the Department of Mechanical and Aerospace Engineering. Her research interests include global optimization, evolutionary computation and low-thrust space trajectory optimization.

**Juan Esteban Diaz Leiva** is a current Ph.D. student in Business and Management at the University of Manchester, United Kingdom. He received his M.Sc. degree in Food and Resource Economics from the University of Bonn, Germany, in 2013 and an Engineering degree from the University of San Francisco de Quito, Ecuador, in 2010. His

research interests include simulation-based optimization, multi-objective optimization, optimization under uncertainty and robust optimization.

**Murat Efe** received his Ph.D. degree in Electronics Engineering from University of Sussex, United Kingdom, in 1998. He is currently an associate professor at Ankara University, Turkey, Department of Electrical-Electronics Engineering. His research interests include Kalman filtering multi-target multi-sensor tracking, detection and estimation, cognitive radar, passive network sensing. He is a member of IEEE.

**Sadi Uçkun Emel** received his B.Sc. degree in Electric and Electronics Engineering from Middle East Technical University of Ankara, Turkey, in 2000. He is currently working as a senior software engineer. His research interests include agile software development methodologies and optimization algorithms.

**Stuart James Gibson** received his Ph.D. degree in Physics from the University of Kent, United Kingdom, in 2007. He is currently a lecturer at the University of Kent. His research interests include interactive evolutionary computation, machine learning, image and signal processing, and face recognition. He is a member of the IEEE.

**Julia Handl** received her Ph.D. degree in Computer Science from the University of Manchester, United Kingdom, in 2006. She is currently lecturer in the Decision and Cognitive Sciences Group at Manchester Business School. Prior to this (April 2007 – June 2011), she was an MRC Special Training Fellow in Bioinformatics in the Faculty of Life Sciences at the University of Manchester and she spent six months as a visiting researcher at the University of Washington, Seattle, USA, in 2009. Her research interests include data-mining, machine learning, optimization, multi-objective optimization, multi-criterion decision making, computational biology and computational protein structure prediction.

**Timotej Jagrič** received the Ph.D. degree in Economics from the University of Maribor, Slovenia, in 2003. He is full professor of applied economics and econometrics and associate professor of finance at Faculty of Economics and Business. He is head of the Institute of finance and banking. His current research interest include risk modeling, forecasting, and development of new econometric methods. He has published

several articles in international journals and regularly attends scientific conferences. He works as consultants for financial institutions.

**David Kaljun** received his B.Sc. degree in Mechanical Engineering from the University of Maribor, Slovenia, in 2011. He is currently a young researcher and Ph.D. student at the University of Ljubljana, Faculty of Mechanical Engineering. His research interests include local optimization, light distribution optimization, computer science, intelligent expert systems. He is also a member of the young researcher group at the Slovenian Research Agency (ARRS).

**Joseph James Mist** has recently submitted his thesis for his Ph.D. in Physics at the University of Kent, United Kingdom. His research interests include interactive evolutionary computation and image processing. He is a member of the IEEE.

**Edmondo Minisci** received his Ph.D. degree in Aerospace Engineering from the Politecnico di Torino, Italy, in 2004. He is currently a lecturer at the University of Strathclyde, Glasgow, UK, in the Department of Mechanical and Aerospace Engineering. His research interests include evolutionary computing, multidisciplinary design optimization, and uncertainty based design. He is currently involved in projects regarding global optimization of trans-atmospheric and interplanetary trajectories, uncertainty based design of space transportation systems, and design of innovative horizontal and vertical axis wind turbines.

**Uroš Mlakar** received his B.Sc. degree in Computer Science from the University of Maribor, Slovenia, in 2014. He is currently affiliated with the University of Maribor, Faculty of Electrical Engineering and Computer Science. His research interests include computer vision and evolutionary algorithms.

**Gregor Papa** received the Ph.D. degree in Electrical engineering from the University of Ljubljana, Slovenia, in 2002. He is a Researcher at the Computer Systems department, Jožef Stefan Institute, Ljubljana, since 1997. He is also an Assistant Professor at the Jožef Stefan International Postgraduate School, Ljubljana. His research interests include optimization techniques, metaheuristic algorithms, and hardware implementations of high-complexity algorithms. His work is published in

international journals and conference proceedings. He is a member of the IEEE and ACM.

**Anatoly Shalyto** received his Ph.D. degree in Automation from the Saint-Petersburg Electrotechnical University "LETI", Russian Federation, in 1999. He is currently a professor at the ITMO University, Saint-Petersburg, Faculty of Information technologies and programming. His research interests include automation, search-based software engineering, and evolutionary algorithms.

**Christopher John Solomon** received his Ph.D. degree in Physics from the Royal Marsden Hospital, University of London, United Kingdom, in 1989. He is currently a Faculty member at the University of Kent, School of Physical Sciences. His research interests include Image Processing, evolutionary computation, pattern recognition, and facial synthesis and recognition.

**Polona Tomašič** received her B.Sc. degree in Computer Science and Mathematics from the University of Ljubljana, Slovenia, in 2013. She is currently a Ph.D. student in Information and Communication Technologies at the Jožef Stefan International Postgraduate School, Ljubljana. Her research interests include evolutionary computing and computational creativity.

**Vladimir Ulyantsev** received his M.Sc. degree in Applied mathematics and informatics from the ITMO University, Saint-Petersburg, Russian Federation, in 2013. He is currently a Ph.D. student at ITMO University, Faculty of Information technologies and programming. His research interests include machine learning, search-based software engineering, and bioinformatics.

**Massimiliano Vasile** received his Ph.D. degree in Aerospace Engineering from Politecnico di Milano, Italy, in 2001. He is currently a professor at the University of Strathclyde, Glasgow, United Kingdom, Faculty of Engineering. His research interests include computational optimization, robust design and optimization under uncertainty. He is a member of IEEE, Computational Intelligence Society's Emerging Technology Committee, and IAF Space Power Committee.

**Dejan Velušček** received his Ph.D. in Mathematics from University of Ljubljana in 2005. He is currently an Assistant Professor of Financial Mathematics at the University of Ljubljana, Slovenia, Faculty of Mathematics and Physics. He is also a researcher at the Institute of Mathematics, Physics and Mechanics in Ljubljana. His research interests include probabilistic numerical methods, modeling with stochastic processes and stochastic optimization.

**Tayfur Yaylagul** received his B.Sc. degree in Computer Science & Engineering from Hacettepe University of Ankara, Turkey, in 1996. He is currently working as a chief software engineer. His research interests include missin planning and optimization algorithms.

**Asım Egemen Yilmaz** received his Ph.D. degree in Electrical-Electronics Engineering from Middle East Technical University, Ankara, Turkey, in 2007. He is currently an associate professor at Ankara University, Department of Electrical-Electronics Engineering. His research interests include computational electromagnetics, nature-inspired optimization algorithms, knowledge-based systems; more generally software development processes and methodologies. He is a member of IEEE.

**Aleš Zamuda** received his Ph.D. degree in Computer Science from the University of Maribor, Slovenia, in 2012. 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.

**Janez Žerovnik** received his Ph.D. degree in Computer Science from the University of Ljubljana, Slovenia, in 1992 and his Ph.D. degree in Mathematics from the Techical University Graz, Austria, in 1994. He is currently a professor mathematics at University of Ljubljana and a part time researcher at the Institute of mathematics, physics and mechanics, Ljubljana, Slovenia. He was a research fellow at Montanuniversitaet Leoben, Austria, École Normale Supérieure Lyon, France, and Royal Holloway, University of London, United Kingdom. He has published more than 100 papers in refereed journals and more than 100 papers in proceedings of scientific conferences. He is a member of editorial boards of Ars Mathematica Contemporanea, ISRN Discrete Mathematics, and Central European Journal of Operations Research, and served as guest

editor of special issues of Central European Journal of Operations Research (twice), Discussiones Mathematicae. Graph Theory, Theoretical Computer Science, and several conference proceedings including Structural information and communication complexity : revised selected papers, (Lecture Notes in Computer Science, 5869). His main research interest are graph theory and optimization, and more general discrete mathematics with applications in theoretical computer science, chemical graph theory and operational research.

**Martin Žnidaršič** is a post-doctoral researcher at the Department of Knowledge Technologies of the Jožef Stefan Institute, Ljubljana, Slovenia. His main research interests are in decision support and data mining with a focus on probabilistic modeling, evaluation modeling and sentiment analysis. He was active in several EU FP5, FP6 and FP7 research projects and is currently active in two FP7 projects concerned with computational creativity: ConCreTe and WHIM. Besides research, he is interested in Web programming and is currently teaching courses on Web programming at the Faculty of Information Sciences Novo mesto, Slovenia, and at the Jožef Stefan International Postgraduate School, Ljubljana.

# ANALYSIS OF TWO ALGORITHMS FOR MULTI-OBJECTIVE MIN-MAX OPTIMIZATION

Simone Alicino
*Mechanical and Aerospace Engineering*
*University of Strathclyde, Glasgow, United Kingdom*
simone.alicino@strath.ac.uk

Massimiliano Vasile
*Mechanical and Aerospace Engineering*
*University of Strathclyde, Glasgow, United Kingdom*
massimiliano.vasile@strath.ac.uk

**Abstract** This paper presents two memetic algorithms to solve multi-objective min-max problems, such as the ones that arise in evidence-based robust optimization. Indeed, the solutions that minimize the design budgets are robust under epistemic uncertainty if they maximize the belief in the realization of the value of the design budgets. Thus robust solutions are found by minimizing with respect to the design variables the global maximum with respect to the uncertain variables. A number of problems, composed of functions whose uncertain space is modelled by means of evidence theory, and presenting multiple local maxima as well as concave, convex, and disconnected fronts, are used to test the performance of the proposed algorithms.

**Keywords:** Evidence-based robust optimization, Multi-objective optimization, Worst-case scenario design.

## 1. Introduction

Worst-case scenario problems arise whenever a performance index, or cost function, has to be optimal with respect to a design vector $\mathbf{d}$, and at the same time robust against an uncertain vector $\mathbf{u}$. This class of problems is common in several fields, such as game theory, decision making, robust control, risk analysis, and robust design. For instance, the lower expectation in the realization of the value of a particular performance

**Algorithm 1** Min-max optimization via restoration.

1: Initialize archive $A_u = \{\mathbf{u}_1\}$, and set $i = 1$
2: **while** the stopping condition is not met **do**
3: Compute $\mathbf{d}_i = \arg\min_{\mathbf{d}\in D} \left\{ \max_{\mathbf{u}\in A_u} f(\mathbf{d}, \mathbf{u}) \right\}$
4: Compute $\mathbf{u}_{i+1} = \arg\max_{\mathbf{u}\in U} f(\mathbf{d}_i, \mathbf{u})$
5: Add $\mathbf{u}_{i+1}$ to the archive $A_u$
6: $i \leftarrow i + 1$
7: **end while**
8: Return $\{\mathbf{d}_i, \mathbf{u}_{i+1}\}$.

index for a model of a system can be defined as the degree of belief that one has in a certain proposition being true, given the available evidence. In the framework of imprecise probabilities, it can be seen as a lower bound to the cumulative distribution function of classical probability theory. Its use is therefore interesting in engineering design, as it gives the lower limit of the confidence that the design budgets under uncertainty will be below a given threshold. In this framework both epistemic and aleatory uncertainties can be treated even when no exact information on the probability distribution associated to an uncertain quantity is available. Stochastic variables and associated probability are replaced by a multivalued mapping from a collection of subsets of an uncertain space $U$ into a lower expectation (Belief function in the case of Evidence Theory). The main drawback of the use of multivalued mappings is that the computation of the lower expectation, i.e. the Belief, has a complexity that is exponential with the number of uncertain variables. Recently, some strategies were proposed in [10] to obtain an estimation of the maximum Belief with a reduction of the computational cost. The approach starts by translating an optimization under uncertainty into a single or multi-objective min-max problem equivalent to a worst-case scenario optimization problem. Several methods have been proposed to address single-objective min-max problems, especially using evolutionary approaches [2, 3], and metamodels [4, 5, 12]. For the multi-objective case, a gradient-based approach is presented in [1]. An interesting approach is based on the procedure proposed in [4, 8] for single-objective problems, and exploited in [6] for interval multi-objective linear programming. Such procedure is based on an iterative minimization over the design space and subsequent restoration of the global maximum over the uncertain space as shown in Algorithm 1. The stopping condition can be the achievement of a desired accuracy, or a maximum number of function evaluations, for example. In this paper we present a multi-

objective version of Algorithm 1 implemented in an algorithm called MACSminmax. MACSminmax, employs MACS2 and IDEA at steps 3 and 4, respectively. Another algorithm, MACS$\nu$, is presented in this paper and compared to MACSminmax. MACS$\nu$ is a variant of MACS2 containing heuristics to deal with min-max problems. The paper starts with a brief introduction to Evidence Theory and its use in the context of robust design optimization in Section 2. Section 3 introduces the two memetic algorithms, MACSminmax and MACS$\nu$. Section 4 finally presents the results on some test cases.

## 2. Evidence-Based Robust Design Optimization

Evidence theory [7] allows to adequately model both epistemic and aleatory uncertainty when no information on the probability distributions is available. For instance, during the preliminary design of an engineering system, experts can provide informed opinions by expressing their belief in an uncertain parameter $u$ being within a certain set of intervals. The level of confidence an expert has in $u$ belonging to one of the intervals is quantified by using a mass function generally known as Basic Probability Assignment (*bpa*). All the intervals form the so-called frame of discernment $\Theta$, which is a set of mutually exclusive elementary propositions. The power set of $\Theta$ is called $U = 2^{\Theta}$, or the set of all the subsets of $\Theta$ (the uncertain space in the following). An element $\theta$ of $U$ that has a non-zero *bpa* is called focal element. When more than one parameter is uncertain, the focal elements are the result of the Cartesian product of all the elements of each power set associated to each uncertain parameter. The *bpa* of a given focal element is then the product of the *bpa* of all the elements in the power set associated to each parameter. All the pieces of evidence completely in support of a given proposition form the cumulative belief function $Bel$, defined as follows:

$$Bel(A) = \sum_{\forall \theta_i \subseteq A} m(\theta_i) \tag{1}$$

where $A$ is the proposition about which the Belief is evaluated. For example, the proposition can be expressed as:

$$A = \{\mathbf{u} \in U \mid f(\mathbf{u}) \leq \nu\} \tag{2}$$

where $f$ is the outcome of the system model and the threshold $\nu$ is the desired value of a design budget. It is important to note that the set $A$ can be disconnected or present holes, likewise the focal elements can be disconnected or partially overlapping. This introduces discontinuities in the search space, making the problem more difficult to solve.

An engineering system to be optimized can be modelled as a function $f : D \times U \subseteq \Re^{m+n} \rightarrow \Re$. The function $f$ represents the model of the system budgets (e.g. power budget, mass budget, etc.), and depends on some uncertain parameters $\mathbf{u} \in U$ and design parameters $\mathbf{d} \in D$, where $D$ is the available design space and $U$ the uncertain space. What is interesting for the designers is the value of the function $f$ for which $Bel = 1$, i.e. it is maximum. This value of the design budget is the threshold $\boldsymbol{\nu}_{\max}$ above which the design is certainly feasible, given the current body of evidence. If $q$ objective functions exist, then the following problem can be solved without considering all the focal elements:

$$\boldsymbol{\nu}_{\max} = \min_{\mathbf{d} \in D} \mathbf{F} = \min_{\mathbf{d} \in D} [\max_{\mathbf{u} \in \bar{U}} f_1(\mathbf{d}, \mathbf{u}), \ldots, \max_{\mathbf{u} \in \bar{U}} f_q(\mathbf{d}, \mathbf{u})]^T \tag{3}$$

Problem in 3 is a multi-objective min-max over the design space $D$ and the uncertain space $\bar{U}$, where $\bar{U}$ is a unit hypercube collecting all the focal elements in a compact set with no overlapping or holes. The transformation between $U$ and $\bar{U}$ is given by:

$$\mathbf{x}_U = \frac{\left(\mathbf{b}^u_{U,i} - \mathbf{b}^l_{U,i}\right)}{\left(\mathbf{b}^u_{\bar{U},i} - \mathbf{b}^l_{\bar{U},i}\right)} \mathbf{x}_{\bar{U},i} + \mathbf{b}^l_{U,i} - \frac{\left(\mathbf{b}^u_{U,i} - \mathbf{b}^l_{U,i}\right)}{\left(\mathbf{b}^u_{\bar{U},i} - \mathbf{b}^l_{\bar{U},i}\right)} \mathbf{b}^l_{\bar{U},i} \tag{4}$$

where $\mathbf{b}^u_{U,i}$ and $\mathbf{b}^l_{U,i}$ (resp. $\mathbf{b}^u_{\bar{U},i}$ and $\mathbf{b}^l_{\bar{U},i}$) are the upper and lower boundaries of the $i-th$ hypercube to which $\mathbf{x}_{U,i}$ (resp. $\mathbf{x}_{\bar{U},i}$) belongs.

## 3. Multi-Objective Min-Max Memetic Optimization

Problem (3) searches for the minimum of the maxima of all the functions over $\bar{U}$ and represents an example of worst-case scenario design optimization. The maximum of every function is independent of the other functions and corresponds to a different uncertain vector. Therefore, all the maxima can be computed in parallel with $q$ single-objective maximizations. The maximization of each function is performed by running a global optimization over $\bar{U}$ using Inflationary Differential Evolution (IDEA). The minimization over $D$ is performed by means of MACS2. IDEA [9] is a population-based memetic algorithm for single-objective optimization. It hybridizes Differential Evolution and Monotonic Basin Hopping in order to simultaneously improve local convergence and avoid stagnation. MACS2 [13] is a memetic algorithm for multi-objective optimization based on a combination of Pareto ranking and Tchebycheff scalarization. The search for non-dominated solutions is performed by a population of agents which combine individualistic and social actions.

The initial population is randomly generated in the search domain. Individualistic actions perform a sampling of the search space in a neighborhood of each agent. Then, subsets of the population perform social actions aiming at following particular descent directions in the criteria space. Social agents implement a Differential Evolution operator and assess the new candidate solutions using Tchebycheff scalarization. Current non-dominated solutions are then stored in an archive. Both social and individualistic actions make use of a combination of the population and the archive.

In a classical minimization problem two solutions $\mathbf{d}_1$ and $\mathbf{d}_2$ are ranked according to which one gives the lower value of the function. In the minimization loop of a min-max problem, the same can be done only if the maximization loop has returned the actual global maxima $\tilde{\mathbf{u}}_1$ and $\tilde{\mathbf{u}}_2$. However, this is usually not true. Therefore a mechanism of cross-check such that also $(\mathbf{d}_1, \mathbf{u}_2)$ and $(\mathbf{d}_2, \mathbf{u}_1)$ are evaluated is needed in order to increase the probability that each maximization identifies the global maximum, and correctly rank two solutions.

### 3.1 MACS$\nu$

MACS$\nu$ (Algorithm 2) is the min-max variant of MACS2. It endows MACS2 with special heuristics to increase the probability of finding the global maxima in $\bar{U}$. More in detail, a Cross-check (lines 7, 18, and 28) compares the values of the objective functions for a newly generated design vector in the trial populations $P_t$ (line 7) and $P_s$ (line 18) against the function values of a solution already archived in $A$ (indicated with subscript *arch* in Algorithm 2). In addition, the cross-check performs a local search or a simple function evaluation in the inner maximization loop depending on whether the location of the maxima changes or not, respectively, for different design vectors. After the cross-check, a Min-Max Selection (lines 11 and 22) compares the population $P$ with the new candidate populations $P_t$ (line 11) and $P_s$ (line 22) and selects the design vectors to attribute to the next generation according to the following rule: If $\mathbf{d}$ (resp. $\mathbf{u}$) is unchanged, the old $\mathbf{u}$ (resp. $\mathbf{d}$) is replaced with the new one, if it yields a higher (resp. lower) value of the objective function; if both $\mathbf{d}$ and $\mathbf{u}$ are different, the new vectors will replace the old ones. At the end of the algorithm, and at the last iteration, a Validation (line 24) mitigates the possibility that the cross-check operators assign the same incorrect $\mathbf{u}$ to all $\mathbf{d}$ vectors in the population and archive. This is done by starting from the minimum value of the first objective in the archived Pareto front, and performing a global search in the uncertain space. If the new uncertain vector gives a higher value of

**Algorithm 2** MACS$\nu$

Initialize population $P$, archive $A = P$, $n_{feval} = 0$, $\epsilon = 0.7$, $\delta = 10^{-6}$
**while** $n_{feval} < n_{feval,max}$ **do**
  Run individualistic moves and generate trial population $P_t$
  **for all** $\mathbf{d} \in P_t$ **do**
    **for all** $\mathbf{d}_{arch} \in A$ **do**
      **if** $\mathbf{d} \succ \mathbf{d}_{arch}$ **then**
        Cross-check$(P_t, A)$
      **end if**
    **end for**
  **end for**
  Min-Max Selection$(P, P_t)$
  Update $P$ and $A$
  $Z \leftarrow \|\mathbf{F}_{arch}^{max} - \mathbf{F}_{arch}^{min}\|$
  Run social moves and generate candidate population $P_s$
  **for all** $\mathbf{d} \in P_s$ **do**
    **for all** $\mathbf{d}_{arch} \in A$ **do**
      **if** $\mathbf{d} \succ \mathbf{d}_{arch}$ **or** $\|\mathbf{F}(\mathbf{d}) - \mathbf{F}(\mathbf{d}_{arch})\| > \epsilon Z$ **then**
        Cross-check$(P_s, A)$
      **end if**
    **end for**
  **end for**
  Min-Max Selection$(P, P_s)$
  Update $P$ and $A$
  Validation$(A)$
  **for all** $\mathbf{d} \in P$ **do**
    **for all** $\mathbf{d}_{arch} \in A$ **do**
      **if** $\mathbf{d} \succ \mathbf{d}_{arch}$ or $\mathbf{d} \prec \mathbf{d}_{arch}$ **then**
        Cross-check$(P, A)$
      **else if** $\|\mathbf{F}(\mathbf{d}) - \mathbf{F}(\mathbf{d}_{arch})\| < \delta$ **then**
        Replace $\mathbf{u} \in P$ with $\mathbf{u} \in A$
      **end if**
    **end for**
  **end for**
**end while**

the function, then it replaces the old one. This operation is repeated for the elements in the archived Pareto front until there is no more variation in their value.

**Algorithm 3** MACSminmax

1: Initialize archive $A_u = \{\mathbf{u}_1\}$, $n_{feval} = 0$
2: **while** $n_{feval} < n_{feval,max}$ **do**
3: Run MACS2 to compute $\mathbf{d}_{min} = \arg\min_{\mathbf{d}\in D} \max_{\mathbf{u}\in A_u} f(\mathbf{d}, \mathbf{u})$ and associated $\mathbf{f_d}$
4: Add $\mathbf{d}_{min}$ to the archive $A_d$
5: **for all** $\mathbf{d}_{min} \in A_d$ **do**
6: **for all** $l \in \{1, \ldots, q\}$ **do**
7: Run IDEA to compute $\mathbf{u}^l_{max} = \arg\max_{\mathbf{u}\in\bar{U}} f^l(\mathbf{d}_{min}, \mathbf{u})$ and associated $f^l_{\mathbf{u}}$
8: **if** $f^l_{\mathbf{u}} > f^l_{\mathbf{d}}$ **then**
9: Add $\mathbf{u}^l_{max}$ to the archive $A_u$
10: **else**
11: Evaluate function to find $\mathbf{u}^l_{max} = \arg\max_{\mathbf{u}\in A_u} f^l(\mathbf{d}_{min}, \mathbf{u})$
12: **end if**
13: **end for**
14: **end for**
15: **end while**
16: **for all** $\mathbf{d}_{min} \in A_d$ **do**
17: **for all** $l \in \{1, \ldots, q\}$ **do**
18: Run local search to refine $\mathbf{u}^l_{max} \in A_u$ associated to $\mathbf{d}_{min}$
19: **end for**
20: **end for**
21: Return non-dominated $\mathbf{d}_{min}$ and associated $\mathbf{u}^l_{max}$

### 3.2 MACSminmax

MACSminmax (Algorithm 3) is a min-max memetic algorithm inspired by the procedure of Algorithm 1. This is the main difference with MACS$\nu$. In MACSminmax, for each agent of the minimization the best function value is computed with respect to an archive $A_u$ of candidate uncertain vectors (line 3). The archive $A_u$ is composed of the results of a global maximization or a simple function evaluation, depending on which one of the two gives the higher function value (as explained above, it is not guaranteed that the maximization finds the global maximum), for each design vector contained in another archive $A_d$ of candidate solutions (lines 5 to 14). Thus, each element in the archive $A_u$ corresponds to an element in the archive $A_d$. This is so if the global maxima change for different design vectors. If they do not change, the archive $A_u$ is composed of only one element. At the end of the main loop, a local

search is run for each element of the archive $A_d$ in order to refine the accuracy of the elements in the archive $A_u$. Finally, because the archive $A_d$ is filled with batches of solutions given in output by MACS2, the solutions are non-dominated only inside each batch. Therefore a further dominance check is necessary to find the non-dominated solutions among the batches.

Interesting is a comparison between MACSminmax and MACS$\nu$. In MACS$\nu$ a maximization is run for every agent of the minimization, whereas in MACSminmax each agent of the minimization is cross-checked with the archive of candidate uncertain vectors through a function evaluation. However, it is worth noting that the evaluation, in MACSminmax, of each $\mathbf{d}$ against an archive $A_u$ of candidate uncertain vectors, as well as the update of $A_u$ for each element of an archive $A_d$ of candidate design vectors, is equivalent to the cross-checks implemented in MACS$\nu$. Furthermore, the local search in MACSminmax after the main loop is similar to the validation procedure in MACS$\nu$, where a global search is run starting from the extrema of the Pareto front. Finally, in terms of balance between exploration (social moves) and exploitation (individualistic moves) of the search space, both MACS$\nu$ and MACSminmax employ the same search algorithms, MACS2 and IDEA, therefore they are equivalent so long as the parameters (population, $F$, $C_R$) are set to the same values.

## 4. Test Cases

MACS$\nu$ and MACSminmax were tested on the six bi-objective and one tri-objective test cases reported in Table 1, where $n$ is the dimension of the design vector $\mathbf{d}$, as well as the uncertain vector $\mathbf{u}$ – therefore the total dimension of the test cases is $2n$ – and $n_{feval,max}$ is the maximum number of function evaluations, i.e. the termination condition for the algorithms. The test cases are composed of the functions in Table 2. The functions are easily scalable and present very challenging landscapes, with multiple maxima that can change significantly with the design vector. Function MV10, in particular, is characterized by having the maxima located on top of multiple sharp, steep peaks. Note also that the test cases present several types of Pareto fronts, convex, concave, linear, and disconnected. The uncertain vector $\mathbf{u}$ is assigned the *bpa* structure reported in Table 3. The uncertain intervals present holes and overlappings, that introduce discontinuities in the uncertain space. The reference solution, i.e. the real front in Figures 1, was computed by merging the results of 200 runs of the same problems solved by means of MACS$\nu$ with the results of 200 runs of MACSminmax.

*Table 1.* Test cases.

| Test Case | Functions | $\mathbf{d}$ | $n$ | $n_{feval,max}$ |
|---|---|---|---|---|
| TC1 | $f_1 = \text{MV1},\ f_2 = \text{MV3}$ | $[1,5]^n$ | 2 | 2E5 |
| TC2 | $f_1 = \text{MV2},\ f_2 = \text{MV8}$ | $[0,3]^n$ | 8 | 1E6 |
| TC3 | $f_1 = \text{MV2},\ f_2 = \text{EM1}$ | $[1,5]^n$ | 8 | 1E6 |
| TC4 | $f_1 = \text{MV8},\ f_2 = \text{MV9}$ | $[1,3]^n$ | 2 | 4E5 |
| TC5 | $f_1 = \text{MV8},\ f_2 = \text{EM1}$ | $[1,5]^n$ | 4 | 1E6 |
| TC6 | $f_1 = \text{MV10},\ f_2 = \text{MV9}$ | $[-4,2\pi]^n$ | 1 | 1E5 |
| TC7 | $f_1 = \text{MV2},\ f_2 = \text{MV8},\ f_3 = \text{EM1}$ | $[1,5]^n$ | 4 | 1E6 |

*Table 2.* Test functions.

| ID | Function |
|---|---|
| MV1 | $f = \sum_{i=1}^{n} d_i u_i^2$ |
| MV2 | $f = \sum_{i=1}^{n} (d_i - u_i)^2$ |
| MV3 | $f = \sum_{i=1}^{n} (5 - d_i)(1 + \cos u_1) + (d_i - 1)(1 + \sin u_i)$ |
| MV8 | $f = \sum_{i=1}^{n} (2\pi - u_i) \cos(u_i - d_i)$ |
| MV9 | $f = \sum_{i=1}^{n} (d_i - u_i) \cos(-5u_i + 3d_i)$ |
| MV10 | $f = \sum_{i=1}^{n} (d_i + u_i) \cos(-u_i(5\lvert d\rvert + 5) + 3d_i)$ |
| EM1 | $f = \sum_{i=1}^{n} (u_i - 3d_i) \sin u_i + (d_i - 2)^2$ |

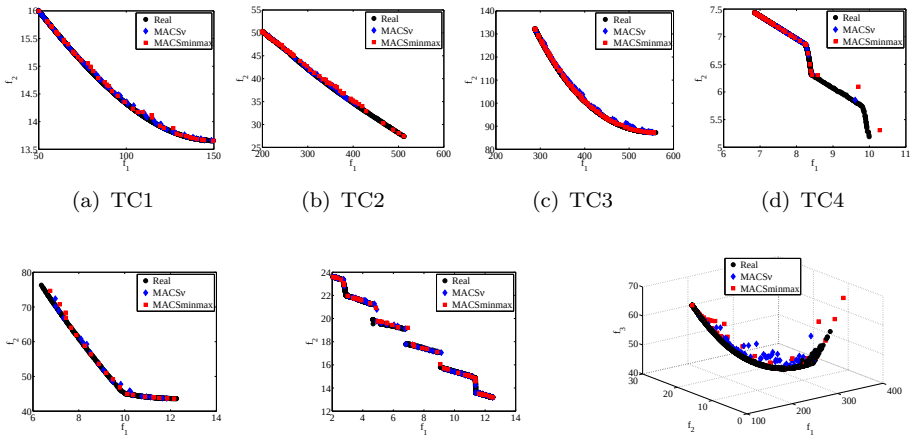
From a sensitivity analysis on total number of agents (5, 10, 20) vs. subset of social agents (1/3, 1/2, 1), and $F$ (0.1, 0.5, 1, 2) vs. $C_R$ (0.1, 0.5, 0.9) for MACS2 and IDEA resulted that the best settings were: $200n$ function evaluations for both MACS2 and IDEA, for 10 agents for MACS2, half of which perform the social actions, 5 agents for IDEA, and $F = 1$ and $C_R = 0.1$ for both MACS2 and IDEA. The sensitivity analyses were run for test case TC4 with a total of 2E5 function evaluations, and the results assessed in terms of success rate of finding the global maximum, as well as convergence $M_{conv}$ and spreading $M_{spr}$ as per definition in [11]. The same settings were used in all the test cases.

Table 4 summarizes the success rates of finding the global maxima, as well as convergence and spreading of MACSminmax in comparison to MACS$\nu$. The results are the average performances obtained from the 200 runs needed to achieve a confidence interval of 95% on the success rate being within a $\pm 5\%$ interval containing its estimated value [9]. Columns $max f_1$, $max f_2$ and $max f_3$ contain the maximization success

*Table 3.* *bpa* structure of the uncertain variables.

| | | | | |
|---|---|---|---|---|
| MV1, MV2, MV3 | Interval | [-5 -4] | [-3 0] | [-1 3] |
| | *bpa* | 0.1 | 0.25 | 0.65 |
| MV8 | Interval | [0 1] | [2 4] | [3 2$\pi$] |
| | *bpa* | 0.1 | 0.25 | 0.65 |
| MV9 | Interval | [-$\pi$/2 -$\pi$/6] | [0 $\pi$] | [3$\pi$/4 3$\pi$/2] |
| | *bpa* | 0.1 | 0.4 | 0.5 |
| MV10 | Interval | [$\pi$ 4] | [5 6] | [5.5 2$\pi$] |
| | *bpa* | 0.1 | 0.25 | 0.65 |
| EM1 | Interval | [0 5] | [7 14] | [12 20] |
| | *bpa* | 0.1 | 0.5 | 0.4 |

rates computed with an accuracy of $10^{-4}$ with respect to the actual maxima, columns $M_{conv}$ and $M_{spr}$ contain the mean value of $M_{conv}$ and $M_{spr}$ respectively, and columns $p_{conv}/t_{conv}$ and $p_{spr}/t_{spr}$ contain the success rate of computing a front which convergence and spreading are below the thresholds $t_{conv}$ and $t_{spr}$ also contained in the columns after the '/' symbol. MACSminmax attains performances similar to MACS$\nu$, with excellent success rates for almost all the test cases. For TC2, TC3 and TC7, MACSminmax provides a significantly better spreading (2.0, 0.3, and 2.1) than MACS$\nu$ (16.1, 7.5, and 9.3). Note also that TC2 and TC3 are the test cases with the higher dimension, 16, whereas TC7 has the highest number of objectives, 3. Moreover, for TC5 MACSminmax has a significantly higher success rate for the maximization of the second objective (87.6% against 54.1%): in function EM1 the global maximum has a jump for a certain value of $\mathbf{d}$. This makes the global maximum been tracked more effectively with the global search implemented in MACSminmax than with the local search of MACS$\nu$. However, for TC5 average converge and spreading computed by MACSminmax, and their success rates, are worse than for MACS$\nu$. MACS$\nu$ also performs better at finding a front for TC6 which spreading is below a threshold equal to 2. In conclusion, MACSminmax has equal or better capability in the maximization in the uncertain space, and also in terms of convergence and spreading, than MACS$\nu$, which in turns performed slightly better in two cases. On one hand, such rather equivalent performances of the two algorithms can be explained by the fact that they have equivalent balance between exploration and exploitation, as explained in subsec-




(a) TC1 (b) TC2 (c) TC3 (d) TC4

(e) TC5 (f) TC6 (g) TC7

*Figure 1.* Pareto fronts of the test cases.

tion 3.2. On the other hand, the better performance of MACSminmax on some of the test cases can be due to more effective archiving, cross-check and validation mechanisms, which are the only aspects that differentiate MACS$\nu$ and MACSminmax. The Pareto fronts for the seven test cases are shown in Figure 1. As a comparison, the fronts computed by means of MACS$\nu$ and of MACSminmax are displayed. One can see that MACSminmax performs as well as MACS$\nu$ at identifying the true Pareto front for all the test cases. In addition, it is worth noting that TC4 presents a deceptive front, as the bottom-right portion of it has a multitude of dominated fronts above it. This resulted to be a very difficult part for both MACS$\nu$ and MACSminmax to identify.

## 5. Conclusions

Two multi-objective min-max memetic algorithms, MACSminmax and MACS$\nu$ have been presented and compared in this paper. MACS$\nu$ is a variant of MACS2 endowed with cross-checks, and selection and validation mechanisms to properly maximize the subproblem. MACSminmax makes use of an iterative restoration of the global maxima in the uncertain space. Despite the different procedures, the two strategies implement similar cross-checks. The two algorithms have been tested on seven scalable test cases that present several types of Pareto fronts. Results show that both MACSminmax and MACS$\nu$ are able to achieve similar very good performances, in terms of finding the global maxima in the uncertain space and the true Pareto front. However, MACSminmax per-

*Table 4.* Results: comparison between MACS$\nu$ and MACSminmax.

| Test Case | Algorithm | $maxf_1$ | $maxf_2$ | $maxf_3$ | $M_{conv}$ | $M_{spr}$ | $p_{conv}$ / $t_{conv}$ | $p_{spr}$ / $t_{spr}$ |
|---|---|---|---|---|---|---|---|---|
| TC1 | MACS$\nu$ | 100% | 100% | - | 0.2 | 1.7 | 100 / 0.5 | 79 / 2 |
| | MACSminmax | 100% | 100% | - | 0.2 | 1.3 | 100 / 0.5 | 100 / 2 |
| TC2 | MACS$\nu$ | 100% | 65% | - | 0.5 | 16.1 | 100 / 1 | 0 / 2 |
| | MACSminmax | 100% | 60% | - | 0.6 | 2.0 | 100 / 1 | 64 / 2 |
| TC3 | MACS$\nu$ | 100% | 100% | - | 0.6 | 7.5 | 46 / 0.5 | 3 / 2 |
| | MACSminmax | 100% | 100% | - | 0.1 | 0.3 | 100 / 0.5 | 100 / 2 |
| TC4 | MACS$\nu$ | 100% | 91.3% | - | 0.3 | 0.9 | 83 / 0.5 | 97 / 2 |
| | MACSminmax | 100% | 85.7% | - | 0.4 | 1.0 | 77 / 0.5 | 91 / 2 |
| TC5 | MACS$\nu$ | 98.6% | 54.1% | - | 1.2 | 5.8 | 48 / 1 | 60 / 6 |
| | MACSminmax | 92.8% | 87.6% | - | 2.7 | 8.0 | 24 / 1 | 42 / 6 |
| TC6 | MACS$\nu$ | 100% | 100% | - | 0.3 | 1.2 | 95 / 0.5 | 97 / 2 |
| | MACSminmax | 100% | 100% | - | 0.3 | 2.0 | 91 / 0.5 | 63 / 2 |
| TC7 | MACS$\nu$ | 100% | 100% | 95.3% | 5.0 | 9.3 | 50 / 5 | 8 / 5 |
| | MACSminmax | 100% | 100% | 98.3% | 4.6 | 2.1 | 66 / 5 | 100 / 5 |

formed significanly better in terms of spreading in the two test cases with the highest dimension and the one with three objectives. Multi-objective min-max optimization algorithms find applicability to worst-case scenario problems, such as evidence-based robust engineering design.

## References

[1] S. Azarm and H. Eschenauer. A Minimax Reduction Method for Multi-Objective Decomposition-Based Design Optimization. *Struct. Optimization*, 6:94–98, 1993.

[2] A. M. Cramer, S. D. Sudhoff, and E. L. Zivi. Evolutionary Algorithms for Minimax Problems in Robust Design. *IEEE T. Evolut. Comput.*, 13(2):444–453, 2009.

[3] R. I. Lung and D. Dumitrescu. A New Evolutionary Approach to Minimax Problems. In *Proc. IEEE Congress on Evolutionary Computation (CEC)*, 2011.

[4] J. Marzat, E. Walker, and H. Piet-Lahanier. Worst-case Global Optimization of Black-Box Functions through Kriging and Relaxation. *J. Global Optim.*, 55:707–727, 2013.

[5] Y.-S. Ong, P. B. Nair, and K. Y. Lum. Max-Min Surrogate-Assisted Evolutionary Algorithm for Robust Design. *IEEE T. Evolut. Comput.*, 10:392–404, 2006.

[6] S. Rivaz and M. A. Yaghoobi. Minimax Regret Solution to Multiobjective Linear Programming Problems with Interval Objective Functions Coefficients. *Cent. Europ. J. Oper. Re.*, 21:625–649, 2013

[7] G. Shafer *A Mathematical Theory of Evidence*. Princeton University Press, 1976.

[8] K. Shimizu and E. Aiyoshi. Necessary Conditions for Min-Max Problems and Algorithms by a Relaxation Procedure. *IEEE T. Automat. Contr.*, 25(1):62–66, 1980.

[9] M. Vasile, E. Minisci, and M. Locatelli. An Inflationary Differential Evolution Algorithm for Space Trajectory Optimization. *IEEE T. Evolut. Comput.*, 15:267–281, 2011.

[10] M. Vasile, E. Minisci, and Q. Wijnands. Approximated Computation of Belief Functions for Robust Design Optimization. In *Proc. 53rd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, 2012.

[11] M. Vasile and F. Zuiani. Multi-agent Collaborative Search: An Agent-based Memetic Multi-Objective Optimization Algorithm Applied to Space Trajectory Design. *J. Aerosp. Eng.* 225:1211–1227, 2011.

[12] A. Zhou and Q. Zhang. A Surrogate-Assisted Evolutionary Algorithm for Minimax Optimization. In *Proc IEEE Congress on Evolutionary Computation (CEC)*, Barcelona, Spain, 2010.

[13] F. Zuiani and M. Vasile. Multi Agent Collaborative Search Based on Tchebycheff Decomposition. *Comput. Optim. Appl.*, 56(1):189–208, 2013.

# COMPARISON BETWEEN SINGLE AND MULTI OBJECTIVE GENETIC ALGORITHM APPROACH FOR OPTIMAL STOCK PORTFOLIO SELECTION

Nejc Cvörnjek
*Faculty of Mechanical Engineering, University of Maribor, Slovenia*
*and*
*Faculty of Economics and Business, University of Maribor, Slovenia*
nejc.cvornjek@gmail.com

Miran Brezočnik
*Laboratory of Intelligent Systems, Faculty of Mechanical Engineering*
*University of Maribor, Slovenia*
miran.brezocnik@um.si

Timotej Jagrič
*Institute for Finance and Banking, Faculty of Economics and Business*
*University of Maribor, Slovenia*
timotej.jagric@uni-mb.si

Gregor Papa
*Computer Systems Department*
*Jožef Stefan Institute, Ljubljana, Slovenia*
*and*
*Jožef Stefan International Postgraduate School, Ljubljana, Slovenia*
gregor.papa@ijs.si

**Abstract** Portfolio selection is one of the most common problem in the field of finance. Many investors would like to allocate their funds in such way that ratio between return and risk will be as high as possible. Up to today,

the problem has been solved with various approaches based on genetic algorithm technique and GA has proved to be suitable. In this paper we applied two different approaches based on genetic algorithm technique in order to solve the problem. First is single objective approach and second is multi objective one (NSGA-II). Results are showing that there is no significant difference between approaches.

**Keywords:** Computational finance, Genetic algorithm, NSGA-II, Portfolio optimization, Portfolio selection.

## 1. Introduction

Few decades ago finance was just one discipline inside of economy. In synergy with other science disciplines like engineering, mathematics, statistics, risk management, and computer science, finance is expanding rapidly. Today finance is independent, heavily interdisciplinary field in science with many sub-disciplines, such as portfolio management and computational finance.

Portfolio is a collection of assets desired to achieve diversification. There are different types of assets on the market. Most known assets are stocks, bonds, derivatives, commodities, etc. Portfolio can include assets of only one type as well as assets of different types. Stock portfolio is portfolio that contains only stocks. There can be any number of stocks in portfolio. By adding stocks in portfolio idiosyncratic risk can be reduced. With portfolios containing 40 or more stocks from different industries almost half of a whole risk can be eliminated. This is called diversification. Due to market risk, entire risk can never be eliminated [1].

Managing any portfolio can be a difficult task. Main goal of portfolio management is choosing best asset on the market and allocating investors capital among these assets in such proportions that there will be a maximum return along with a minimum risk. The fact which makes problem difficult is that return and risk are conflicting. Assets with high return would often have a high risk. Risk can be measured with different metrics such as variance, semi-variance, VaR, cVaR, etc.

Portfolio selection problem (PSP) is a quadratic programming (QP) problem. However, heuristic techniques could be used in optimal PSP. Among heuristic techniques genetic algorithm (GA) are very common. Shoaf and Foster demonstrated effectiveness of GA where they proved that GA has smaller time complexity than QP [10].

Until today, problem was solved with single and multi objective GA approaches. In this paper, we applied both, single as well as multi objective approach, in order to find optimal stock portfolio and compare results to see if there is any significant difference.

Paper is organized as follows. In Section 2 a problem of stock portfolio optimization is presented. In Section 3 both used techniques are described in detail. In Section 4 we give a brief description about related work. In Sections 5 a practical problem and methodology of work are presented. We show results and discuss about them in Section 6. Last section is conclusion.

## 2. Problem Presentation

State of the art of today's modern portfolio theory is the mean-variance model introduced by H. Markowitz [7] in 1952. Markowitz developed his mean-variance model (M-V model) where is assumed that there is a trade-off between return and risk. M-V model includes two parameters. First is mean which stands for expected return of portfolio. Expected return is mathematically described as

$$E\left(r_p\right) = \sum_{i=1}^{n} E\left(r_i\right) w_i \tag{1}$$

where $E\left(r_p\right)$ is an expected portfolio return, $E\left(r_i\right)$ is an expected return of $i$-th stock in portfolio and $w_i$ is a proportion of $i$-th stock in portfolio.

A second parameter is variance which stands for risk. Portfolio variance can be computed by using the equation below

$$\sigma_p^2 = \sum_{i=1}^{n} \sum_{j=1}^{n} w_i w_j \sigma_{ij} \tag{2}$$

where $\sigma_p^2$ is portfolio variance, $w_i$ and $w_j$ are weights of $i$-th and $j$-th stock, and $\sigma_{ij}$ is covariance between $i$-th and $j$-th stock.

There are weight constraints in portfolio optimization problem. The basic model has two constraints

$$\sum_{i=1}^{n} w_i = 1 \tag{3}$$

and

$$0 \leq w_i \leq 1 \tag{4}$$

where $i, j = 1, \ldots, N$.

We must warn that the last constraint applies only when long positions are allowed. If short selling is allowed portfolio weights can be negative.

To improve basic model other constraints could be included. Typical constraints are constraints on cardinality, floor-ceiling, transaction costs, etc. More on constraints in PSP can be found in [5].

## 3. Genetic Algorithm

Genetic algorithm [4] (GA) is stochastic nonlinear optimization and search technique developed by J. Holland. GA is based on principles of nature. Those principles are natural selection, reproduction and mutation.

Each organism has his own fitness, which represents organism's ability to survive. The higher is the organism's fitness, the higher is the probability that organism would be selected for reproduction and for retaining organism into the next generation.

### 3.1 Simple GA

Multi objective problem can be easily converted into single objective problem. Most often used methods are weighted sum method, $\epsilon$-constrained method, etc. [2].

Simple GA starts with randomly generated population P of size $N$. Population is a set of organisms. Each organism represent a possible solution of the problem. Then, we assign fitness to each organism, and expose them to evolutionary operations. First operation is natural selection. With natural selection best organisms in present generation are carried into the next generation without making any changes. Next operation is reproduction. Reproduction consists of two operations: Selection of parents and crossover. Operation stars with selection. Most often used is a tournament selection. There are $k$ participants in the tournament, and organism with best fitness is a winner, which becomes a parent. When two parents are selected, crossover can happen with some probability $P_c$. If crossover is happened offspring is produced. The process of reproduction is repeated until new population of offsprings with size $P$ is created. Next operation is mutation. Mutation is a random change in genetic material of organism and it occurs with some probability $P_m$.

Now new generation of organism is made. Procedure is repeated until number of generation or stopping criteria is reached.

### 3.2 NSGA-II

With multi objective approach instead of a single solution we get a whole set of solutions. This set is called a Pareto front. Every solution in set is not worse than other solutions. In multi objective approach we implemented NSGA-II algorithm, but there exist many multi objective approaches based on GA. NSGA-II was developed by Deb et. al [3] and

has been proved as an efficient algorithm for multi objective optimization, with better time efficiency than other similar approaches.

In NSGA-II, first population of parents $P_0$ of size $N$ is randomly generated. This population then produces a population of offsprings $O_0$ also of size $N$. Both populations are combined into one population $R_0$. Then population $R_0$ is transferred to non-dominate sorting procedure.

Non-dominated sorting is a procedure in which a rank or level is assigned to each organism. Organisms that are not dominated by any other organism have the best rank. Thus, organisms are removed from population, and procedure is repeated until all organisms in population have their ranks. Organism one is dominated by organism two if two conditions are satisfied. First, if organism two is strictly better in at least one criteria, and second, if organism two is not worse than organism one in any criteria.

Furthermore, new population of parents $P_1$ is made according to organism rank. Population gets filled with organisms with the same rank. When, by adding new front in population, its size exceeds, organisms in that front are selected with crowding distance. Crowding distance is a distance between neighboring organisms. For boundary organisms is assigned as $c = \infty$, but for other organisms is computed. Organisms with bigger crowding distance are added into population $P_1$ until size $N$ for population $P_1$ is reached.

Now new generation of offsprings can be made. Procedure is repeated until the number of generation or stopping criteria is reached.

## 4. Related Work

Until today, fairly large amount of research has been done. Most of research is based on M-V model.

Problem can be solved by converting it into single objective approach. There are two popular approaches. First, used in [11], parameter $\lambda$ is included and it stands for risk factor. Evaluation function is

$$maximise \quad (1-\lambda)\sum_{i=1}^{n} E\left(r_i\right) w_i - \lambda \sum_{i=1}^{n}\sum_{j=1}^{n} w_i w_j \sigma_{ij} \tag{5}$$

$$0 \leq \lambda \leq 1 \tag{6}$$

With iterating through $\lambda$ efficient frontier of portfolios could be produced. If $\lambda = 1$ risk is disinterested and portfolio with maximum return could be found. If $\lambda = 0$ global minimum portfolio could be found because return is disinterested. The second approach used in [6] is a Sharpe ratio with risk free rate ignored. Function is maximization of a

ratio between return and risk

$$maximise \ \frac{\sum_{i=1}^{n} E(r_i) w_i}{\sum_{i=1}^{n} \sum_{j=1}^{n} w_i w_j \sigma_{ij}} \tag{7}$$

and this approach we are going to use.

In MOGA approaches Mishra et al. [8] compare four elitist approaches PAES, APAES, PESA and NSGA-II. In research they evaluated their performance through three metrics of Pareto front performance, S metric, $\delta$ metric, C metric. Results were demonstrating that NSGA-II is superior against other approaches. In [9] Mishra et al. compare PESA SPEA-II and NSGA-II. Their results were showing that NSGA-II significantly outperform other two approaches. Based on that we will compare simple GA and NSGA-II.

## 5. Problem Definition and Methodology

According to our case, we attempt to find optimal stock portfolio with two different approaches, depending on the size of portfolio, and then compare their results in order to see if there is any significant difference. To implement M-V model we need historical data on prices of each stock in portfolio. Data we used was observed within the period from 01.01.2013 to 01.01.2014. In this case, all stocks are a part of S&P 500 stock market index. Data were obtained from [12]. Abbreviation of stocks used on market are in Table 1. For algorithms implementation we used Python 2.7.5 and Python(x, y) 2.7.5.0 environment.

*Table 1.* Abbreviation of stocks.

| *Portfolio size* | *Stocks included in portfolio* |
|---|---|
| 5 | CAD, TIF, AXP, NOC, FRX |
| 10 | CAD, TIF, AXP, NOC, FRX, AA, CVX, KO, F, GOOG |
| 20 | CAD, TIF, AXP, NOC, FRX, AA, CVX, KO, F, GOOG, GS, JEC, KSU, MCS, NVDA, PFE, TAP, PM, GPS, MHK |

Sizes of portfolios were 5, 10 and 20 stocks. According to this, cardinality constraint was ignored. In both approaches we used the same parameters and they are shown in Table 2. Parameters were selected based on multiple runs. With these values performance was the best.

Generation sizes were 100, 250, 500 and 1000 generations. Each organism was encoded as vector of weights. This is showed in Figure 1.

*Table 2.* Parameters used in research.

| *Parameter* | *Simple GA* | *NSGA-II* |
|---|---|---|
| Population size | 50 | 50 |
| Natural selection | 0.05 | / |
| Tournament size | 2 | 2 |
| Crossover rate | 0.9 | 0.9 |
| Mutation size | 0.2 | 0.2 |

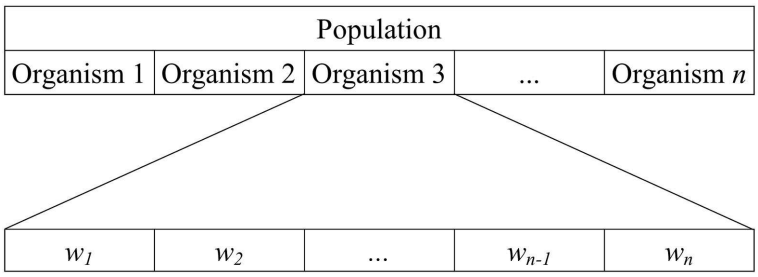



*Figure 1.* Organism encoding.

In reproduction process the tournament selection was used. Fitness function in single objective approach was a formula for Sharpe ratio, but risk free interest rate was ignored. Fitness function is defined as

$$maximize f_{(x)} = \frac{E(r_p)}{\sigma_p^2} \tag{8}$$

In multi objective approach we optimize

$$f_{(x)} = \begin{cases} maximize\ f_1(x) = \sum_{i=1}^{n} E(r_i)\, w_i \\ minimize\ f_2(x) = \sum_{i=1}^{n} \sum_{j=1}^{n} w_i w_j \sigma_{ij} \end{cases} \tag{9}$$

## 6. Results

Results are given in Figures 2, 3, and 4. On every plot axes are labeled as Return and Variance. Return in case stand for expected return of portfolio $E(r_p)$ and Variance stand for portfolio's risk defined as $\sigma_p^2$.

In Figure 2 we include five stocks in portfolio. Four of them have positive return and one has negative return. With NSGA-II we get a Pareto front with all different and equivalent portfolios. Solution obtained with simple GA is on that front regardless to size of generations. Stocks included in five stocks portfolio are in Table 1.

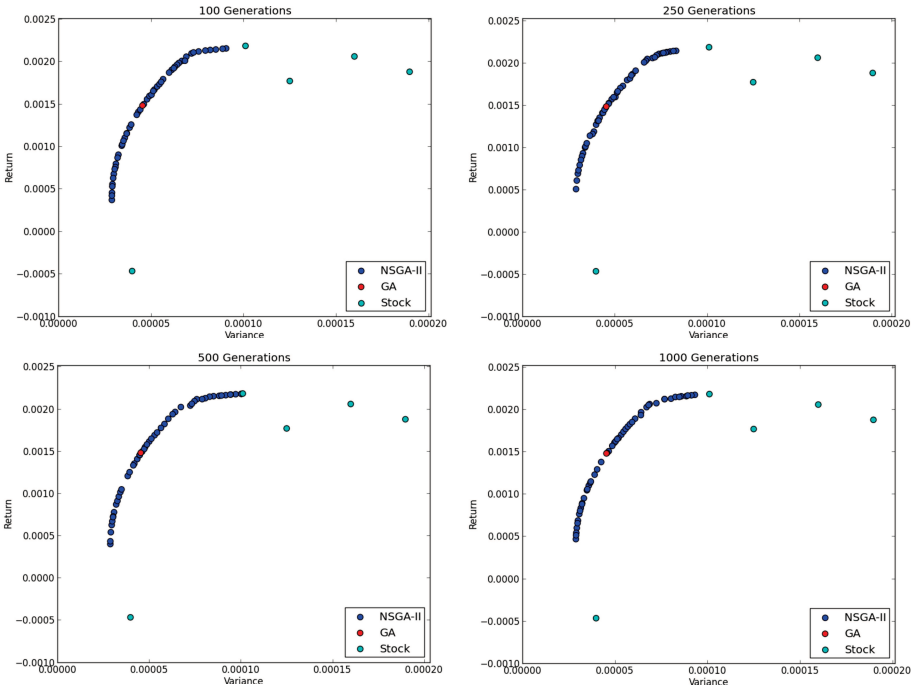



*Figure 2.* Results for five stocks in portfolio.

In Figure 3 we include ten stocks in portfolio. Nine of them has positive return and one with negative return and the lowest variance. Solution obtained with simple GA is again on the Pareto front in every generation size. Stocks included in ten stocks portfolio are in Table 1.

In Figure 4 we include twenty stocks in portfolio. Nineteen of them has positive return and one with negative return and the lowest variance. Again, solution obtained with simple GA is on the Pareto front regardless to generation size. Stocks included in twenty stocks portfolio are in Table 1.

We can say that results obtained with simple GA approach are comparable with results obtained with NSGA-II approach. Sometimes it even happened that a neighbor solution on the Pareto front was dominated by solution obtained with simple GA. We also made measurements of computation times for both techniques depending on portfolio size and number of generations. Simple GA needed significantly less time for its computation, regardless to number of generations or portfolio size. More details on computational are in Tables 3 and 4.

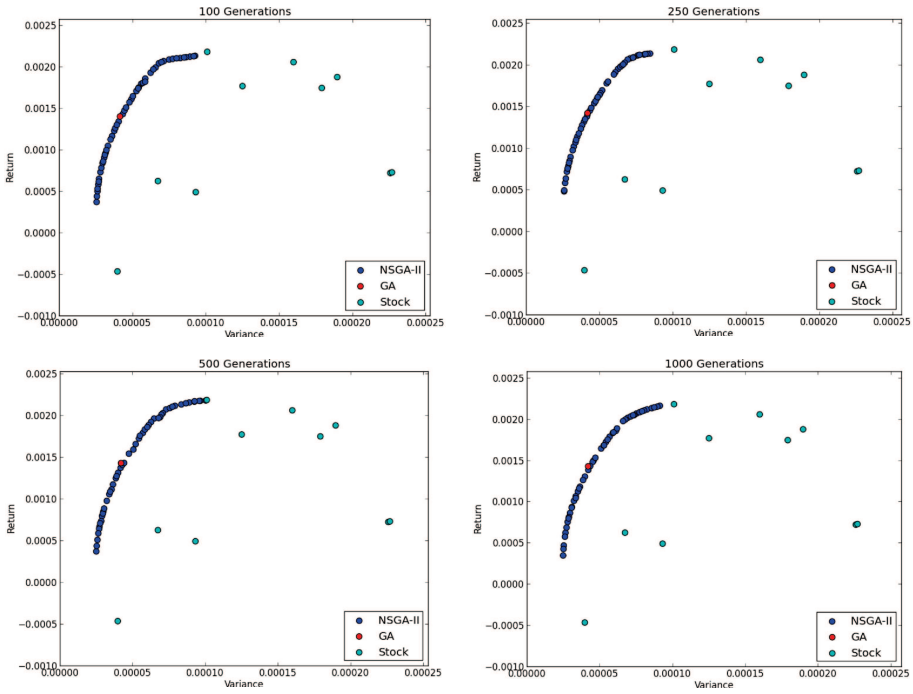



*Figure 3.* Results for ten stocks in portfolio.

*Table 3.* Computational times[a] of simple GA.

| | *Portfolio size* | | |
|---|---|---|---|
| *Number of generations* | *5* | *10* | *20* |
| 100 | 0,62 | 0,7 | 0,83 |
| 250 | 1,55 | 1,78 | 2,02 |
| 500 | 3,25 | 3,43 | 4,04 |
| 1000 | 6,42 | 7,06 | 8,06 |

[a] All computational times are in seconds.

## 7. Conclusion

In this paper, we applied stock portfolio optimization problem. Purpose of portfolio optimization is to achieve highest possible return at known risk rate or vice versa. Because exact methods like QP are time complex we solve problem with GA.

We compare performance both, with simple as well as multi objective GA. In research we used simple GA and NSGA-II approach in order

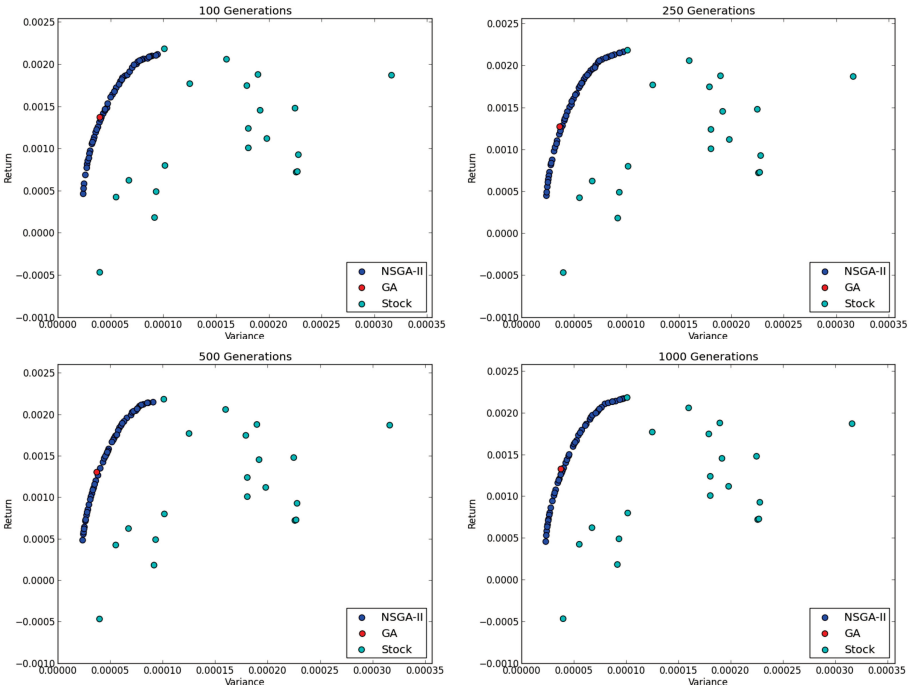



*Figure 4.* Results for twenty stocks in portfolio.

*Table 4.* Computational times[a] of NSGA-II.

| | *Portfolio size* | | |
|---|---|---|---|
| *Number of generations* | *5* | *10* | *20* |
| 100 | 83,19 | 83,8 | 84,67 |
| 250 | 206,16 | 209,08 | 210,33 |
| 500 | 414,24 | 418,86 | 423,97 |
| 1000 | 827,01 | 841,79 | 857,36 |

[a] All computational times are in seconds.

to find optimal stock portfolio, according to Markowitz mean-variance model. Results show that even if NSGA-II is more complex algorithm, its performance was not significantly better than the performance of simple GA. On the contrary, sometimes simple GA solution dominate its neighbor on the Pareto front. Simple GA also had significantly lower computation time irrespective of portfolio size or number of generations.

There are still some open questions, like how approaches will perform with different risk metrics because in mean-variance model stocks with lower variance are favorized regardless to their returns, or how they will perform if we add some real world constraints in model. And that is a starting point for future work.

## References

[1] E. Brigham and M. Ehrhardt. *Financial management theory and practice, 12th edition*, South-Western Cengage Learning, 2008

[2] K. Deb. *Multi-Objective Optimization Using Evolutionary Algorithms*. John Wiley & Sons, Chishester, 2001.

[3] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE T. Evolut. Comput.*, 6(2):182–197, 2002.

[4] R. L. Haupt and S. E. Haupt. *Practical Genetic Algorithm, 2nd edition*, Wiley, 2004.

[5] P. N. Kolm, R. Tütüncü, and F. Fabozzi. 60 years of portfolio optimization: Practical challanges and current trends. *Eur. J. Oper. Res.*, 234(2):356–371, 2014.

[6] C.-M. Lin and M. Gen. An effective decision-based genetic algorithm approach to multiobjective portfolio optimization problem. *Applied Mathematical Sciences*, 1(5):201–210, 2007.

[7] H. Markowitz. Portfolio selection. *J. Financ.*, 7(1):77–91, 1952.

[8] S. K. Mishra, G. Panda, S. Meher, R. Majhi, and M. Singh. Portfolio management assessment by four multiobjective optimization algorithm. In *Recent Advances in Intelligent Computational Systems (RAICS)*, 2011, pages 326–331.

[9] S. K. Mishra, G. Panda, S. Meher, and S. S. Sakhu. Optimal Weighting of Assets using a Multi-objective Evolutionary Algorithm. *Int. J. Recent Trends in Engineering*, 2(5):161–166, 2009.

[10] J. Shoaf and J. A. Foster. The efficient set GA for stock portfolios. In *Proc. IEEE World Congress on Computational Intelligence (WCCI)*, 1998, pages 354–359.

[11] Y. Xia, B. Liu, S. Wang, and K. K. Lai. A model for portfolio selection with order to expected returns. *Comput. Oper. Res.*, 27(5):409-422, 2000.

[12] `http://finance.yahoo.com/`

# SIMULATION-BASED GA OPTIMIZATION FOR PRODUCTION PLANNING

Juan Esteban Diaz Leiva
*Manchester Business School*
*University of Manchester, United Kingdom*
juan.diaz@postgrad.mbs.ac.uk

Julia Handl
*Manchester Business School*
*University of Manchester, United Kingdom*
j.handl@manchester.ac.uk

**Abstract** Effective production planning requires models that are capable of accounting for the complexity and uncertainty intrinsic to manufacturing systems. While the identification of a globally optimal plan is desirable, a more important requirement is the ability of a model to produce production plans that are sufficiently realistic to be implemented in practice and are robust to perturbations in the system. Here, we present a simulation-based optimization approach that employs discrete event simulation and a genetic algorithm as a methodology to support decision making in the area of production planning. The model aims to minimize the sum of expected backorders and inventory costs, while incorporating system constraints and the uncertainty that derives from variations of manufacturing lead times, occurrence of work centre failures and repair service times. Preliminary results for a real-world problem indicate that the model is capable of producing feasible production plans that correctly account for the uncertainty intrinsic to the underlying manufacturing system.

**Keywords:** Discrete event simulation, Genetic algorithms, Production planning, Stochastic variables, Uncertainty.

## 1. Introduction

Production planning, which specifies how resources should be allocated to production activities [16], forms an integral part of medium-term planning within manufacturing processes. Given the increasing

pressures faced by manufacturers, the development and deployment of effective models that support production planning is essential.

Ideally, an optimal production plan should be able to achieve customer satisfaction [18] along with profit maximization, while considering uncertainty in the system [14, 16]. Therefore, an appropriate methodology needs to perform optimization while accounting for the effects that uncertain parameters may have on the implementation of a production plan. This should then lead to an optimized solution that is robust towards various source of uncertainty in the manufacturing system. The lack of an instrument that is fully able to meet this requirement is one of the main reasons why, currently, decisions in production planning are often made in a subjective manner (based on the experience and "sixth sense" of a few people) or guided by inappropriate (simplistic) methodologies.

Optimization and simulation models have been previously deployed to solve the production planning problem, albeit independently. Optimization models are able to generate optimal or near-optimal solutions, but the real applicability of these solutions is often limited. This is because of the oversimplifying assumptions made by many optimization models and their inability to fully incorporate uncertainty [9, 17]. Furthermore, when trying to incorporate the high level of complexity and the stochastic [13] and dynamic nature of manufacturing systems [3] into optimization models, standard approaches become computationally intractable. On the other hand, simulation approaches are capable of capturing the uncertainty of the system [16] and of accurately reproducing its behaviour [11]. Therefore, simulation often provides a better representation of a real production system, since the variability introduced through exogenous and endogenous factors can be explicitly considered and the impact of these factors can be assessed [2]. However, in contrast to optimization approaches, the results obtained from simulation models are fundamentally descriptive: while a clear picture of the system is obtained, the results do not provide explicit guidance towards improved solutions.

In an attempt to combine the respective advantages of simulation and optimization techniques, simulation-based optimization has been suggested as a means of handling problems where the high level of complexity precludes a complete analytic formulation and the ultimate goal is the identification of a robust, near-optimal solution [10]. More specifically, the combined application of discrete event simulation (DES) and genetic algorithms (GAs) has been successfully applied to address several problems related to manufacturing systems. For instance, Azzaro-Pantel et al. [3] were able to improve the efficiency of a multi-purpose,

multi-objective plant with limited storage by accurately modelling the dynamic behaviour of the production system through DES and solving the scheduling problem using a GA. Al-Aomar [1] combined DES and a GA to determine robust design parameters. The author integrated Taguchis's robustness measures of signal-to-noise ratio and the quality loss function into a GA in order to enhance the selection scheme. Ding et al. [8] employed DES to capture the uncertainty involved in the supplier selection process and used a GA to optimize the supplier portfolio. Cheng and Yan [5] applied an integration of DES and a messy GA to determine the near optimal combination of resources in order to enhance the performance of construction operations. This approach enabled the authors to cope with the complexity and large dimensionality of the problem and to obtain adequate solutions. Wu et al. [21] integrated DES with a GA to determine the order point for different product types of a cross-docking center in order to minimize total cost. Through this approach the solution space was efficiently reduced and more simulation effort was allocated to promising areas via smart computing budget allocation. Korytkowski et al. [12] proposed an evolutionary simulation-based heuristics, where DES and a GA were deployed to find near optimal solutions for dispatching rules allocation. The sequence of orders determined through this approach improved the performance of a complex multi-stage, multi-product manufacturing system.

Here, we describe a simulation-based optimization approach for production planning. The long-term aim of our work is to derive an effective modeling approach that is capable of determining feasibe and robust monthly production plans. Here, we formulate production planning as an optimization problem that requires the minimization of the expected sum of backorders and inventory costs, subject to a set of constraints of the manufacturing system (e.g. resource constraints) and uncertainties deriving from variations of manufacturing lead times, occurrence of work centre failures and repair service times. Our choice of methodology is motivated by the proven success of simulation-based optimization in related problems (see [1, 3, 5, 8, 12, 21] and above), and we develop a model based on the combination of DES and a meta-heuristic optimizer (specifically, a GA). Finally, we describe preliminary results on a real-world production planning problem.

## 2. Simulation-based Optimization Model

The production planning problem considered here is based on the real manufacturing system of a large company that specializes in the production of cleaning products, edible shortenings, fats, and oils. This

study focuses exclusively on its activities related to the manufacturing of cleaning products.

Discrete Event Simulation (DES) is a good option to model the dynamic behaviour of this production system [3], as it allows for the incorporation of stochastic events and the variations of processes that occur in complex systems [19]. Specifically, the use of DES enables us to capture the uncertainty intrinsic to production planning that cannot be represented by deterministic models [16].

The application of simulation-based optimization implies the absence of an analytical problem formulation, i.e. the functional relationships between dependent and independent variables are not known explicitly [20]. Consequently, a suitable optimization approach needs to be able to perform optimization based exclusively on function values obtained via simulation, a so called "black-box approach". Considering the complexity and large dimensionality of the solution space, a suitable search strategy should be able to find near-optimal solutions in a large and complex solution space and be capable of escaping local optima. Finally, the optimization method needs to be robust with respect to noise, since the optimization procedure relies on stochastic responses generated by the simulation model [10]. Meta-heuristics present suitable candidates for this setting, and, in this study, a GA was selected as the optimizer. This choice was motivated by previous research indicating the robust performance of GAs under noisy conditions [4, 15], and, specifically, in the context of DES optimization [13].

The DES model of the production system was developed in SimEvents® (The MathWorks, Inc., 2013). This was integrated with MATLAB® R2013a (The MathWorks, Inc., 2013), and MATLAB's standard GA implementation was employed as the optimizer. Details of the simulation model and optimizer are described in the following sections.

## 2.1 Simulation Model

The DES model represents the production of 31 products $k$ within 7 work centres $l$. A work centre corresponds to the set of resources (e.g., machines, people, etc.) needed to manufacture certain products. Given that some products can be manufactured in several work centres a total of 41 processes $j$ are considered in the DES model. A process $j$ includes all series of events involved in the initialization of orders of a product $k$, its manufacture in a specific work centre $l$ and its storage in an specific sink $s$ (with $s = 1, 2, \ldots, 41$). Here orders are measured in number of lots. The simulation time $t$ of each simulation replication is 24 days, which corresponds to the number of working days in a month.

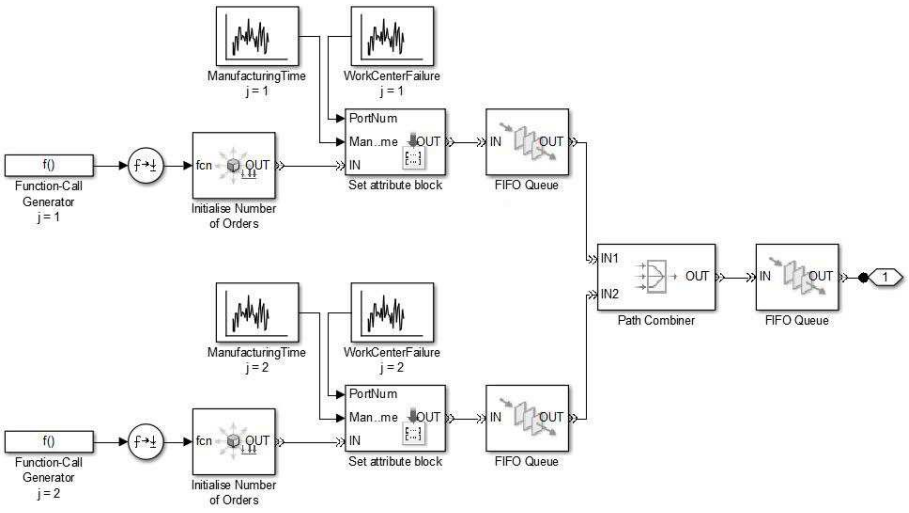



*Figure 1.* Order processing subsystem for work centre $l$.

The model component designed for the generation of orders for a single work centre $l$ is illustrated in Figure 1. The production plan to be simulated is determined by the decision variables $x_j$ (used as inputs for the function-call generator blocks), specified by the GA, and then the number of production orders for each process $j$ are initialized (by event-based entity generator blocks). Given that some products $k$ are required as raw materials during the manufacturing process of other products $k$, a higher priority is assigned to the initialization of orders for those sub-products in order to assure the static logic of the model.

Attributes are assigned to the different product lots (via attribute blocks). Specifications about the entity sink $s$ (with $s = 1, 2, \ldots, 41$) where final products will be stored are assigned via an attribute called $OutputPort_j$. Furthermore, the time required to manufacture a specific production lot ($ManufacturingTime_j$) and the occurrence of a failure in a work centre while processing a production lot ($WorkCentreFailure_j$) are additional attributes assigned to each lot of product. Two different event-based random number generators are employed to set the last two attributes mentioned. Both event-based random number generators produce a signal sampled randomly from the probability distribution functions (PDFs) assigned to them. A synthetic data set was employed to estimate PDFs for each stochastic variable included in the current study, as data collection for these aspects of the system is currently incomplete.

Once the attributes have been assigned to the production orders, those orders are transferred to a queue following a first-in first-out (FIFO)

discipline. Subsequently, those queues of production orders that have to be processed by the same work centre are merged (by a path combiner block) into a single FIFO queue.

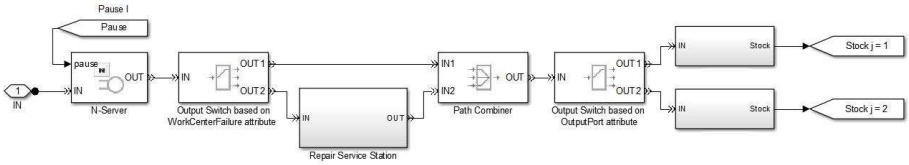



*Figure 2.* Production subsystem for work centre $l$.

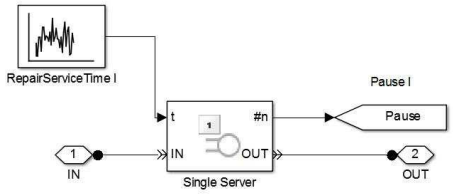



*Figure 3.* Repair service centre of work centre $l$.

The model components of a production subsystem and repair service centre are illustrated in Figure 2 and Figure 3, respectively. Each order is manufactured as soon as the corresponding work centre (represented by an N-server block) becomes available. In case of failure, the activity of that work centre is blocked by the control signal $Pause_l$. This signal is generated from the corresponding repair service centre and it outputs the number of entities present in that repair centre. Therefore, a signal with value greater than zero indicates that the work centre $l$ is being repaired and stops its activity until that signal becomes zero (no entities present in the repair service centre).

In case that no failure occurs ($WorkCentreFailure_j = 1$), the production batch is transferred to the corresponding sink $s$ (determined by $OutputPort_j$). Whereas if a failure occurs ($WorkCentreFailure_j = 2$), that product batch is transferred to a repair service centre prior to its storage. The delay caused by the work centre failure is sampled from the corresponding PDF assigned to $RepairServiceTime_l$. One important assumption made is that after a production batch has left the repair service centre no re-manufacture is required, since the manufacturing process has been already completed (passed through the N-server block). This is an effective way to model system failure without having conflicting events.

The stock of product $k$ manufactured in work centre $l$, denoted by $Stock_{kl}$, is collected at the end of every replication and it is measured in number of lots. Based on $Stock_{kl}$, the total stock of product $k$ ($Stock_k$) is calculated at the end of every replication as follows:

$$Stock_k = \sum_{l=1}^{7} Stock_{kl} \tag{1}$$

consequently, for products manufactured by a single work centre this formula reduces to:

$$Stock_k = Stock_{k1} \tag{2}$$

## 2.2 Optimization Model

The decision variables, denoted by $x_j$, are the number of lots to be produced in process $j$. A black box optimization approach is applied in which the decision variables specified by the GA provide the input to the DES model and the responses $Stock_k$ from the DES model are employed to compute the value of the fitness function. A total of 41 decision variables $x_j$, which are constrained to be positive integers, are considered in the model. Given the stochastic nature of the DES outputs, fitness is evaluated across $n$ independent simulation trials (with $n = 10$). Specifically, the fitness value $f$ is estimated for each individual $x$ as follows:

$$f(x) = \bar{c} = \frac{1}{n} \sum_{m=1}^{n} c_m \tag{3}$$

For each replication $m$, the response ($Stock_k$) of the DES model is used to calculate $c_m$ as follows:

$$c_m = \sum_{k=1}^{31} InventoryCost_k + BackorderCost_k. \tag{4}$$

Inventory and Backorder Costs are defined as:

$$InventoryCost_k = \begin{cases} (Stock_k - D_k) \times Cost_k & \text{if } Stock_k > D_k \\ 0 & \text{if } Stock_k \leq D_k \end{cases}$$

and

$$BackorderCost_k = \begin{cases} (D_k - Stock_k) \times Price_k & \text{if } Stock_k < D_k \\ 0 & \text{if } Stock_k \geq D_k \end{cases}$$

where, $D_k$ indicates the demand for product $k$. Unsold amounts of product $k$ are penalized proportionally to the corresponding standard cost

per lot ($Cost_k$), whereas backorders receive a fine equal to the product price, which is the income lost ($Price_k$) for not selling that specific amount of product. Given a lack of information on real inventory costs and total cost derived from backorders per product (cost of customer dissatisfaction, cost of non-future purchases, cost of customers switching to other brands, etc.), standard costs and product prices are currently employed to penalize inventory and backorders, respectively. These two assumptions are not valid in reality for several reasons. First, excess of inventory can be sold in future periods and inventory costs are not equal to standard costs. Second, considering product price as the total loss caused by product backorders is inaccurate and unrealistic.

Additional constraints are imposed given that some products $k$ are required as raw materials during the manufacturing process of other products $k$. Therefore, the requirement of sub-products is represented through linear constraints as follows:

$$\sum_{j=1}^{41} a_{ij} \times x_j \leq b_i \qquad (i = 1, 2, \ldots, 4) \tag{5}$$

where $b_i$ denotes the quantity available of sub-product $i$ and $a_{ij}$ is the amount required of sub-product $i$ to produce one lot in process $j$.

The default MATLAB implementation for solving integer and mixed integer problems using a GA is applied in the current study. A detailed description of the (MI-LXPM) GA and its truncation procedure (which ensures compliance with integer constraints after crossover and mutation) can be found in [7]. The inbuilt constraint-handling approach is the parameter free penalty function approach proposed by Deb [6].

## 3. Preliminary Results

The model enables an accurate incorporation of uncertainty derived from variations of manufacturing lead times, occurrence of work centre failures and repair service times. The time required to run 15 iterations of the GA is 12.15 hours. For this reason, very limited results are reported in the present study, and we mostly focus on the validity of the model designed. More extensive benchmarking of the approach (including longer optimization runs and statistics across multiple trials) is currently in progress.

As shown in Figure 4, when run for 15 iterations, the GA successfully reduces the expected sum of backorders and inventory costs. The best production plan after 15 iterations is presented in Table 1. For reference, the amount of demand to be covered, the consolidated number of lots per product to be manufactured and the actual number of lots produced is

shown in Table 2. The allocation for work centre 204 provides a suitable illustration of the results obtained. For the majority of products (except for product B), work centre 204 displays a more reliable performance than work centre 203. For this reason, the suggested production plan (see Table 1) allocates a greater number of orders to work centre 204. This solution is in accordance with our expectations and illustrates that the reliability of work centres is correctly accounted for in the production plan generated.

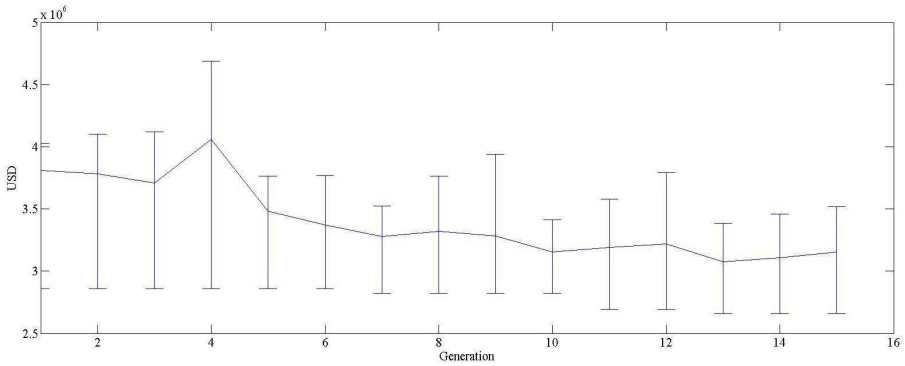



*Figure 4.* Best, mean and worst fitness value of the population at each iteration.

## 4. Future Research

There are a number of ways in which this research will be extended in future work. Regarding the simulation component of the work, data collection (from the company) needs to be completed. The data obtained will be used to estimate PDFs of all stochastic variables, so that the use of synthetic data can be avoided.

Regarding the optimizer, future work will include an investigation of parameter settings, the sensitivity to noise and, potentially, the comparison to alternative meta-heuristic optimization approaches. Moreover, the number ($n$) of simulation trials employed to evaluate fitness will be further analysed in order to balance quality of estimations and computational cost. Furthermore, a multi-objective formulation of the problem will be explored in order to account for the robustness of solutions in a more explicit manner. Specifically, the maximization of the signal-to-noise ratio may be used as an additional objective to directly account for the variability in the fitness values obtained [1].

*Table 1.* Number of product lots to be manufactured in a specific work centre.

| *Product* | *Work centre* | *Prod. plan*[b] |
|---|---|---|
| A | 204 | 14 |
| B | 203 | 7 |
| B | 204 | 6 |
| C | 203 | 10 |
| D | 203 | 6 |
| E | 203 | 10 |
| F | 203 | 8 |
| G | 203 | 7 |
| G | 204 | 14 |
| H | 203 | 3 |
| H | 204 | 11 |
| I | 203 | 3 |
| I | 204 | 8 |
| J | 203 | 14 |
| K | 203 | 6 |
| L | 203 | 12 |
| L | 204 | 13 |
| M | 203 | 7 |
| M | 204 | 19 |
| N | 204 | 14 |
| O | 203 | 5 |
| O | 204 | 10 |
| P | 203 | 4 |
| P | 204 | 4 |
| Q | 203 | 12 |
| R | 202 | 5 |
| R | 203 | 3 |
| R | 204 | 7 |
| S | 202 | 7 |
| T | 203 | 9 |
| U | 203 | 12 |
| V | 203 | 13 |
| W | 203 | 2 |
| X | 202 | 9 |
| Y[b] | 101 | 3 |
| Z[b] | 204 | 10 |
| AA | 204 | 6 |
| AB[b] | 205 | 8 |
| AC[b] | 205 | 16 |
| AD | 208 | 7 |
| AE | 301 | 9 |

[a] best production plan generated by the model after 15 iterations with $n = 10$.
[b] Sub-products.

*Table 2.* Demand, consolidated production plan per product and actual production achieved.

| *Product* | *Demand* [a] | Production [a] *Planned* | *Actual* [b] |
|---|---|---|---|
| A | 10 | 14 | 7.3 |
| B | 9 | 13 | 6.1 |
| C | 12 | 10 | 4 |
| D | 10 | 6 | 2.4 |
| E | 15 | 10 | 4 |
| F | 13 | 8 | 3.2 |
| G | 12 | 21 | 10.1 |
| H | 11 | 14 | 7 |
| I | 9 | 11 | 5.5 |
| J | 8 | 14 | 5.6 |
| K | 15 | 6 | 2.4 |
| L | 12 | 25 | 11.6 |
| M | 13 | 26 | 12.6 |
| N | 12 | 14 | 7.3 |
| O | 13 | 15 | 7.3 |
| P | 9 | 8 | 3.9 |
| Q | 11 | 12 | 4.8 |
| R | 10 | 15 | 10 |
| S | 9 | 7 | 7 |
| T | 15 | 9 | 3.6 |
| U | 15 | 12 | 4.8 |
| V | 12 | 13 | 5.2 |
| W | 9 | 2 | 0.8 |
| X | 9 | 9 | 9 |
| Y[c] | 0 | 3 | 3 |
| Z[c] | 0 | 10 | 5.3 |
| AA | 10 | 6 | 3.3 |
| AB[c] | 0 | 8 | 8 |
| AC[c] | 0 | 16 | 16 |
| AD | 10 | 7 | 7 |
| AE | 10 | 9 | 9 |

[a] measured in number of lots.
[b] average value of 10 independent replications.
[c] Sub-products.

## References


[1] R. Al-Aomar. Incorporating robustness into genetic algorithm search of stochastic simulation outputs. *Simul. Model. Prac. Th.*, 14(3):201–223, 2006.

[2] J. April, M. Better, F. Glover, J. Kelly, and M. Laguna. Enhancing business process management with simulation optimization. In *Proc. 38th Conference on Winter Simulation*, pages 642–649, 2006.

[3] C. Azzaro-Pantel, L. Bernal-Haro, P. Baudet, S. Domenech, and L. Pibouleau. A two-stage methodology for short-term batch plant scheduling: discrete-event simulation and genetic algorithm. *Comput. Chem. Eng.*, 22(10):1461–1481, 1998.

[4] E. B. Baum, D. Boneh, and C. Garrett. On genetic algorithms. In *Proc. 8th Annual Conference on Computational Learning Theory*, pages 230–239, 1995.

[5] T.-M. Cheng and R.-Z. Yan. Integrating messy genetic algorithms and simulation to optimize resource utilization. *Comput.-Aided Civil .Inf.*, 24(6):401–415, 2009.

[6] K. Deb. An efficient constraint handling method for genetic algorithms. *Comput. Methods Appl. M.*, 186(2):311–338, 2000.

[7] K. Deep, K. P. Singh, M. Kansal, and C. Mohan. A real coded genetic algorithm for solving integer and mixed integer optimization problems. *Appl. Math. Comput.*, 212(2):505–518, 2009.

[8] H. Ding, L. Benyoucef, and X. Xie. A simulation optimization methodology for supplier selection problem. *Int. J. Comp. Integ. M.*, 18(2-3):210–224, 2005.

[9] M. Gnoni, R. Iavagnilio, G. Mossa, G. Mummolo, and A. Di Leva. Production planning of a multi-site manufacturing system by hybrid modelling: A case study from the automotive industry. *Int. J. Prod. Econ.*, 85(2):251–262, 2003.

[10] G. Gray, K. Fowler, and J. Griffin. Hybrid optimization schemes for simulation-based problems. *Procedia Computer Science*, 1(1):1349–1357, 2010.

[11] S.-J. Hsieh. Hybrid analytic and simulation models for assembly line design and production planning. *Simul. Model. Prac. Th.*, 10(1-2):87–108, 2002.

[12] P. Korytkowski, T. Wiśniewski, and S. Rymaszewski. An evolutionary simulation-based optimization approach for dispatching scheduling. *Simul. Model. Prac. Th.*, 35:69–85, 2013.

[13] T. Lacksonen. Empirical comparison of search algorithms for discrete event simulation. *Comput. Ind. Eng.*, 40(1):133–148, 2001.

[14] P. Li, M. Wendt, and G. Wozny. Optimal production planning for chemical processes under uncertain market conditions. *Chem. Eng. Technol.*, 27(6):641–651, 2004.

[15] M. Mitchell. *An introduction to genetic algorithms.* MIT press, 1998.

[16] L. Monostori, G. Erdős, B. Kádár, T. Kis, A. Kovács, A. Pfeiffer, and J. Váncza. Digital enterprise solution for integrated production planning and control. *Comput. Ind.*, 61(2):112–126, 2010.

[17] A. Nikolopoulou and M. G. Ierapetritou. Hybrid simulation based optimization approach for supply chain management. *Comput. Chem. Eng.*, 47:183–193, 2012.

[18] Y. Pochet. Mathematical programming models and formulations for deterministic production planning problems. In *Computational Combinatorial Optimization*, pages 57–111. Springer, 2001.

[19] L. A. Riley. Discrete-event simulation optimization: a review of past approaches and propositions for future direction. In *Proc. 2013 Summer Computer Simulation Conference*, page 47, 2013.

[20] I. Steponavičė, S. Ruuska, and K. Miettinen. A solution process for simulation-based multiobjective design optimization with an application in the paper industry. *Comput. Aided Design*, 47:45–58, 2014.

[21] Y. Wu, M. Dong, and D. Yang. Cross-docking centre operation optimization using simulation-based genetic algorithm. *P. I. Mech. Eng. B-J. Eng.*, 225(7):1175–1187, 2011.

# MULTI-POPULATION ADAPTIVE INFLATIONARY DIFFERENTIAL EVOLUTION

Marilena Di Carlo
*Mechanical and Aerospace Engineering*
*University of Strathclyde, Glasgow, United Kingdom*
marilena.di-carlo@strath.ac.uk

Massimiliano Vasile
*Mechanical and Aerospace Engineering*
*University of Strathclyde, Glasgow, United Kingdom*
massimilano.vasile@strath.ac.uk

Edmondo Minisci
*Mechanical and Aerospace Engineering*
*University of Strathclyde, Glasgow, United Kingdom*
edmondo.minisci@strath.ac.uk

**Abstract** In this paper, a multi-population version of Adaptive Inflationary Differential Evolution, which automatically adapts the crossover probability and the differential weight of the Differential Evolution, is presented. The multi-population algorithm exploits the use of different populations, and the local minima found by each population, to assess the distance between minima; a probabilistic kernel based approach is then used to automatically adapt the dimension of a bubble in which the population is re-initialized after converging to a local minimum. The algorithm is tested on two real case functions and on two difficult academic functions.

**Keywords:** Adaptive algorithms, Differential evolution, Global optimization.

## 1. Introduction

Differential Evolution (DE), [13], is a population-based stochastic algorithm for solving optimization problems. Although it has proved to be a very efficient global optimizer, work has been done to enhance its performance by combining it with deterministic or stochastic optimizers [4, 5, 15]. In [19], Inflationary Differential Evolution Algorithm (IDEA) was introduced. IDEA is based on the hybridization of Differential Evolution (DE) with the restarting procedure of Monotonic Basin Hopping (MBH) algorithm [20]. The performance of IDEA was found to be dependent upon the parameters controlling both the DE and MBH heuristics [19]. In particular, the DE performance is strongly influenced by the crossover probability, CR, and the differential weight, F, whose best settings are heavily problem dependent [8].

The need to have an algorithm capable of self-adapting these two parameters has resulted in many works [1, 3, 10, 12, 14]. The next step in the development of IDEA has therefore been the adaptation of $CR$ and $F$, leading to Adaptive Inflationary Differential Evolution Algorithm (AIDEA) [11]. This algorithm uses a probabilistic kernel based approach to automatically adapt the values of both CR and F.

Starting from the successful results of AIDEA, this paper introduces a multi-population version of AIDEA (MP-AIDEA), using different strategies to create the mutant vector of the DE, different strategies to adapt $CR$ and $F$ and a new mechanism to adapt the dimension of the search space in which the population is re-initialized. Other multi-populations DE algorithms have been presented in [16, 21, 22].

In the first part of this paper MP-AIDEA is described. Then the results of four test cases are presented.

## 2. Multi-Population Adaptive Inflationary Differential Evolution Algorithm

The algorithm presented in this paper is a further development of AIDEA [11]. In the following, a summary of AIDEA and a detailed description of MP-AIDEA are given.

**AIDEA.** The first step in the run of AIDEA is a DE process in which each element of the population is associated to a different value of $CR$ and $F$. During the advancement of the population from parents to children the values of $CR$ and $F$ are adapted using a kernel based approach. The DE is run until the population contracts below a threshold, identified by the parameter $\rho$. When this contraction condition is satisfied a local search is performed from the best individual in the population.

The found local minimum is archived in a matrix of minima and the population is restarted in a bubble of dimension $\delta_{local}$ around the local minimum (local restart). Local restart is iterated up to a predefined maximum value, identified by the value $iun$. When this value is reached the population is restarted at a distance $\delta_{global}$ from the cluster of local minima found thus far (global restart). The algorithm stops when the maximum number of function evaluation is reached.

**MP-AIDEA.** In MP-AIDEA the single population of AIDEA is replaced by many populations. The common archive of local minima of all the populations can be used to create the mutant vector of the DE. Three strategies have been considered for the generation of the mutant vector: 1) *DE/best/1-DE/rand/1*: the mutant vector is created randomly using the best element or a random element of the population; 2) *DE/arch/1-DE/rand/1*: the mutant vector is created randomly using an element from the archive of local minima or a random element of the population: 3) *DE/arch/1-DE/best/1*: the mutant vector is created randomly using an element from the archive of local minima or the best element of the population.

As regards the adaptation, the presence of many populations can be exploited to adapt $CR$ and $F$ in a different way with respect to AIDEA. Two strategies for the adaptation of $CR$ and $F$ are proposed: 1) MP-AIDEA-CRF1 ($CR$ and $F$ adaptivity realized using $CR$ and $F$ values equal for every element of each population and comparing the populations to each other) and 2) MP-AIDEA-CRF2 ($CR$ and $F$ adaptivity realized using different $CR$ and $F$ values for each element of each population and comparing elements of each single population to each other, as in AIDEA [11]).

Finally, a strategy is proposed to adapt also the dimension of the bubble for the local restart of the population, using a kernel based approach similar to the one used for the adaptation of $CR$ and $F$. Considering all these possibilities, twelve different versions of the algorithm have been developed and tested:

- MP-AIDEA 1: MP-AIDEA-CRF1-*DE/best/1-DE/rand/1*
- MP-AIDEA 2: MP-AIDEA-CRF1-*DE/arch/1-DE/rand/1*
- MP-AIDEA 3: MP-AIDEA-CRF1-*DE/arch/1-DE/best/1*
- MP-AIDEA 4 to 6: MP-AIDEA 1 to 3 with $\delta_{local}$ adaptation
- MP-AIDEA 7 to 12: MP-AIDEA 1 to 6 based on CRF2 strategy

A detailed description of the common structure of the algorithms is given in Algorithm 1. The procedure starts by setting values for $n_{pop}$

(number of elements in each population), $N_{pop}$ (number of populations), $iun$ (maximum number of local restart), $\bar{\rho}$ (size of the convergence box), $\delta_{global}$ (distance from the cluster centres for the global restart) and $\delta_{local}$ (dimension of the bubble for the local restart, if not adapted) as in line 1 and initializing the populations (line 3). The joint PDF for $CR$ and $F$ is then initialised to be a uniform distribution (lines 4 and 5). For MP-AIDEA-CRF1, DE is run (line 11) drawing probabilistically a value for $F$ and $CR$ from **CRF** for each population (line 9) and **CRF** is updated on the basis of the improvement of the populations (step 15). For MP-AIDEA-CRF2, lines 9, 15 and 16 are to be considered inside the **for** cycle over the elements of the population (different values of $CR$ and $F$ for each element of the populations). If the populations contracts below a predefined threshold (step 18), a local optimizer is run from the current minimum (line 19) and the found local minimum is saved in an archive of local minima of all the populations (line 32). $iun_m$ is updated based on the improvement of the value of $f_{min,m}$ (lines 23 to 28). If the adaptation of $\delta_{local}$ is performed, when all the population have performed the local search, a matrix **B** for the adaptation of the dimension of the bubble can be created (step 34, Algorithm 3) using the local minima found thus far. At this point the populations go through local or global restart according to lines 39 to 45. In particular, if the local optimizer failed to improve the value of $f_{min}$ more than $iun_{max}$ times, the population is restarted globally and $iun$ is set to 0, otherwise the population is restarted within a local bubble and $iun = iun+1$. The dimension of the bubble for the local restart is sampled from matrix **B** (line 40) or is the one defined at line 1 if $\delta_{local}$ is not adapted. The adaptation of **B** (line 36) is done only when the local optimizer has been applied to all the population for the second time (for each population, the adaptation can be performed only if two local minima are known for that population). At this point, the loop restart from the initialization of **CRF**. As a terminal criterion the algorithms stops if the maximum number of function evalutation $n_{feval,max}$ has been reached.

**CR and F adaptation.** The updating procedure for **CRF** is detailed in Algorithm 2 for MP-AIDEA-CRF1. For each population, the maximum objective function difference between parents and children, $dd_{max}$, is computed (line 1). Then the element of **CRF** are sequentially evaluated and the first time that the $dd$ value associated to the considered row is lower than $dd_{max}$ (line 4) the value of $F$ used for the considered population substitutes the corresponding elements $\mathbf{CRF}_{j,2}$ (line 5). For $CR$, the value associated to the considered population sub-

stitutes $\mathbf{CRF}_{j,1}$ (line 7) only if $dd_{max}$ is greater than a given value $CRC$ (line 6), [11].

For MP-AIDEA-CRF2 the **for** cycle in line 2 is substituted by a **for** cycle over the elements of the single population and $dd_{max}$ is replaced by $f(\mathbf{x}_{i,k}) - f(\mathbf{x}_{i,k+2})$ for each element $i$ of the population, as in [11].

**$\delta$ adaptation.** The generation of the matrix $\mathbf{B}$ for the adaptation of the dimension of the bubble for the local restart is described in Algorithm 3; it reflects the procedure for the creation of **CRF** and is based on the computation of the distance between the local minima found by the different populations and stored in the common archive of local minima. The updating procedure for $\mathbf{B}$, detailed in Algorithm 4, follows the same approach used for the adaptation of **CRF**; the distance between local minimum found at subsequent local restart is used to assess the validity of the used dimension of the bubble for the local restart (a local restart is effective if the algorithm move from a local minimum to another).

The process of adaptation of the dimension of the bubble for the local restart will be presented in greater details in the first two test cases of the Test Results section.

## 3. Test Results

The test cases are taken from the technical report of the CEC 2005 and CEC 2011 competitions [6,17]. The considered problems are: Spread Spectrum Radar Polyphase Code Design and Tersoff Radar Function Minimization Problem from CEC 2011; Schwefel's Problem, Function 12, and Rotated Version of Hybrid Composition Function, Function 16, from CEC 2005. The statistics reported are computed on the results obtained from 100 independent runs in which new populations are generated at each run. The success rate reported in the next tables and figures is computed as number of times (over the 100 runs) in which the minimum found by the algorithm is lower than $f_{min}+\epsilon$ where $f_{min}$ is the minimum value of the function and $\epsilon$ is a given threshold [18]. $\epsilon = 0.001$ for the test cases from CEC 2011 and $\epsilon = 0.01$ for the CEC 2005 problems [17].

### 3.1 Spread Spectrum Radar Polyphase Code Design

This problem has dimension $n_D = 20$ and the best solution found is $f_{min} = 0.5$. The maximum number of function evaluation is 1.5e5 [6]. The same parameters setting of [11] was used, that is $\delta_{local} = \delta_{global} = 0.1$, $\rho = 0.2$ and $iun = 10$. The results obtained using AIDEA in [11] (where the DE strategy was *DE/best/1*) and new results obtained using AIDEA with the DE strategy *DE/best/1-DE/rand/1* are reported in

**Algorithm 1** Multi-Population Adaptive Inflationary Differental Evolution Algorithm

Set values for $n_{pop}$, $N_{pop}$, $iun$, $\bar{\rho}$, $\delta_{global}$
Set $n_{feval,m} = 0$ and $k_m = 1$ for all populations $m \in [1, \ldots, N_{pop}]$
Initialize population $\mathbf{x}_{m,i,k}$ for all $m \in [1, \ldots, N_{pop}]$ and for all $i \in [1, \ldots, n_{pop}]$
A regular mesh with $(n_D + 1)^2$ points (where $n_D$ is the dimensionality of the problem) in the space $CR \in [0.1, 0.99] xF \in [-0.5, 1]$ is created
Initialize **CRF** with points of the mesh: $\mathbf{CRF}_{j,1} \leftarrow \mathbf{CR}_j$ and $\mathbf{CRF}_{j,2} \leftarrow \mathbf{F}_j$ for all $j \in [1, \ldots, (n_D + 1)^2]$
Associate to each row of **CRF** an element $dd_j = 0$ for all $j \in [1, \ldots, (n_D + 1)^2]$
Row sort **CRF** in terms of $dd$ values
**for** $m \in [1, \ldots, N_{pop}]$ **do**
    Sample $\mathbf{CR}_{m,k}$ and $\mathbf{F}_{m,k}$ from **CRF**
    **for** $i \in [1, \ldots, n_{pop}]$ **do**
        $\mathbf{x}_{m,i,k+1} \leftarrow \mathrm{DE}(\mathbf{x}_{m,i,k}, \mathbf{CR}_{m,k}, \mathbf{F}_{m,k})$
        $n_{feval,m} = n_{feval,m} + 1$
    **end for**
    $k_m = k_m + 1$
    Update **CRF** (see Algorithm 2)
    Row sort **CRF** in terms of $dd$ values
    $\rho_m = \max\left(||\mathbf{x}_{m,i,k} - \mathbf{x}_{m,j,k}||\right) \quad \forall \mathbf{x}_{m,i,k}, \mathbf{x}_{m,j,k} \in P_{m,k}$
    **if** $\rho_m < \bar{\rho} \cdot \rho_{max,m}$ **then**
        Run a local optimizer from $\mathbf{x}_{best,m}$ and let $\mathbf{x}_{l,m}$ be the local minimum found by the local optimizer
        **if** $f(\mathbf{x}_{l,m}) < f(\mathbf{x}_{best,m})$ **then**
            $f(\mathbf{x}_{best,m}) \leftarrow f(\mathbf{x}_{l,m})$
        **end if**
        **if** $f(\mathbf{x}_{best,m}) < f_{min,m}$ **then**
            $f_{min,m} \leftarrow f(\mathbf{x}_{best,m})$
            $iun_m = 0$
        **else**
            $iun_m = iun_m + 1$
        **end if**
    **else**
        **Termination** Unless $n_{feval,m} \geq n_{feval,max}$ goto (10)
    **end if**
    Add $\mathbf{x}_{best,m}$ to the archive of minima of population: $A_{g,m} = A_{g,m} + \{\mathbf{x}_{best,m}\}$
**end for**
Create matrix **B** for adaptation of the dimension of the bubble (see Algorithm 3)
**if** (All population went for the 2nd time through the local minimizer) **then**
    Update **B** (see Algorithm 4)
**end if**
**for** $m \in [1, \ldots, N_{pop}]$ **do**
    **if** $iun \leq iun_{max}$ **then**
        Sample $\delta_{local,m}$ from **B** to define the bubble $D_m$
        Initialize population $\mathbf{x}_{m,i,k}$ for all $i \in [1, \ldots, n_{pop}]$ in the bubble $D_m$
    **else**
        Define clusters in the archive and compute baricentre $\mathbf{x}_{c,m}$ of each cluster
        Initialize population $\mathbf{x}_{m,i,k}$ for all $i \in [1, \ldots, n_{pop}]$ such that $\forall i, j ||\mathbf{x}_{m,i,k} - \mathbf{x}_{m,j,k}|| > \delta_{global}$
    **end if**
**end for**
**Termination** Unless $n_{feval,m} \geq n_{feval,max}$ goto (4)

Table 1, along with the results of two of the best performing algorithms of the CEC 2011 competition, the Genetic Algorithm with Multi Parent Crossover (GA-MPC) [7] and the Weed Inspired Differential Evolution (WI-DE) [9].

AIDEA gives better results than GA-MPC or WI-DE. In Figure 1 the success rates obtained using different population number $N_{pop}$ composed

**Algorithm 2** Updating procedure for **CRF**

For each population compute $dd_{max,m} = max||f(\mathbf{x}_{m,i,k+1}) - f(\mathbf{x}_{m,i,k})||$ for all $i \in [1, \ldots, n_{pop}]$
**for** $m \in [1, \ldots, N_{pop}]$ **do**
  **for** $j \in [1, \ldots, (n_D + 1)^2]$ **do**
    **if** $dd_j < dd_{max,m}$ **then**
      $\mathbf{CRF}_{j,2,k} \leftarrow \mathbf{F}_{m,k}$
      **if** $dd_{max,m} > CRC$ **then**
        $\mathbf{CRF}_{j,1,k} \leftarrow \mathbf{CR}_{m,k}$
      **end if**
    **end if**
  **end for**
**end for**

**Algorithm 3** Generation of matrix **B** for the adaptation of the bubble

Compute mean and minimum distance between all local minima in $A_{gm}$ for all $m \in [1, \ldots, N_{pop}]$: $\mathbf{d}_{minMIN}$ and $\mathbf{d}_{minMEAN}$
Create regular mesh with $(n_D + 1)^2$ points in the space $[\mathbf{d}_{minMIN}, \mathbf{d}_{minMEAN}]$
Initialize **B** with points of the mesh
Associate to each row of **B** an element $dd_{bj} = 0$ for all $j \in [1, \ldots, (n_D + 1)^2]$
Row sort **B** in terms of $dd_b$ values

**Algorithm 4** Updating procedure for **B**

For each population compute $p_m = ||\mathbf{x}_{l,m,k+1} - \mathbf{x}_{l,m,k}||$
**for** $m \in [1, \ldots, N_{pop}]$ **do**
  **for** $j \in [1, \ldots, (n_D + 1)^2]$ **do**
    **if** $dd_{bj} < p_m$ **then**
      $\mathbf{B}_{j,1,k} \leftarrow \delta_{local,m}$
    **end if**
  **end for**
**end for**

by $n_{pop} = 10$ elements are shown for MP-AIDEA 1, 4, 7 and 10 and MP-AIDEA 2, 5, 8 and 11. Results from MP-AIDEA 3, 6, 9, 12 are very similar to MP-AIDEA 2, 5, 8, 11 and therefore are not shown. AIDEA was tested with a number of individuals in the single population equal to the total number of individuals of MP-AIDEA. The most successful versions of MP-AIDEA are 1, 7 and 10; their results are always better than AIDEA's one. MP-AIDEA versions 2, 5, 8 and 11 show a success

*Table 1.* Spread Spectrum Radar Polyphase Code Design – AIDEA, GA-MPC and WI-DE results.

| Algorithm | $n_{pop}$ | Min | Mean | Max | Str.Dev. | S.Rate |
|---|---|---|---|---|---|---|
| AIDEA *DE/best* | 20 | 0.5000 | 0.5150 | 0.6509 | 0.0343 | - |
| AIDEA DE/*best*-DE/*rand* | 20 | 0.5000 | 0.5130 | 0.6422 | 0.0263 | 75 |
| GA-MPC | - | 0.5000 | 0.7484 | 0.9334 | 0.1249 | - |
| WI-DE | - | 0.5000 | 0.656 | 0.993 | 0.116 | - |

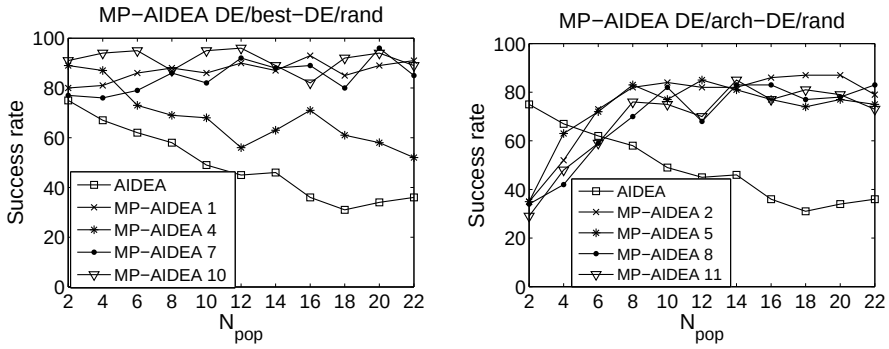



*Figure 1.* Spread Spectrum Radar Polyphase Code Design – MP-AIDEA success rate.

rate increasing with $N_{pop}$ and greater than the success rate of AIDEA for $N_{pop}$ sufficiently high.

In Figure 2 the process of adaptation of the dimension of the bubble for the local restart is shown for MP-AIDEA 4 and $N_{pop} = 3$ for a sequence of 19 subsequent local restarts before the global restart of the algorithm. The bold line represents the mean value of $\delta_{local}$ over all the populations. It is evident that $\delta_{local} = 0.1$ proves to be a good guess for the value of $\delta_{local}$.

## 3.2 Tersoff Potential Function Minimization Problem

This problem has dimension $n_D = 30$ and the best solution is $f_{min} = -36.9286$. The maximum number of function evaluation is 1.5e5. AIDEA and MP-AIDEA were tested using two different sets of parameters settings: $\delta_{local} = \delta_{global} = 0.1$, $\rho = 0.2$, $iun = 10$ (Case 1) and $\delta_{local} = 0.3$, $\delta_{global} = 0.1$, $\rho = 0.2$, $iun = 10$ (Case 2). The results obtained using AIDEA in [11] and new results obtained using AIDEA with the DE strategy DE/*rand/1*-DE/*best/1* are reported in Table 2, along with the results obtained by GA-MPC and WI-DE.

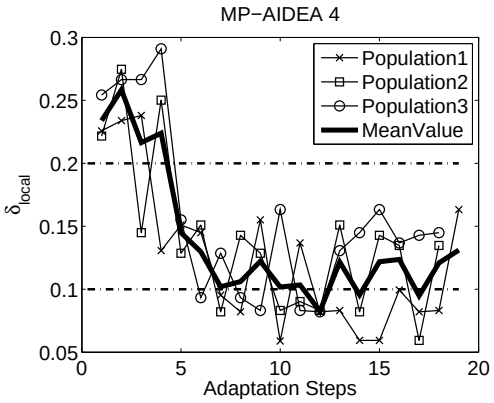



*Figure 2.* Spread Spectrum Radar Polyphase Code Design – adaptation of $\delta_{local}$.

*Table 2.* Tersoff Potential Function Minimization Problem – AIDEA, GA-MPC and WI-DE results.

| Algorithm | $n_{pop}$ | Min | Mean | Max | Str.Dev. | S.Rate |
|---|---|---|---|---|---|---|
| Case 1 | | | | | | |
| AIDEA DE/*best* | 20 | -36.9286 | -36.8527 | -35.5171 | 0.2442 | - |
| AIDEA DE/*best-rand* | 20 | -36.9286 | -36.8046 | -35.9700 | 0.2483 | 34 |
| Case 2 | | | | | | |
| AIDEA DE/*best-rand* | 20 | -36.9286 | -36.6219 | -35.4467 | 0.4694 | 11 |
| GA-MPC | - | -36.8457 | -35.03883 | -34.1076 | 0.8329 | - |
| WI-DE | - | -36.8 | -35.6 | -34.2 | 0.904 | - |

In Figure 3 the results obtained from different combinations of $N_{pop} \times n_{pop}$ are shown for the best variants of MP-AIDEA and for both Case 1 and Case 2.

For Case 1 the best results are given by MP-AIDEA 1 and MP-AIDEA 7. Changing the values of $\delta_{local}$ from 0.1 to 0.3 (Case 2) results however in a successful performance of the algorithms with adaptation of $\delta_{local}$, that is MP-AIDEA 4 and MP-AIDEA 10. This is due to the fact that for Case 1 the chosen value of $\delta_{local}$ was close to the optimal value for this problem. This is proved in Figure 4, where the process of adaptation of $\delta_{local}$ is shown for MP-AIDEA 4 using 3 populations. Arbitrary chosen values, such as $\delta_{local} = 0.3$ (Case 2) are very dissimilar from the one obtained through the adaptation process ($\delta_{local} = 0.1$).

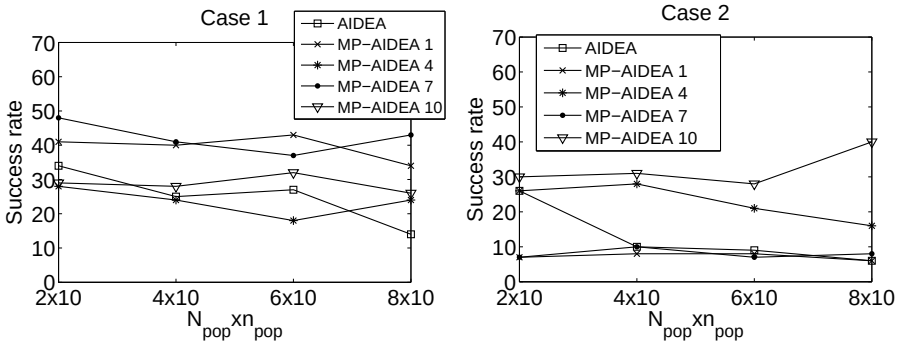



*Figure 3.* Tersoff Potential Function Minimization Problem – MP-AIDEA success rate.

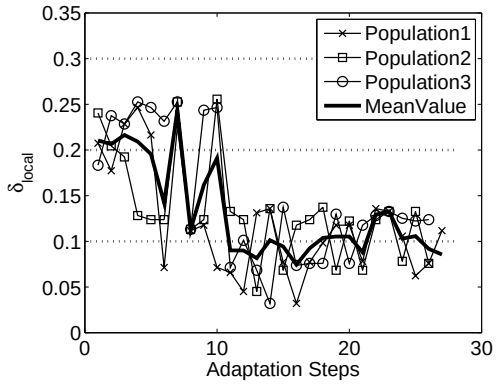



*Figure 4.* Tersoff Potential Function Minimization Problem – adaptation of $\delta_{local}$.

## 3.3 Schwefel's Problem

Schwefel's problem was tested with dimension $n_D = 30$ and $n_D = 50$. The best solution is $f_{min} = -460$; the parameters settings is $\delta_{local} = \delta_{global} = 0.1$, $\rho = 0.2$ and $iun = 5$. The maximum number of function evalutions is 3e5 for the 30D problem and 5e5 for the 50D problem, [17]. The results obtained using AIDEA with DE strategy DE/*rand/1*-DE/*best/1* are reported in Table 3 as statistics of the objective function error values with respect to $f_{min}$, as required by [17], along with the results obtained by one of the best performing algorithms of the CEC 2005 competition, the Restart Covariance Matrix Adaptation Evolution Strategy with Increasing Population Size (IPOP-CMA-ES) [2]. AIDEA gives better results for the 30D problem and for the 50D problem it locates the global minimum, while CMA-ES was not able to find it.

*Table 3.* Schwefel's Problem – AIDEA and IPOP-CMA-ES results.

| Algorithm | $n_D$ | $n_{pop}$ | Min | Mean | Max | Str.Dev. | S.Rate |
|---|---|---|---|---|---|---|---|
| AIDEA | 30 | 20 | 2.01e-9 | 1.03e+2 | 1.00e+3 | 1.97e+2 | 43 |
| IPOP-CMA-ES | 30 | - | 3.79e-9 | 4.43e+4 | 1.10e+6 | 2.19e+5 | - |
| AIDEA | 50 | 40 | 1.45e-8 | 2.63e+3 | 1.75e+4 | 2.74e+3 | 1 |
| IPOP-CMA-ES | 50 | - | 9.67e+0 | 2.27e+5 | 5.57e+6 | 1.11e+6 | - |

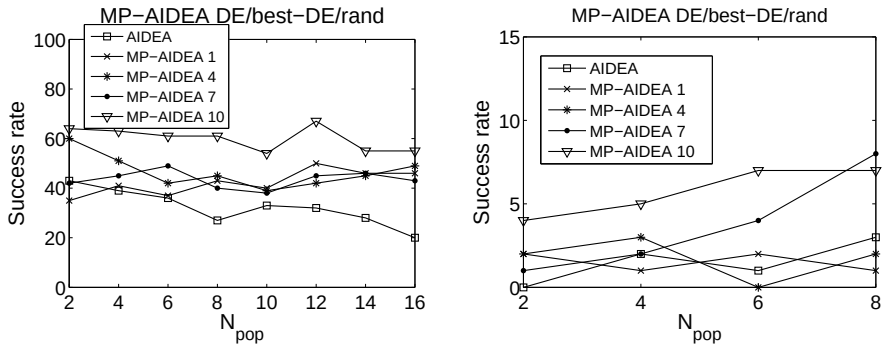



*Figure 5.* Schwefel's Problem – MP-AIDEA success rate.

In Figure 5 the success rates obtained for different values of $N_{pop}$, with $n_{pop} = 10$ for the 30D problem and $n_{pop} = 20$ for the 50D problem, are shown for the most successful versions of MP-AIDEA. For the 30D problem MP-AIDEA gives better results than AIDEA in most of the cases; for the 50D problem MP-AIDEA is able to find the global minimum of the function.

## 3.4 Rotated Version of Hybrid Composition Function

For this function the best solution is $f_{min} = 120$, $n_D = 10$, $\delta_{local} = \delta_{global} = 0.1$, $\rho = 0.2$ and $iun = 5$. The maximum number of function evaluations is 1e5 [17]. The results obtained using AIDEA, IPOP-CMA-ES and MP-AIDEA are shown in Table 4 and Table 5, where results from different combinations of $N_{pop} \times n_{pop}$ are presented.

The most successful variants of MP-AIDEA were able to locate the global minimum of this function, a minimum that neither IPOP-CMA-ES nor AIDEA were able to find.

*Table 4.* Rotated Version of Hybrid Composition Function – AIDEA and IPOP-CMA-ES results.

| Algorithm | $n_{pop}$ | Min | Mean | Max | Str.Dev. | S.Rate |
|---|---|---|---|---|---|---|
| AIDEA | 40 | 5.38e+1 | 1.02e+2 | 1.14e+2 | 8.42e+0 | 0 |
| IPOP-CMA-ES | - | 7.92e+1 | 9.13e+1 | 9.68e+1 | 3.49e+0 | - |

*Table 5.* Rotated Version of Hybrid Composition Function – MP-AIDEA success rate.

| Algorithm | 2x20 | 4x20 | 6x20 | 8x20 |
|---|---|---|---|---|
| MP-AIDEA 4 | **2** | **1** | **1** | 0 |
| MP-AIDEA 10 | **1** | 0 | **1** | **3** |

## 4. Conclusions

In this paper a multi-population version of AIDEA have been presented and tested. Results have shown that MP-AIDEA can give results which are better, or at least comparable, to the ones provided by AIDEA. The new strategies DE/*arch*/*1*-DE/*rand*/*1* and DE/*arch*/*1*-DE/*best*/*1* have shown to be effective when the number of populations is not too low. The adaptation of the bubble dimension has proven to give good results, having moreover the advantage of not requiring the setting of the parameter $\delta_{local}$ for the dimension of the bubble of the local restart. In addition, the most successful versions of MP-AIDEA were able to locate for the first time the global minima of two difficult academic functions.

## References

[1] M. M. Ali and A. Torn. Population set based global optimization algorithms: Some modifications and numerical studies. *Comput. Oper. Res.*, 31(10):1703–1725, 2004.

[2] A. Auger and N. Hansen. A Restart CMA Evolution Strategy with Increasing Population Size. *Proc. IEEE Congress on Evolutionary Computation (CEC)*, pp. 1769–1776, 2005.

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

[4] J. P. Chiou and F. S. Wang. A Hybrid Method of Differential Evolution with Application to Optimal Control Problem of a Bioprocess System. In *Proc. IEEE World Congress on Computational Intelligence (WCCI)*, pages 627–632, 1998.

[5] L. Coelho and V. C. Mariani. A Hybrid Method of Differential Evolution and SQP for Solving the Economic Dispatch Problem with Valve-Point Effect. *Applications of Soft Computing*, pages 311–320, 2006.

[6] S. Das, and P. N. Suganthan. Problem Definitions and Evaluation Criteria for CEC 2011 Competition on Testing Evolutionary Algorithms on Real World Optimization Problems. Technical Report, 2010.

[7] S. M. Elsayed, R. A. Sarker, and D. L. Essam. GA with a New Multi-Parent Crossover for Solving IEEE-CEC2011 Competition Problems. *Proc. IEEE Congress on Evolutionary Computation (CEC)*, pp. 1034–1040, 2011.

[8] R. Gamperle, S.D. Muller, and P. Koumoutsakos. A Parameter Study for Differential Evolution. In *Proc. WSEAS International Conference on Advances in Intelligent Systems, Fuzzy Systems, Evolutionary Computation*, pages 293–298, 2002.

[9] U. Halder, S. Das, D. Maity, A. Abraham, and P. Dasgupta. Self Adaptive Cluster Based and Weed Inspired Differential Evolution Algorithm for Real World Optimization. *Proc. IEEE Congress on Evolutionary Computation (CEC)*, pp. 750–756, 2011.

[10] J. Liu and J. Lampinen. A fuzzy adaptive differential evolution algorithm. *J. Soft Computing*, 9:448–462, 2005.

[11] E. Minisci and M. Vasile. Adaptive Inflationary Differential Evolution. In *Proc. IEEE World Congress on Computational Intelligence (WCCI)*, 2014.

[12] M. G. H. Omran, A. Salman, and A. P. Engelbrecht. Self-adaptive Differential Evolution. *Lect. Notes Artif. Intell.*, 3801:192–199, 2005.

[13] K. V. Price, R. M. Storn, and J. A. Lampinen. *Differential Evolution. A Practical Approach to Global Optimization. Natural Computing Series*, Springer, 2005.

[14] 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.

[15] A. Qinq. *Differential Evolution: Fundamentals and Applications in Electrical Engineering*. John Wiley & Sons, 2009.

[16] D. Shen, Y. Li, B. Wei, and X. Xia. Adaptive Forking Multipopulation Differential Evolution Algorithm for Multimodal Optimization. *J. Convergence Information Technology*, 7(5):218–227, 2011.

[17] P. N. Suganthan, N. Hansen, J. J. Liang, K. Deb, Y. P. Chen, A. Auger, and S. Tiwari. Problem Definitions and Evaluation Criteria for the CEC 2005 Special Session on Real-Parameter Optimization. Technical Report, 2005.

[18] M. Vasile, E. Minisci, and M. Locatelli. Analysis of Some Global Optimization Algorithms for Space Trajectory Design. *J. Spacecraft Rockets*, 47:334–344, 2010.

[19] M. Vasile, E. Minisci, and M. Locatelli. An Inflationary Differential Evolution Algorithm for Space Trajectory Optimization. *IEEE T. Evolut. Comput.*, 15:267–281, 2011.

[20] D.J. Wales, and J.P.K. Doye. Global Optimization by Basin-Hopping and the Lowest Energy Structures of Lennard-Jones Clusters Containing up to 110 Atoms. *J. Phys. Chem.*, 10:5111–5116, 1997.

[21] W. Yu and J. Zhang. Multi-population Differential Evolution with Adaptive Parameter Control for Global Optimization. In *Proc. 13th Annual Conference on Genetic and Evolutionary Computation (GECCO)*, pages 1093–1098, 2011.

[22] D. Zaharie. A Multipopulation Differential Evolution Algorithm for Multimodal Optimization. In *Proc. 10th International Conference on Soft Computing (MENDEL)*, pages17–22, 2004.

# AUTOMATED SLOGAN PRODUCTION USING A GENETIC ALGORITHM

Polana Tomašič
*XLAB, Pot za Brdom 100, 1000 Ljubljana, Slovenia*
*and*
*Jožef Stefan International Postgraduate School, Ljubljana, Slovenia*
polona.tomasic@xlab.si

Gregor Papa
*Computer Systems Department*
*Jožef Stefan Institute, Ljubljana, Slovenia*
*and*
*Jožef Stefan International Postgraduate School, Ljubljana, Slovenia*
gregor.papa@ijs.si

Martin Žnidaršič
*Department of Knowledge Technologies*
*Jožef Stefan Institute, Ljubljana, Slovenia*
*and*
*Jožef Stefan International Postgraduate School, Ljubljana, Slovenia*
martin.znidarsic@ijs.si

**Abstract** Invention of slogans is an intelligent and highly creative task. As such, it is a challenging problem for computational methods. In this paper we present our solution based on the use of linguistic resources and evolutionary computing.

**Keywords:** Computational creativity, Genetic algorithms, Slogan generation.

## 1. Introduction

Generation of slogans for companies and products is one of the less explored problems in the field of Computational Creativity. To our knowledge, there is only one scientific study dedicated particularly to slogan

(and other creative sentences) generation, namely the BrainSup framework [13]. This approach requires the user to provide keywords, domain, emotions and similar properties of the slogans. This shrinks the huge search space of slogans and improves the quality of results. We, however, have aimed at a completely autonomous approach that is not influenced by the user in any way, apart from providing a short textual description of the target entity.

In this paper, we present our slogan generation procedure, which is based on a genetic algorithm (GA) [1]. Genetic algorithms ensure good coverage of the search space and are relatively often used in Computational Creativity. For instance, they have been successfully used for generating recipes [11], poetry [7] and trivial dialog phrases [10]. However, genetic algorithm has not previously been used for slogan generation. Our method is the first to use it for that purpose. It follows the BrainSup framework in the initial population generation phase, and it uses a collection of heuristic slogan functions in the evaluation phase.

The results of the experiments indicate some deficiencies of our method. The generated slogans nonetheless present a good starting point for brainstorming.

## 2. Resources

Our slogan generation method requires some linguistic and semantic resources for the generation of initial population:

- **Database of the existing slogans**
  The database of existing slogans serves as a basis for the initial population generation and for comparison with generated slogans. It contains famous slogans obtained from the Internet.

- **Database of the frequent grammatical relations**
  For the acquisition of the frequent grammatical relations between words in sentences we used the Stanford Dependencies Parser [8]. Stanford dependencies are triplets containing two words, called *governor* and *dependent*, and the name of the relation between them. The parser also provides part-of-speech (POS) tags and phrase structure trees. An example of its output is in Figure 1. To get representatives of frequent grammatical relations between words, we parsed 52,829 random Wikipedia pages, sentence by sentence, and obtained 4,861,717 different dependencies.

- **Database of the slogan skeletons**
  A slogan skeleton contains information about each position in the sentence - its POS tag and all its dependencies relations with other

words in the sentence. It does not contain any content words, only stop words. An example of a skeleton from [13] is in Figure 2. Skeletons were obtained by parsing existing slogans with the Stanford Dependencies Parser.

```
Jane is walking her new dog in the park.

(ROOT
  (S
    (NP (NNP Jane))
    (VP (VBZ is)
      (VP (VBG walking)
        (NP (PRP$ her) (JJ new) (NN dog))
        (PP (IN in)
          (NP (DT the) (NN park)))))
    (. .)))

nsubj(walking-3, Jane-1)
aux(walking-3, is-2)
root(ROOT-0, walking-3)
poss(dog-6, her-4)
amod(dog-6, new-5)
dobj(walking-3, dog-6)
det(park-9, the-8)
prep_in(walking-3, park-9)
```

*Figure 1.* Stanford dependencies parser's output for the sentence "Jane is walking her new dog in the park."

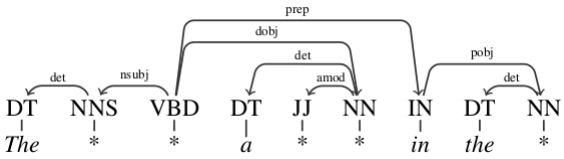



*Figure 2.* Example of a skeleton from [13].

## 3. Slogan Generation

In this section we describe our slogan generation approach in terms of its inputs, outputs and algorithmic steps. The whole procedure is shown in the Algorithm 1.

### 3.1 Extraction of the Keywords and the Main Entity

Target keywords are extracted from the input text using the Nodebox English Linguistics library [12]. The aim is to generate positive slogans.

**Algorithm 1** SloganGenerator

**Input:** A textual description of a company or a product $T$, Size of the population $S_{\mathrm{P}}$, Maximal number of iterations $Max\_iter$, Crossover probability $p_{\mathrm{crossover}}$, Mutation probability $p_{\mathrm{mutation}}$, Set of evaluation weights $W$.

**Output:** A set of generated slogans $S$.

```
1: Keywords, Entity ⇐ GetKeywordsAndEntity(T)
2: P ⇐ CreateInitialPopulation(S_IP, Keywords, Entity)
3: Evaluate(P)
4: while Max_iter > 0 do
5:     ChooseParentsForReproduction(P)
6:     Crossover(P, p_crossover)
7:     Mutation(P, p_mutation)
8:     DeleteSimilarSlogans(P)
9:     while Size(P) < S_P do
10:        AddARandomSeed(P)
11:    end while
12:    Evaluate(P)
13:    Max_iter ⇐ Max_iter − 1
14: end while
15: S ⇐ P
```

That is why all the sentences with the negative polarity in the input text are being removed. A sentence has a negative polarity if it contains words that are associated with negative emotions. After the removal, the most frequent words are selected as keywords. The main entity is usually the name of the company and is obtained by selecting the most frequent entity in the whole text using *nltk* library [2].

Example of the keywords and the entity, extracted from the Coca-Cola Wikipedia page:
keywords = ['win', 'celebrate', 'enjoy', 'follow', 'available', 'raspberry', 'snowy', 'cherry', 'famous', 'wonderful', 'familiar', 'sugar', 'sparkle', 'passion', 'beloved', 'fountain', 'bubble', 'enjoyment', 'drink', 'fluid', 'diet', 'candy', 'tour', 'beverage', 'contribution', 'dream', 'vision', ... ]
entity = Coke

## 3.2 Generation of the Initial Population of Slogans

The procedure of generating the initial population of slogans is based on the BrainSup framework [13], with some modifications. It follows the steps in Algorithm 2. Skeletons are obtained from the database of slogan skeletons, and fillers are words from the database of all grammat-

ical relations between words in sentences and must satisfy all predefined dependencies and POS tags. If there are any keywords in a set of all possible filler words, the algorithm assigns them higher priority for the selection phase. The main difference between our algorithm and the BrainSup method is in selection of filler words. We select them at random, while the BrainSup framework uses a beam search in the space of all possible lexicalizations of a skeleton to promote the words with the highest likelihood of satisfying the user specifications.

**Algorithm 2** CreateInitialPopulation

**Input:** Size of the population $S_{\mathrm{P}}$, a set of target keywords $K$, and the target entity $E$.
**Output:** A set of initial slogans $S$.

```
1: S ⇐ ∅
2: while S_IP > 0 do
3:     SloganSkeleton ⇐ SelectRandomSloganSkeleton()
4:     while not AllEmptySlotsFilled(SloganSkeleton) do
5:         EmptySlot ⇐ SelectEmptySlotInSkeleton(SloganSkeleton)
6:         Fillers ⇐ FindPossibleFillerWords(EmptySlot)
7:         FillerWord ⇐ SelectRandomFillerWord(Fillers)
8:         FillEmptySlot(SloganSkeleton, FillerWord)
9:     end while
10:    AddFilledSkeleton(S, SloganSkeleton)
11:    S_P ⇐ S_P − 1
12: end while
```

### 3.3 Evaluation of Slogans

To order the slogans by their quality, an aggregated evaluation function was constructed. It is composed of 9 different sub-functions, each assessing a particular feature of a slogan with scores in the interval [0,1]. Parameter of the aggregation function is a list of 9 weights that sum to 1. They define the proportions of sub-functions in the overall score. In this subsection we give a short description for every one of them.

**Bigram Function.** In order to work with 2-grams, we obtained the data set of 1,000,000 most frequent 2-grams and 5000 most frequent words in Corpus of Contemporary American English (COCA) [3]. We assume that slogans containing many frequent 2-grams, are more likely to be semantically coherent. That is why they get higher bigram evaluation score.

**Length Function.** This function assigns score 1 to slogans with less than 8 words, and score 0 to longer ones.

**Diversity Function.** The diversity function evaluates a slogan by counting the number of repeated words. The highest score goes to a slogan with no repeated words. If a slogan contains identical consecutive words, it receives score 0.

**Entity Function.** It returns 1, if slogan contains the main entity, and 0, if it doesn't.

**Keywords Function.** If one up to half of the words in a slogan belong to the set of keywords, the keywords function returns 1. If a slogan doesn't contain any keyword, the score is 0. If more than half of the words in the slogan are keywords, the score is 0.75.

**Word Frequency Function.** This function prefers slogans with many frequent words, because slogans with many infrequent words are considered bad. The score is obtained by dividing the number of frequent words by the number of all words in the slogan. Word is considered to be frequent, if it is among 5000 most frequent words in COCA.

**Polarity and Subjectivity Functions.** To calculate the polarity and subjectivity scores based on the adjectives in the slogan, we used the *sentiment* function from *pattern* package for Python [4].

**Semantic Relatedness Function.** This function computes the relatedness between all pairs of content words in the slogan. Stop words are not taken into account. Each pair of words gets a score based on the path distance between corresponding synsets (sets of synonyms) in WordNet [9]. The final score is the sum of all pairs' scores divided by the number of all pairs.

## 3.4 Production of a New Generation of Slogans

A list of all generated slogans is ordered descending with regard to the evaluation score.

We use a 10% elitism [5]. The other 90% of parent slogans are selected using a roulette wheel [6].

A new generation is built by pairing parents and performing the crossover function followed by the mutation function, which occur with probabilities $p_{crossover}$ and $p_{mutation}$ respectively. Offspring are then evaluated and compared to the parents, in order to remove very sim-

ilar ones. If the number of the remaining slogans is smaller than the size of the population, some additional random slogans are generated using the method for initial slogans production. After that, slogans proceed into the next generation. These steps are repeated until the predefined maximal number of iterations is achieved.

**Crossover.** There are two types of crossover function, the *big* and the *small* one. Both inspect POS tags of the words in both parents, and build a set of possible crossover locations. Each element in the set is a pair of numbers. The first one provides a position of crossover in the first parent and the second one in the second parent. The corresponding words must have the same POS tag. Let the chosen random pair from the set be $(p, r)$. Using the *big* crossover, the part of the first parent, from the $p^{th}$ position forward, is switched with the part of the second parent, from the $r^{th}$ position forward. For the *small* crossover only the $p^{th}$ word in the first parent and the $r^{th}$ word in the second parent are switched. Examples for the *big* and the *small* crossover are in Figure 3.

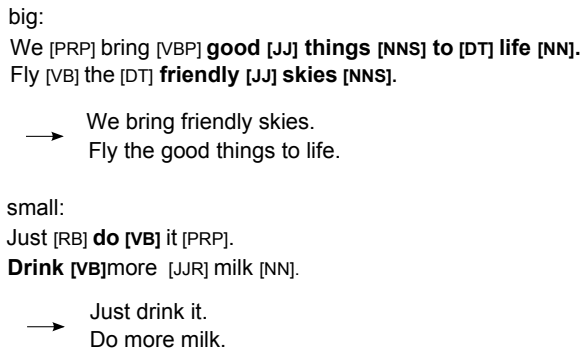



*Figure 3.* Examples for the *big* and the *small* crossover.

**Mutation.** Two types of mutation are possible. Possible *big* mutations are: deletion of a random word; addition of an adjective in front of a noun word; addition of an adverb in front of a verb word; replacement of a random word with new random word with the same POS tag. *Small* mutations are replacements of a word with its synonym, antonym, meronym, holonym, hypernym or hyponym. A meronym is a word that denotes a constituent part or a member of something. The opposite of a meronym is a holonym - the name of the whole of which the meronym is a part. A hypernym is a general word that names a broad category that includes other words, and a hyponym is a subdivision of more general word.

Functions for obtaining such replacements are embedded into the Nodebox English Linguistics library and are based on the WordNet lexical database [9].

**Deletion of similar slogans.** Every generated slogan is compared to all its siblings and to all the evaluated slogans from the previous generation. If a child is identical to any other slogan, it gets removed. If more than half of child's words are in another slogan, the two slogans are considered similar. Their evaluation scores are being compared and the one with the higher score remains while the other one is removed. The child is also removed, if it contains only one word or if it is longer than 10 words. Deletion of similar slogans prevents the generated slogans to converge to the initial ones.

## 4. Experiments

We made a preliminary assessment of the generator with experiments as described in this section.

### 4.1 Experimental Setting

In presented experiments and results we use a case of the U. S. soft drinks manufacturer Coca-Cola. The input text was obtained from Wikipedia [15].

First, we tried to find the optimal weights for the evaluation function. We tested different combinations of weights on a set of manually evaluated slogans. The comparison of the computed and the manually assigned scores showed that the highest matching was achieved with the following weights: [bigram: 0.22, length: 0.03, diversity: 0.15, entity: 0.08, keywords: 0.12, frequent words: 0.1, polarity: 0.15, subjectivity: 0.05, semantic relatedness: 0.1].

In our experiments we used probabilities for crossover and mutation $p\_crossover = 0.8$, $p\_mutation = 0.7$. The probability for mutation was set very high, because it affects only one word in a slogan. Consequently the mutated slogan is still very similar to the original one. Thus the high mutation probability does not prevent population to converge to an optimum solution. For the algorithm to decide which type of crossover to perform, we set probabilities for the *big*, the *small* and *both* crossovers to 0.4, 0.2 and 0.4 respectively. The mutation type is chosen similarly. Probabilities of the *big* and the *small* mutation were set to 0.8 and 0.2. These control parameters were set according to the results of testing on a given input text, as their combination empirically leads to convergence.

Due to the high computational complexity of our method, the maximal number of iterations was set to 150. We performed 3 experiments and for each of them we executed 20 runs of the algorithm using the same input parameter values. The only difference between these three tests was in the size of the population - 25, 50 and 75.

## 4.2 Results and Discussion

All 20 runs of the algorithm on the same input data had similar statistical results. Statistics of average slogans' scores for each of the experiments are gathered in Table 1. Slogans' scores increased with each iteration. The results in Figure 4 show that the slogans' scores increased very fast in relation to the number of evaluations when the size of the population was set to 25. They increased a bit slower when the size of the population was set to 50 and 75.

*Table 1.* Comparison of average slogans' scores for sizes of population: 25, 50 and 75. (F = final slogans, IP = initial population).

| | *Minimum* | *Maximum* | *Average* | *Median* | *Standard Deviation* |
|---|---|---|---|---|---|
| IP (25) | 0.000 | 0.720 | 0.335 | 0.442 | 0.271 |
| IP (50) | 0.000 | 0.721 | 0.318 | 0.377 | 0.270 |
| IP (75) | 0.000 | 0.736 | 0.311 | 0.412 | 0.270 |
| F (25) | 0.542 | 0.874 | 0.736 | 0.754 | 0.089 |
| F (50) | 0.524 | 0.901 | 0.768 | 0.775 | 0.082 |
| F (75) | 0.497 | 0.920 | 0.778 | 0.791 | 0.086 |

The numbers in graph show that our method ensures higher slogan scores with each new iteration of genetic algorithm, for a given experimental cases. Examples of slogans for one specific run of the algorithm are listed in the following two lists. The first list contains 10 best rated initial slogans and the second one contains 10 best rated final slogans for the case when the size of the population was set to 25. Evaluation scores are in the brackets.

**Initial population:**

1 The lucky player to sign the language in (0.714)

2 it should enjoy lead without learning line (0.706)

3 collection remains available more. fit more (0.647)

4 growing child have (0.600)

5 not a speed in a generator (0.595)

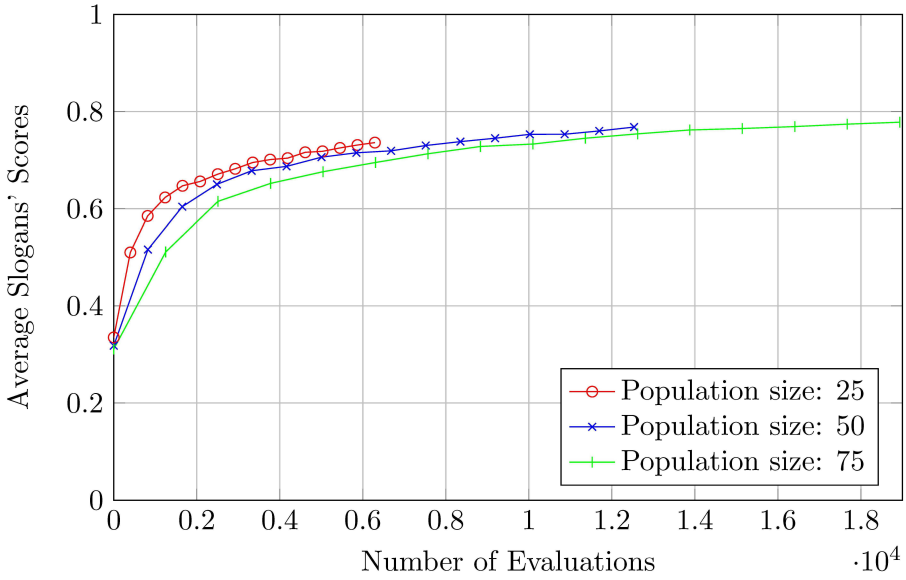



*Figure 4.* Average slogans' scores in relation to the number of evaluations.

6 Where the dream lives the environment (0.594)

7 Coke on the legal test (0.592)

8 called skip (0.560)

9 add to Coke Capazoo (0.559)

10 also stories that provide alternatives might read due as our community (0.527)

**Final slogans:**

1 love to take The Coke size (0.906)

2 rampage what we can take more (0.876)

3 love the man binds the planetary Coke (0.870)

4 devour what we will take later (0.859)

5 you can put The original Coke (0.850)

6 lease to take some original nose candy (0.848)

7 contract to feast one's eyes the na keep (0.843)

8 it ca taste some Coke in August (0.841)

9 hoy despite every available larger be farther (0.834)

10 you Can love the simple Coke (0.828)

The analysis of initial populations and final slogans in all runs of experiments shows that the majority of slogans are semantically incoherent and have grammatical errors.

Our system currently lacks an evaluation function for detection or correction of these mistakes.

Some seemingly good slogans can be found already in the initial populations. The evaluation function seems not yet aligned well with human evaluation, as such slogans often do not make it to the final round.

## 5. Conclusions

The proposed slogan generation method works and could be potentially useful for brainstorming. The genetic algorithm ensures that new generations of slogan candidates have higher evaluation scores. The critical part of the method is the evaluation function, which is inherently hard to formalize and needs further improvement. The definitions of evaluation sub-functions are currently too simplified. We believe that the refinement of semantic and sentiment evaluation functions would increase the quality of slogans, not only their scores.

The current algorithm is suitable only for production of slogans in English, because there is a wide range of lexical and semantic resources for it. There is a possibility of generating slogans in a language with different characteristics, for instance Slovenian. However, the lack of resources and different language properties would require a lot of work in order to adapt our algorithm for a non-English language.

There are also many other ideas for the future work that would improve the quality of slogans. One is checking for grammatical errors and correcting them if possible. New weights for the evaluation could be computed periodically with semi-supervised learning on manually assessed slogans. Also, control parameters for GA could be adaptively calculated during the optimization process [14].

## Acknowledgment

This research was partly funded by the European Union, European Social Found, in the framework of the Operational Programme for Human Resources Development, by the Slovene Research Agency and supported through EC funding for the project ConCreTe (grant number 611733) and project WHIM (grant number 611560) that acknowledge

the financial support of the Future and Emerging Technologies (FET) programme within the Seventh Framework Programme for Research of the European Commission.

## References

[1] T. Bäck. *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms*. Oxford University Press, 1996.

[2] S. Bird, E. Klein, and E. Loper. *Natural language processing with Python*. O'Reilly Media, 2009. `http://www.nltk.org/`

[3] M. Davies. N-grams data from the Corpus of Contemporary American English (COCA). Downloaded from `http://www.ngrams.info` on April 15, 2014.

[4] T. De Smedt and W. Daelemans. Pattern for Python. *J. Mach. Learn. Res.*, 13:2063–2067, 2012.

[5] D. Dumitrescu, B. Lazzerini, L. C. Jain, and A. Dumitrescu. *Evolutionary Computation*. CRC Press, 2000.

[6] J. H. Holland. *Adaption in Natural and Artificial Systems*. MIT Press, 1992.

[7] R. Manurung, G. Ritchie, and H. Thompson. Using genetic algorithms to create meaningful poetic text. *J. Exp. Theor. Artif. In.*, 24:43–64, 2012.

[8] M. Marneffe, B. MacCartney, and C. Manning. Generating typed dependency parses from phrase structure parses. In *Proc. 5th International Conference on Language Resources and Evaluation (LREC)*, pages 449–454, 2006.

[9] G. A. Miller. WordNet: A Lexical Database for English. *Comm. ACM*, 38:39–41, 1995.

[10] C. S. Montero and K. Araki. Is it correct?: towards web-based evaluation of automatic natural language phrase generation. In *Proc. Joint Conference of the International Committee on Computational Linguistics and the Association for Computational Linguistics (COLING/ACL)*, pages 5–8, 2006.

[11] R. G. Morris and S. H. Burton. Soup over bean of pure joy: Culinary ruminations of an artifcial chef. In *Proc. 1st IEEE International Conference on Communication in China (ICCC)*, pages 119–125, 2012.

[12] Nodebox. `http://nodebox.net/code/index.php/Linguistics`

[13] G. Özbal, D. Pighin, and C. Strapparava. BRAINSUP: Brainstorming Support for Creative Sentence Generation. In *Proc. 51st Annual Meeting of the Association for Computational Linguistics*, pages 1446–1455, 2013.

[14] G. Papa. Parameter-less algorithm for evolutionary-based optimization. *Comput. Optim. Appl.*, 56:209–229, 2013.

[15] Wikipedia. `http://en.wikipedia.org/wiki/Coca-Cola` on April 29, 2014.

# A COMPARISON OF SEARCH SPACES AND EVOLUTIONARY OPERATORS IN FACIAL COMPOSITE CONSTRUCTION

Joseph James Mist
*School of Physical Sciencese*
*University of Kent, Canterbury, United Kingdom*
jm441@kent.ac.uk

Stuart James Gibson
*School of Physical Sciencese*
*University of Kent, Canterbury, United Kingdom*
s.j.gibson@kent.ac.uk

Christopher John Solomon
*School of Physical Sciencese*
*University of Kent, Canterbury, United Kingdom*
c.j.solomon@kent.ac.uk

**Abstract** In this paper three experiments concerning the use of interactive evolutionary algorithms in the creation of facial composites are reported. A reduced dimension human evaluation based search space is created from a larger search space using a pairwise face comparison task. The human reduced search space is used in the comparison of two mutation operators and two recombination operators. Finally, three search spaces are compared: large, human reduced, and a mathematically reduced search space. No statistically significant differences are found between the performances of the operators or the search spaces.

**Keywords:** Facial composites, Interactive evolutionary algorithm.

## 1. Introduction

An unknown perpetrator may be seen committing a crime by one or more people. In these circumstances it is often useful to create a pictorial

likeness of the perpetrator's face based on an eyewitness's description. Such a likeness, known as a *facial composite*, may subsequently be used in a criminal investigation for the purpose of locating a suspect. The traditional method for facial composite construction involves a witness selecting individual facial features from a catalogue or database. A composite operator then assembles the selected features to form a face image. However, psychological research has shown that humans recognise faces not by their individual components but as whole objects [2, 8]. It is also known that people are better able to recognise faces than they can recall and describe them. Accordingly, a holistic method for facial composite construction has been developed that uses the cognitive Gestalt processes involved in face recognition. Two commercial systems based on these principles were developed in the early 2000s; EvoFIT [4] and EFIT-V (originally called EigenFIT) [5]. EFIT-V is now used frequently by the majority of police constabularies in the UK and by law enforcement agencies in many other countries.

The holistic approach requires a multidimensional search space or, more appropriately, a *face-space* in which an approximate likeness to any face can be represented by single point. The location of this point with respect to the origin of the face-space may be encoded as a string of coordinates. This is achieved through the use of a *face model*.

Here, and in our related previous work, we use a face model [1] that is constructed by processing a set of training images and determining their principal components (PCs) (see Section 2.1). The PCs are typically ordered according to the amount of image variance in the training set they explain. Hence, the first component is numerically more important than the second component which is more important than the third etc. Due to the imperfect nature of human face recognition, it is very likely that the required face-space can be constructed using a relatively small number of PCs and the other PCs can be discarded with no perceptible deterioration in performance. A distinction should also be made between an ordering of PCs according to numerical importance and an ordering according to perceptual importance. Using human evaluation to select $n$ PCs may provide an $n$-dimensional face-space which accounts for more perceptual variation than simply using the first $n$ PCs returned by a principal components analysis as calculated by computer software. This idea is investigated in Section 4.

In simple terms, a facial composite is constructed by searching the face-space for an acceptable facial likeness. The search is achieved by an iterative process whereby faces, generated by the model, are assessed by the witness according to their similarity to the perpetrator. This itera-

tive process immediately suggests the use of an *interactive evolutionary algorithm* (IEA).

IEAs differ from evolutionary algorithms (EAs) in one major respect: human evaluation replaces the fitness function. The use of human evaluation places a number of limitations on the use of IEAs which are generally not present in EAs. The most obvious two effects are that the total number of fitness evaluations that can be performed by the user and the number of possible fitness values it is feasible to assign to an individual are both limited [9]. The IEA used in the second and third experiments of this paper is detailed in Section 2.2.

Intuitively, selection of an appropriate IEA, associated operators, and the values of any associated parameters may have an effect on the composite creation process. Real valued interactive genetic algorithms (IGAs) are used in both EvoFIT and EFIT-V. Design decisions such as the population size, the mutation rates, and the use of elitism were made with the aid of virtual users that attempt to model how human users evaluate individuals in a generation. Very little work has been done to compare the performances of different IEAs for use in the creation of facial composites. A series of small experiments evaluating the performances of various nature-inspired metaheuristic algorithms have been conducted [6, 7]. The results indicate that the choice of algorithm has some effect on the recognition rates of the composites.

The focus of this paper is the comparison of search spaces and evolutionary operators in facial composite construction. In the first of three experiments reported in this paper, a 12-dimensional 'human reduced' face-space is constructed using human evaluation of the differences between pairs of faces from the 'large' 30-dimensional face-space. The second experiment compares the performances of two different mutation operators and two different recombination operators using a task which requires participants to create composites from memory. In the third experiment a 'mathematically reduced' face-space, in which only the first 12 PCs are used, is constructed. The performances of searches using the large, the human reduced and the mathematically reduced face-spaces are compared using the same composite creation task.

## 2. Theory

### 2.1 The Face Model

The set of photographs used in the training set to build the face models used in the experiments reported in this paper is composed of 27 males and 63 females of various ages. A number of points common to all of the photographs are landmarked. These common points are facial features

such as the corners of the eyes, the bottom of the chin, and the outline of the eyebrows. The set of landmarks on a particular face form a face shape. Thus, there is one face shape for each face in the training set. Each face shape consists of 190 two-dimensional landmarks and thus the resulting shape model has 380 dimensions.

The mean face shape $\bar{\mathbf{s}}$ is found by aligning the face shapes using an iterative Procrastes alignment process. Principal components analysis (PCA) is used to reduce the 380-dimensional shape model to a smaller number of dimensions. Any face shape $\mathbf{s}$ can be approximated to $\widehat{\mathbf{s}}$ in the shape model using

$$\widehat{\mathbf{s}} = \mathbf{P}_s \mathbf{b}_s + \bar{\mathbf{s}} \tag{1}$$

where $\mathbf{P}_s$ are the PCs of the shape model ordered from most important (the PCs which account for the most variance in the data) to least important and $\mathbf{b}_s$ are parameters that determine how the shape PCs are combined to make the face shape.

In order to create the texture model, each photograph in the training set is partitioned using its landmark points and Delaunay triangulation. Piecewise affine transforms are used to map the texture information (the pixel values of the photographs in the training set) from each training photograph's face shape to the mean face shape to form normalised texture patterns. PCA is then used to find a texture model with fewer dimensions than that formed by the tens of thousands of pixels within each normalised texture pattern. As with the face shapes, any face texture $\mathbf{g}$ may be approximated using

$$\widehat{\mathbf{g}} = \mathbf{P}_g \mathbf{b}_g + \bar{\mathbf{g}}. \tag{2}$$

where $\mathbf{P}_g$ are the PCs of the face texture ordered from the most important to least important and $\mathbf{b}_s$ are parameters that determine how the texture PCs are combined to make the face texture. Finally, a face-model is created from the combined shape and texture models using PCA to further reduce the number of dimensions in the final face-space. Thus, the appearance model parameters, $\mathbf{c}$, of any face can be approximated to $\widehat{\mathbf{c}}$ using

$$\widehat{\mathbf{c}} = \mathbf{Q}^T \begin{bmatrix} w\mathbf{b}_s \\ \mathbf{b}_g \end{bmatrix} \equiv \mathbf{Q}^T \begin{bmatrix} w\mathbf{P}_s^T (\widehat{\mathbf{s}} - \bar{\mathbf{s}}) \\ \mathbf{P}_g^T (\widehat{\mathbf{g}} - \bar{\mathbf{g}}) \end{bmatrix} \tag{3}$$

where $\mathbf{Q}$ are the appearance PCs of the training set ordered from the most important to the least important and $w$ is a weighting scale that scales the shape parameters such that equal significance is assigned to shape and texture.

New faces can be created by setting the values of an $n$-dimensional parameter vector $\mathbf{c}$ and performing the above process in reverse. Starting

with the extraction of $\mathbf{b}$

$$\mathbf{b} = \sum_{i=1}^{n} \mathbf{q_i} c_i \tag{4}$$

where $\mathbf{q}_i$ is the $i$-th column of matrix $\mathbf{Q}$ in Equation 3. The shape and texture parameters $\mathbf{b}_s$ and $\mathbf{b}_g$ are extracted from $\mathbf{b}$ and are used in Equations 1 and 2 to find the shape parameters $\mathbf{s}$ and texture parameters $\mathbf{g}$. The pixel intensities in $\mathbf{g}$ are rearranged into a two-dimensional (or three-dimensional for colour images) array of pixels which then form an intermediate face image with mean face shape. Aspects of the edge of the face image which were due to the landmarking process had a dominant unwarranted effect on the perception of the face. To counter this effect the generated face texture was inserted and blended into a softened background. The resulting image was subsequently warped according to the shape parameters, $\mathbf{s}$, to form the final face image.

## 2.2 The IEA Used

The IEA used is a simple real valued IGA which we refer to as the *simple IGA*. The representation used is an $n$-dimensional real valued vector where $n$ is the number of dimensions in the face-space.

In the simple IGA each individual in the following generation has two parents and each pair of parents produces only one child. Eight new individuals are needed to fill the following generation (as the best individual from the previous generation is carried through to the following generation). Thus, a mating pool of sixteen parents is required.

Stochastic universal sampling is used to select parents to go into the mating pool. The simple IGA follows Frowd's method [3] and allows only three levels of selection: preferred (best), selected, and not selected. When building the sampling wheel it was decided that all of the selected individuals are assigned equal sized wedges except the preferred individual which is assigned a double sized wedge.

Two recombination methods are used in the experiments reported in this paper: uniform crossover and arithmetic crossover, and two mutation methods are used: nonuniform mutation and Gaussian replacement.

In the implementation of uniform crossover used each gene's value in an offspring has an equal chance of coming from either parent. In the implementation of arithmetic crossover used each gene's value in an offspring is the mean of the values for that gene in the parents.

In the nonuniform mutation used in this paper the mutated gene value $c_i'$ is given by $c_i' = c_i + \sigma_i \cdot m \cdot N(0,1)$ where $\sigma_i$ is the standard deviation (SD) of the data on the $i$-th PC, $m$ is the mutation factor set by the user on the interface, and $N(0,1)$ is a random number from the Gaussian

distribution. Gaussian replacement is the name given in this paper to an analogous method to the uniform mutation operator. In uniform mutation, there is some probability $p_m$ for each gene in an offspring's genotype that its value will be replaced by a uniformly distributed random value where $c_i, c_i' \in [\text{Lower limit}, \text{Upper limit}]$. The Gaussian replacement operator is similar except that $c_i'$ is a random number taken from $N(0,1)$ and multiplied by the SD of the data on the $i$-th PC. $c_i'$ has the further restriction that it is bounded by the hyperrectangle which designates the edge of the face space, that is $c_i, c_i' \in [-2.5, 2.5]$ SDs. The mutation probability is set by the mutation slider and is restricted to the range $[0, p_{\max}]$ where $p_{\max} = 5/$ (the dimensionality of the face space).

The population size is limited by three factors: the number of images that can be displayed on the screen simultaneously, the time required to create each image, and the cognitive burden placed on the user when comparing the images. After considering these three factors a population size of nine was chosen.

## 3. The User Interface for Experiments 2 and 3

A screenshot of the interface developed for the experiment is given in Figure 3. Every generation the participants would choose a preferred composite that best resembled the face they were trying to recreate and select it using the left mouse button. The participants also had the option of selecting any composites that they thought were also good by selecting them using the right mouse button. Anywhere from zero to eight composites could be selected with the right mouse button. A green border was placed around the composite the participant preferred, a yellow border for those composites the participant thought were also good, and a black border for those composites that were not selected. Once they were satisfied that they had selected the best match and any other matches they considered to be good, the participant would go to the next generation by pressing the 'Next' button. The preferred composite was carried forward into the next generation. The participants would continue the process until they thought they had successfully recreated the target face, or until they thought no further improvement was possible, by clicking on the 'Finish' button. It was observed during previous (unpublished) experiments that using a self adaptive step size in an IEA often resulted in the algorithm becoming stuck at a suboptimal solution. This problem was addressed by adding a slider which enabled the user to set the mutation value manually. However, many participants did not adjust the mutation slider. Thus, the mutation slider was decremented by 0.03 per generation by the software (the slider's range was $[0, 1]$). A

'back' button was included which enabled the participant to go back to the previous generation and make alternative selections or adjust the mutation slider if they were not satisfied with the current generation.

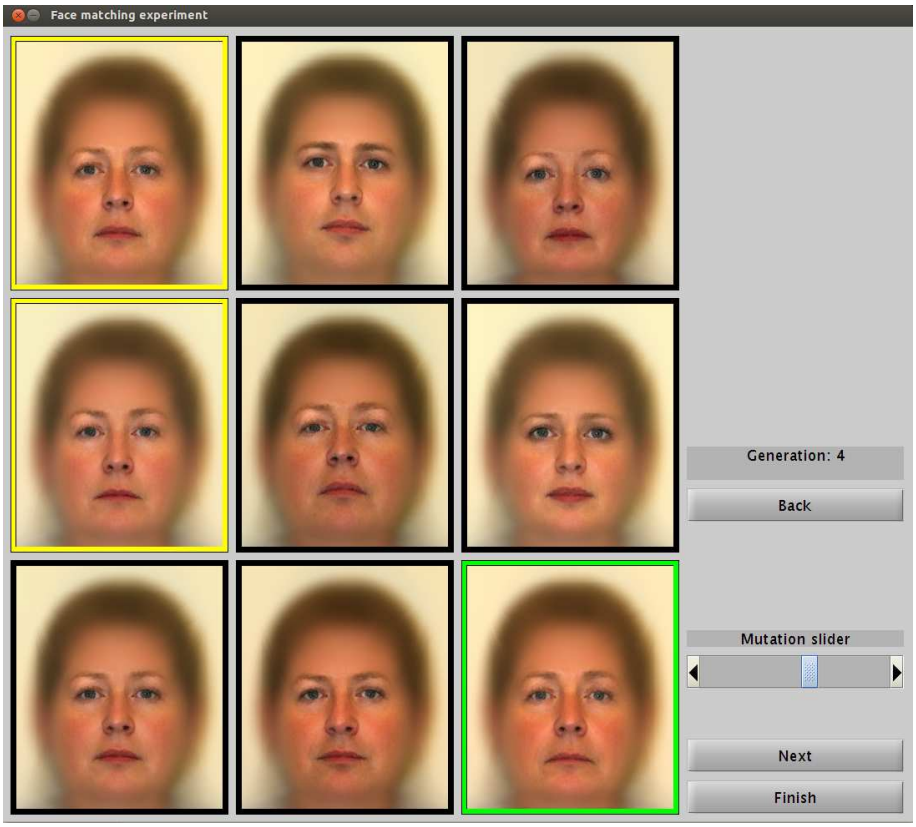



*Figure 1.* Screenshot of the interface for the facial composite tasks.

## 4. Experiment 1: Identifying the Most Perceptually Significant PCs

### 4.1 Method

In this experiment 32 participants were used to determine which 12 of the 30 PCs of the 30-dimensional (large) face-space are perceptually the most significant. Thirty pairs of faces were generated from the large PCA face-space. If a face's representation in the large face-space is given by the point $\mathbf{c} = [c_1, c_2, \ldots, c_i, \ldots, c_{30}]$, then each pair of points

$(\mathbf{c}_{+k}, \mathbf{c}_{-k})$ representing a pair of faces has coordinates defined by

$$c_{\pm i} = \begin{cases} \pm 3 & \text{SDs} \quad \text{if } i = k \\ 0 & \text{SDs} \quad \text{otherwise} \end{cases} \tag{5}$$

The faces were printed in their respective pairs on matt photographic paper. Each pair was 5.8 cm high by 10.2 cm wide. Thirty was the number of pairs chosen because: thirty pairs of faces could fit comfortably on a desk top, the difference between a pair faces becomes more difficult to discern for the higher-dimensional PCs, and the more pairs a participant has to sort through the harder the task becomes. This decision also defined the number of dimensions that were used in the large face-space.

At the start of the experiment the pairs of faces were arranged randomly in a grid six pairs high by five pairs wide. The participants were instructed to group the 12 pairs of faces which 'exhibited the most within pair dissimilarity'. Once the participants had done this they were instructed to sort the 12 pairs of faces from the most similar to the least similar. In preliminary testing, it was observed that the degree of dissimilarity between pairs of faces became very hard to discern beyond the 12 or so most dissimilar pairs. Accordingly, 12 dimensions were used for the small search space.

### 4.2 Results

The 12 most significant PCs perceptually were found to be 1, 2, 3, 4, 5, 6, 7, 9, 13, 14, 15, and 18. Thus, these are the PCs that were used to build the human reduced face-space. It can be seen that 8 of the 12 PCs in the human reduced face-space are in the first 12 PCs of the large PCA face-space.

## 5. Experiment 2: Comparison of Recombination and Mutation Operators

### 5.1 Method

In this experiment 15 participants were used to compare two recombination operators (uniform crossover and arithmetic crossover) and two mutation operators (Gaussian replacement and nonuniform mutation).

The 12-dimensional human reduced face-space was used in this experiment. This face-space was chosen because it was not thought that the face-space used would have an effect on the relative performances of the different recombination and mutation operators. However, it was thought that a lower-dimensional face-space may lead to a face match more quickly and thus induce less fatigue in the participants.

Testing each combination of recombination and mutation operator required $2 \times 2 = 4$ runs per participant. Each participant also did a practice run at the start of the experiment.

The initial population was the same for every run of the experiment and was designed to be roughly evenly distributed in the human reduced face-space. To generate the initial population, 1000 points were generated using a 12-dimensional uniform distribution with the limits being at $\pm 2.5$ SDs on each axis. Matlab's *kmeans* function was used to group the points into nine clusters. The centroids of the nine clusters were used as the genotypes for the initial population of faces.

At the start of each run the participants were given 10 seconds to study the target face which they then had to try to recreate from memory using the interactive evolutionary facial composite process. The target face was not shown to the participants again until the end of the run.

The target faces were chosen to be equidistant from the centre of the human reduced face-space.

At the end of every run, the participants were shown the composite they had just created and were asked to rate its similarity to the target on a scale from 1 to 10. Immediately after rating their composite the participants were shown the target face alongside their composite and asked to rate the similarity between their composite and the target again.

Three sets of objective data were gathered: the time taken to create the composites, the number of generations it took to create the composites, and the number of times the back button was used. If the back button was used more often for a particular operator, it indicates that the operator is not particularly suitable for the task.

## 5.2 Results

The means and standard deviations of the measure variables (number of generations, time taken, number of times the back button was used, participant rating of their composite without reference to the target, participant rating of the their composite with reference to the target) are given in Table 1. Each of the measure variables were subjected to two-way ANOVA having two mutation operators (nonuniform mutation and Gaussian replacement) and two recombination operators (uniform crossover and arithmetic crossover) (Table 2). It can be seen that the main effects of mutation operator and recombination operator were not significant for any of the measure variables, nor was the interaction of the two operators significant.

*Table 1.* Means (standard deviations) of the dependent variables in the comparison of mutation and recombination operators in the creation of facial composites.

| Mutation | Recombination | Generations | Back count | Time taken | Without rating | With rating |
|---|---|---|---|---|---|---|
| Gaussian replacement | uniform | 10.6 (5.10) | 0.73 (1.33) | 195s (91.5s) | 6.27 (1.22) | 4.40 (2.10) |
| Gaussian replacement | arithmetic | 12.5 (8.64) | 0.47 (0.74) | 222s (155s) | 5.47 (2.00) | 5.07 (2.19) |
| Nonuniform mutation | uniform | 11.5 (4.73) | 0.87 (1.41) | 220s (71.1s) | 6.07 (1.03) | 4.60 (2.41) |
| Nonuniform mutation | arithmetic | 9.73 (2.49) | 0.47 (0.64) | 188s (66.2s) | 6.07 (1.49) | 4.40 (2.32) |

*Table 2.* Two-way ANOVA of the dependent variables in the comparison of mutation and recombination operators in the creation of facial composites.

| | Mutation | | Recombination | | Interaction | |
|---|---|---|---|---|---|---|
| Variable | $F(1, 56)$ | p-value | $F(1, 56)$ | p-value | $F(1, 56)$ | p-value |
| Generations | 0.43 | 0.513 | 0.00 | 0.946 | 1.56 | 0.217 |
| Back Count | 0.06 | 0.813 | 1.41 | 0.240 | 0.06 | 0.813 |
| Time taken | 0.03 | 0.874 | 0.01 | 0.904 | 1.21 | 0.275 |
| Without comparison rating | 0.27 | 0.603 | 1.10 | 0.300 | 1.10 | 0.300 |
| With comparison rating | 0.16 | 0.691 | 0.16 | 0.691 | 0.55 | 0.461 |

*Table 3.* Means (standard deviations) of the dependent variables in the comparison of the large, human reduced and mathematically reduced face-spaces in the creation of facial composites.

| Face-space | Gens | Back count | Time taken | Without rating | With rating |
|---|---|---|---|---|---|
| Large | 10.7 (4.73) | 0.50 (0.55) | 205s (80.3s) | 5.81 (1.13) | 4.10 (1.25) |
| Human reduced | 9.38 (4.31) | 0.36 (0.42) | 186s (91.8s) | 6.02 (1.08) | 3.95 (1.33) |
| Mathematically reduced | 10.5 (4.75) | 0.48 (0.56) | 193s (85.6s) | 5.86 (1.16) | 4.12 (1.82) |

## 6. Experiment 3: Comparison of Face-spaces

### 6.1 Method

In this experiment 21 participants were used to compare three face-spaces: a face-space constructed from the first 30 PCs of the PCA analysis (the large face-space), a face-space constructed from the first 12 PCs (the mathematically reduced face-space), and a face-space constructed form the 12 most perceptually important PCs identified in the first experiment (the human reduced face-space).

As the results of the second experiment showed no significant difference between the operators on any of the recorded measures, arithmetic crossover and nonuniform mutation were arbitrarily chosen as the operators used for this experiment.

As there were only three test conditions (large face-space, human reduced face-space, and mathematically reduced face-space) each participant performed two runs for each of the test conditions so that they performed $2 \times 3 = 6$ runs.

The initial populations for each of the face-spaces were constructed in the same way as that for the second experiment. The target faces were chosen to be equidistant from the centre of the 30-dimensional face-space.

### 6.2 Results

The means and standard deviations of the measure variables over all of the runs for each of the algorithms are presented in Table 3.

Performing one-way ANOVA on each of the measure variables (averaged over both runs for each of the test conditions) showed that the differences between the face-spaces were not significant for any of the measure variables (number of generations: $F(2,60) = 0.51, p = 0.604$, number of times the 'back' button was used: $F(2,60) = 0.47, p = 0.629$, time taken: $F(2,60) = 0.28, p = 0.758$, without comparison rating: $F(2,60) = 0.21, p = 0.811$, and with comparison rating: $F(2,60) = 0.08, p = 0.926$).

## 7. Conclusion

A human evaluation based reduced face-space for use with an IEA in the creation of facial composites was derived from a larger PCA based face-space. The performances of searches for faces in the human reduced face-space was compared to those of a mathematically reduced face-space and the larger face-space. The human reduced face-space was also used in the comparison between different mutation and recombination operators in the simple IGA.

The prioritisation of the PCs with regards to human evaluation was found to be similar to the numerical ordering of returned by principal component analysis itself. The human reduced face-space was found to share eight of its 12 PCs with the mathematically reduced face-space. We note that our data set comprised images captured under conditions of controlled pose, lighting and facial expression. If this were not the case, one might expect greater differences between the perceptual and numerical orderings of PCs.

No significant differences in the performances of the searches conducted using the different operators were detected. The difficulty and uncertain nature of creating a facial composite render any differences in the performances of the operators or the face-spaces insignificant.

Likewise, no significant differences in the performances of the searches conducted in the different face-spaces was observed. The implication of this is that it is possible to reduce the dimensionality of the face model without any loss of performance.

## References

[1] T. F. Cootes, G. J. Edwards, and C. J. Taylor. Active appearance models. In *Proc. European Conference on Computer Vision (ECCV)*, volume 2, pages 484–498, 1998.

[2] G. Davies and D. Christie. Face recall: An examination of some factors limiting composite production accuracy. *J. Appl. Psychol.*, 67(1):103, 1982.

[3] C. D. Frowd. *EvoFIT: A Holistic, Evolutionary Facial Imaging System.* PhD thesis, Department of Psychology, University of Stirling, 2001.

[4] C. D. Frowd, P. J. B. Hancock, and D. Carson. Evofit: A holistic, evolutionary facial imaging technique for creating composites. *ACM Trans. Appl. Percept.*, 1(1):19–39, 2004.

[5] S. J. Gibson, C. J. Solomon, and A. Pallares Bejarano. Synthesis of photographic quality facial composites using evolutionary algorithms. In *Proc. Britsh Machine Vision Conference*, volume 1, pages 221–230, 2003.

[6] B. Kurt, A. S. Etaner-Uyar, T. Akbal, N. Demir, A. E. Kanlikilicer, M. C. Kus, and F. H. Ulu. Active appearance model-based facial composite generation with interactive nature inspired heuristics. *Lect. Notes Comput. Sc.*, 4105:183–190, 2006.

[7] C. J. Solomon, S. J. Gibson, and J. J. Mist. Interactive evolutionary generation of facial composites for locating suspects in criminal investigations. *Appl. Soft Comput.*, 13(7):3298–3306, 2013.

[8] J. W. Tanaka and M. J. Farah. Parts and wholes in face recognition. *Q. J. Exp. Psychol.*, 46A:225–245, 1993.

[9] D.-M. Yoon and K.-J. Kim. Comparison of scoring methods for interactive evolutionary computation based image retouching system. In *Proc. 14th Annual Conference Companion on Genetic and Evolutionary Computation (GECCO)*, pages 617–618, 2012.

# LOCAL SEARCH BASED OPTIMIZATION OF A SPATIAL LIGHT DISTRIBUTION MODEL

David Kaljun
*Faculty of Mechanical Engineering*
*University of Ljubljana, Slovenia*
david.kaljun@fs.uni-lj.si

Janez Žerovnik
*Faculty of Mechanical Engineering*
*University of Ljubljana, Slovenia*
*and*
*Institute of Mathematics, Physics and Mechanics*
*Ljubljana, Slovenia*
janez.zerovnik@fs.uni-lj.si

**Abstract** Recent development of LED technology enabled production of lighting systems with nearly arbitrary light distributions. A nontrivial engineering task is to design a lighting system or a combination of luminaries for a given target light distribution. Here we use heuristics for solving the problem restricted to symmetrical distributions. A genetic algorithm and several versions of local search heuristics are compared showing that practically useful approximations can be achieved with majority of the algorithms.

**Keywords:** Genetic algorithm, Light distribution model, Local search, Iterative improvement.

## 1. Introduction

Nowadays, technology of Light Emitting Diodes (LEDs) enables to lower the energy consumption of luminaires and to design more efficient lighting systems that make it possible to deliver the light to the environment in a controlled way. The many possible designs lead to new problems of choosing the optimal or at least a very good design depending

on possibly different goals such as optimization of energy consumption, production cost, and, last but not least, the light pollution of the environment. Nowadays, in many cases trial and error method followed by simulation is used in practice. We believe that using analytical models and optimization tools may speed up the design and at the same time possibly improve the quality of solutions. Here we adopt an analytical model for a version of the general problem, and use heuristic methods based on the model to provide nearly optimal solutions. The heuristics used in this study are three versions of local search and a genetic algorithm. We also compute solutions provided by blind random search to avoid trivialities. The rest of the paper is organized as follows. In the next section we briefly explain the practical motivation for this research. Section 3 gives the analytical model and the optimization problem that is addressed in the paper. In Section 4, the algorithms are outlined. Experimental results are presented in Section 5. The paper ends with a summary of conclusions and idea for future work.

## 2. Motivation

Only a few years ago, emerging new technology of Light Emitting Diodes (LEDs) was in the first stage of implementation [1]. Meanwhile the demand to lower the energy consumption of luminaires and to build more efficient lighting systems that can deliver the light where needed has pushed the development of high power LEDs. Following the development of LEDs, many luminaire manufacturers developed LED luminaires as a replacement for the existing energy inefficient luminaires. Naturally, the use of LEDs introduces new and unique challenges to the development engineers. One of these challenges is to design and simulate an efficient light engine for the luminaire. The light engine consist of the source which in this case are LEDs, and the appropriate secondary optics. The choice of the secondary optics is the key in developing a good system. For designing a good system nowadays technology enables two options. Having the know-how and the resources, a specific lens to accomplish the task can be developed. However, the resources coupled with the development and production of optical elements may be enormous. Therefore a lot of manufactures are using the second option, that is to use readymade of the shelf lenses. There are specialized companies in the world that produce different type of lenses for all of the major brands of LEDs. The trick here is to choose the best combination of lenses to get the most efficient system. The current practice in development process is a trial and error procedure, where the developer chooses a combination of lenses, and then simulates the system via Monte Carlo

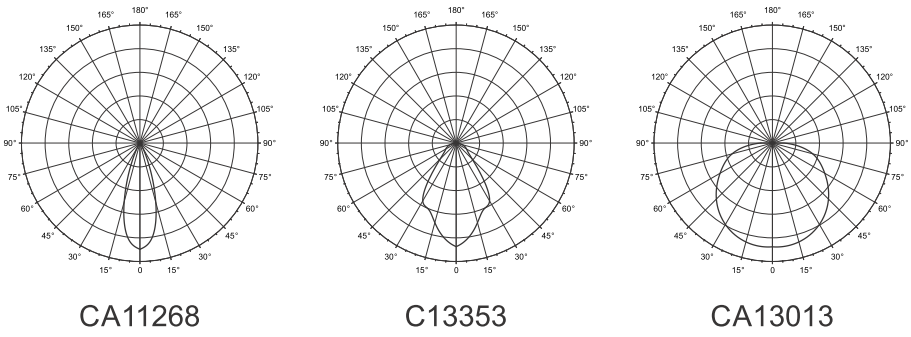



*Figure 1.* Modeled spatial light distribution presented in polar diagrams.

ray-tracing methods. The success heavily depends on the engineers' intuition and experience and still needs sizeable computation resources for checking the proposed design by simulation.

A natural avenue of research related to the second approach is to replace trial and error method by a more efficient design method based on analytical and algorithmic tools. For these aim, a theoretical framework is needed. Among the first known theoretical results is the analytical model [8] that was proposed for LEDs without secondary optics. Namely, the usual practical situation is that we have the target light distribution given by large dataset of points in the space with (desired or measured) light intensity. The idea of [8] that is at the same time already a part way towards the solution is to fit the data with suitable functions that in turn can provide a construction of a light engine which approximates the target light distribution. Later we will explain a modification of the analytical model of [8] that we use in our study where we successfully approximate symmetric spatial distributions. The general problem is much more challenging. Development of a useful analytical model for general case is to the best of our knowledge an open research problem. Having the target light distribution in a manageable form, the design of the light engine may be possible. Depending on the application, several questions/tasks are natural, for example: (1) design the engine having exactly (or, approximately) the target light distribution (2) design such engine using as few LEDs as possible (3) design such engine as cheap as possible. A combination of these goals may be of interest which in turn leads to a number of multicriteria optimization problem(s).

## 3. Analytical Model and Problem Definition

The fact that there are many different LEDs with different beam patterns and many different secondary optics to choose from indicates that

providing a general analytical model for all of them is presumably a very challenging open research problem. Therefore in this study we restrict attention only to LED-lens combinations that have symmetrical spatial light distribution. In other words, the cross section of the surface which represents the spatial distribution with a section plain that is coincident with the vertical axis of the given coordinate system is alike at every azimuthal angle of offset. This enables us to define the analytical model in two dimensions, so it describes a curve rather a surface. To produce the desired surface, we just revolve the given curve around the central vertical axis with the full azimuthal angle of 360°. Three examples from our dataset are given in Figure 1. For the special case of symmetrical spatial light distribution, an analytic model for the radiation pattern of a single LED without the secondary optics was proposed in [8]. The author proposed two different models, one based on Gaussian and one on cosine-power function. Based on our preliminary manual test fittings of the models to measured data of three randomly chosen lenses from the dataset [5] we use here we have concluded that the cosine-power functions $I(\Theta) = a * \cos(\Theta - b)^c$ have slight advantage over the Gaussian ones. Another argument is that the cosine-power functions seem to be a more natural choice in this context because there are basic LEDs with simple secondary optics for which the light distribution can be approximated with a single cosine-power function. We therefore start with the analytical model from [8] using cosine functions:

$$M(\Theta) = \sum_i a_i * \cos(\Theta - b_i)^{c_i} \tag{1}$$

It was observed in [8] that a sum of only three cosine-power functions is sufficient in most cases. Our preliminary tests confirmed this observation, so we assumed that the sum of three cosine-power functions will probably be enough to fit LEDs with lenses that have symmetric radiation patterns with sufficient quality. In addition to the parameters of the original model, we introduce a normalizing parameter $I_{max}$, as this simplifies (unifies) the range of the other three parameters: $a = \{0, 0.001, 0.002, \ldots, 1\}$, $b = \{-90, -89.9, -89.8, \ldots, 90\}$ and $c = \{0, 1, 2, \ldots, 100\}$, for all test lenses. Doing all of the above we have rewritten the definition (1) as follows:

$$I(\Theta) = I_{max} \sum_i a_i * \cos(\Theta - b_i)^{c_i} \tag{2}$$

The expression (2) thus represents our analytical model to fit LEDs with attached secondary optics and symmetric spatial light distribution.

The goodness of fit is as usual [8, 10] defined to be minimizing the root mean square error (RMS), formally defined as:

$$RMS = \sqrt{\frac{1}{M}\sum_{i}\left[I(\Theta_i)_m - I(\Theta_i)\right]^2} \tag{3}$$

For a sufficiently accurate fit, the RMS value must be less than 5% [8, 10]. On the other hand, current standards and technology allows up to 2% noise in the measured data. Therefore, the target results of the fitting algorithms are at less than 5% RMS error, but at the same time there is no practical need for less than 1% or 2% RMS error.

We will assume that all the data are written in the form of vectors $v$ = (polar angle [$\Theta$], intensity [$I$]). In reality, measured photometric data from the lens manufacturers are available in one of the two standard coded formats. That are the IESNA photometric digital format *.ies [11] used primarily in the USA and the European format EULUMDAT *.ldt [2]. Conversion of the data in the two standard formats can easily be transformed into the list of vectors. In addition, due to the novel parameter $I_{max}$ each dataset will be normalized during the preprocessing so that in each instance the maximal intensity of the vectors will be 1, and the normalizing value $I_{max}$ is given as additional input value to the algorithms.

The problem can formally be written as:

INPUT: $I_{max}$ and a list of vectors $v$= (polar angle [$\Theta$], intensity [$I$])
TASK: Find parameters ($a_1$, $b_1$, $c_1$, $a_2$, $b_2$, $c_2$, $a_3$, $b_3$, $c_3$) that minimize the RMS error (3).

Different fitting algorithms were used to minimize RMS error. The algorithms are presented and compared in the next sections.

## 4. Fitting Algorithms

In this section we describe five fitting algorithms. As the main objective of this study was to obtain good solutions to the practical problem, the algorithms were chosen with this primary goal in sight. The selection is thus quite arbitrarily, and there may be some other algorithms or some other versions that might outperform the selected versions.

Note that here the problem is a continous optimization problem and hence, compared to discrete optimization, there are even more possibilities to define a neighborhood for the local search based heuristics. In fact, the neighborhoods we use can be seen as variable neighborhoods [7], although they are all similar. Of course, there may be other neighborhoods that would be worth consideration. The reason we keep the

selected neighborhoods and did not try to look for other possibilities is simply the fact that they already gave us results of sufficient quality. Another natural question that may be asked here is why use discrete optimization heuristics on a continuous optimization problem. First, there is no analytical solution for MST best approximation of this type of functions, and second, in order to apply continuous optimization methods such as the Newton method, usually we need a good approximation in order to assure convergence. As the target RMS error is between 1% and 5%, the fine approximation based on continuous optimization methods could be used as postprocessing. On the other hand, in view of the at least 2% noise in the data, this postprocessing is not of practical interest in this case.

We have started our experiments with two basic local search algorithms, steepest descent and iterative improvement, where in both cases the neighborhoods were defined in the same way, explained in more detail below. We call this neighborhood fixed stepsize neighborhood. The third local search algorithm is a variation of iterative improvement where we introduce random step size; roughly speaking, given a step size and direction as before, we randomly make a step in the direction that is at most as long as in the fixed size neighborhood search. Naturally, whenever local search is used, the multi start version is worth consideration. As preliminary testing of multi start version was not competitive with single longer runs, therefore we decided to use a more advanced heuristics that would on one hand take advantage of the seemingly successful local search and possibly accumulate information obtained by independent local searches. Our choice was to use a genetic algorithm. Finally, we also run and compare results of a simple generation of random solutions.

## 4.1 Steepest Descent – SD

The algorithm begins with the initialization of the initial function parameter values that are $a_1 = a_2 = a_3 = 0.5$, $b_1 = b_2 = b_3 = 0$, and $c_1 = c_2 = c_3 = 1$. Next it initializes the search step values which are for $da = 0.01$, for $db = 1$ and for $dc = \frac{I_{max}}{10}$ giving the 512 neighbors of the initial solution: $(a_1 \pm da, b_1 \pm db, c_1 \pm dc,\ a_2 \pm da, b_2 \pm db, c_2 \pm dc,$ $a_3 \pm da, b_3 \pm db, c_3 \pm dc)$. If there is a neighbor with better RMS value, the search moves to the neighbor with minimal RMS value (if there are more minimal neighbors, all are chosen with the same probability). If none of the 512 is better than the current solution a new set of neighboring solutions are generated, this time with a double step. It goes for ten steps and if there still is no better solution it breaks the search, multiplies the step value with 0.9, so the step is finer, and begins the search from start

in the neighborhood of the current solution. The algorithm stops when the number of generated solutions reaches $T_{max}$.

### 4.2 Iterative Improvement – Fixed Neighborhood – IF

The algorithm uses the same neighborhood as SD. Instead of considering all 512 neighbors at once, the algorithm generates a neighbor randomly, and moves to the neighbor if its RMS value is better than the current RMS value. If no better neighbor is found after 1000 trials, it is assumed that no better neighbor exists. As above, the algorithm changes to a new neighborhood, this time with a double step. It goes for ten steps and if there is still no better solution, it breaks the search, multiplies the step value with 0.9, so the step is finer and begins search from the start in the neighborhood of the current solution. The algorithm stops when the number of generated solutions reaches $T_{max}$.

### 4.3 Iterative Improvement – Random Neighborhood – IR

The search begins as the previous two algorithms. It initializes the same initial function parameter values. Next it initializes the search step value within a range, rather than a static fixed value. The ranges are for $da_1 = da_2 = da_3 = \{-0.1, -0.099, -0.098, \ldots, 0.1\}$, for $db_1 = db_2 = db_3 = \{-9, -8.9, -8.8, \ldots, 9\}$ and $dc_1 = dc_2 = dc_3 = \{-10, -9, -8, \ldots, 10\}$. It begins generating solutions, using the step range around the initial solution and calculating their RMS error. As soon as it generates a better solution, it stops, shifts the focus on that solution, resets the step range to the initial value, and continues the search in the neighborhood of the new best solution. If after 400000 generated solutions no better is found, than the step range gets doubled, and the search continues in the current neighborhood with a larger neighborhood. The stopping condition are the same as before, whichever is achieved first stops the search.

### 4.4 Genetic Algorithm – GA

The genetic algorithm mimics the evolutionary behavior [4,6,9]. Three genetic operators are used. The natural selection [4] where the best in a population survive, mutation [9] where randomly chosen parameters of a surviving solution are changed producing a new solution, and crossover [4,6,9] or breading where a new solution is created by randomly combin-

ing and crossing parameters from two randomly chosen solutions that have survived from the previous generation.

The algorithm begins with the generation and calculation of one hundred and fifty thousand solutions for the zero population. It then chooses via the natural selection ten best solutions. These ten are then locally optimized with the algorithm that implements the iterative improvement with random neighborhood for sixty thousand iterations. After that it generates the next generation from the ten best solutions of the previous generation. It generates twenty-five thousand mutant solutions and twenty-five thousand crossover solutions. More precisely, the mutation operator works as follows: in the randomly chosen individual, one to nine parameters are chosen to be changed (mutated) by adding to the current parameter value a randomly chosen value for $da_1 = da_2 = da_3 = \{-0.01, -0.009, -0.008, \ldots, 0.01\}$, for $db_1 = db_2 = db_3 = \{-0.25, -0.24, -0.23, \ldots, 0.25\}$ and $dc_1 = dc_2 = dc_3 = \{-2.5, -2.4, -2.3, \ldots, 2.5\}$. The crossover operation takes two randomly chosen individuals and choses one to nine parameters to be changed. The new parameter values are generated by calculating the difference of the according parameter pair of the two individuals $(a_1 i a_2 i, b_1 i - b_2 i, c_1 i - c_2 i)$, randomly choosing a value larger than zero and smaller than the calculated difference and adding the chosen value to the smaller parameter value of the pair. Then it chooses the best ten from that generation and optimizes them. It compares the optimized solutions from the previous generation and the current one. Again it chooses the ten best from both. After that it begins with the generation of the next generation, following the same steps as before.

The stopping condition is based on the number of generated solutions as for other algorithms. In our experiment, the usual number of generated solutions was four million, which here means that the algorithm stops when it optimizes the best solutions of generation five.

### 4.5 Blind Random Search – RAN

$T_{max}$ times generates random values of the parameters, evaluates the RMS error, and remembers the best so far solution.

## 5. Results

We discuss the results of a comparative experiment in which all the algorithms were run with $T_{max} = 4$ million. All algorithms were saving a log file during the runtime process, so we can extract the values at any particular time of the process.

*Table 1.* RMS error after 4 million calculating operations.

| Lens/Algorithm | SD | IF | RAN | IR | GA |
|---|---|---|---|---|---|
| C13353 | 3.991 | **3.408** | 7.013 | 8.671 | **3.061** |
| CA11265 | 2.775 | **2.372** | 4.936 | 4.798 | **2.729** |
| CA11268 | **2.227** | **2.229** | 4.100 | 2.471 | 2.578 |
| CA11483 | **3.100** | **3.066** | 4.130 | 3.387 | 3.141 |
| CA11525 | 3.150 | **1.108** | 3.217 | 1.907 | **1.087** |
| CA11934 | 3.940 | **2.514** | 4.196 | 3.543 | **2.909** |
| CA12392 | **1.636** | **1.641** | 3.424 | 2.445 | 2.277 |
| CA13013 | 1.202 | **0.695** | 2.136 | 2.241 | **0.916** |
| CP12632 | 5.537 | 5.493 | **4.918** | 4.974 | **4.362** |
| CP12633 | 2.431 | **2.415** | 4.063 | 3.708 | **2.347** |
| CP12636 | **2.348** | **2.107** | 4.571 | 4.217 | 2.479 |
| FP13030 | **2.267** | **2.257** | 3.762 | 3.659 | 2.414 |

For the purpose of the algorithm evaluation, we have chosen a set of real available lenses to be approximated. The set was chosen from the online catalogue of one of the biggest and most present manufacturer in the world Ledil Oy Finland [5]. The choosing from the broad spectrum of lenses in the catalogue was based on the decision that the used LED is Cree XP-E [3], and the demand that the lenses have a symmetric spatial light distribution. We have preserved the lens product codes from the catalogue, so the reader can find the lens by searching the catalogue for the code from the first column in table 1 .

### 5.1 Quality Comparison

In the Table 1 below the overall best solutions after the long runs of all algorithms on all twelve instances from the dataset are given. Recall that the results are acceptable if they have RMS values lover than 5% and that the approximation better than 1% is not of any use because of the noise in data. The best two results for each instance are in bold.

First, observe that all the algorithms in most of the cases give acceptable results, i.e. lower than 5 which is the same as 5% recalling the meaning of the normalizing parameter $I_{max}$. If we take a closer look, at the values we can see that the iterative improvement with fixed size IF is the winner when counting the number of best solutions, achieving the best solution in six out of twelve instances. The second best is the genetic algorithm with four best solutions, followed by the steepest

*Table 2.* RMS error after 750 thousand calculating operations.

| Lens/Algorithm | SD | IF | RAN | IR | GA |
|---|---|---|---|---|---|
| C13353 | 6.617 | **4.284** | 9.252 | 9.909 | **3.784** |
| CA11265 | **3.,477** | **2.700** | 7.282 | 5.073 | 4.183 |
| CA11268 | **2.376** | 2.620 | 5.893 | **2.471** | 2.932 |
| CA11483 | 4.181 | **3.400** | 4.130 | 3.784 | **3.641** |
| CA11525 | 3.813 | **3.395** | 4.811 | 3.789 | **1.601** |
| CA11934 | 4.032 | **1.662** | 4.988 | **3.543** | 3.789 |
| CA12392 | **1.814** | **1.661** | 3.597 | 2.717 | 2.577 |
| CA13013 | 2.804 | 3.115 | **2.136** | 2.241 | **1.331** |
| CP12632 | 9.501 | 9.839 | 8.474 | **5.054** | **4.703** |
| CP12633 | **2.465** | 4.511 | 4.757 | 4.296 | **2.613** |
| CP12636 | 5.000 | 6.297 | 5.506 | **4.217** | **3.803** |
| FP13030 | **2.800** | 5.679 | 6.611 | 3.659 | **3.233** |

descent with two. Second, comparing the three local search algorithms and the genetic algorithm in terms of the quality of their best solutions on particular instances, we see that all best solutions are within 1%. We can conclude that all four are fairly comparable in terms of the expected quality of the solution. On the other hand, the blind random search on average does not produce as good results as the other four, however it may luckily guess good solutions, in one case even the best solution obtained (on instance CP12632).

As we have so many results of acceptable quality, a natural question is whether the time limit chosen above could be shortened. The long runs in our implementations took 30 minutes for every run on a Intel Core I3-4130 @ 3,5 Ghz, programmed in C++. (The code was not optimized). Therefore it is interesting to compare shorter runs, see Table 2.

The shorter runs again show that most algorithms achieve the 5% error bound already in short runs. It may be interesting to note that the genetic algorithm is the only one that in the short runs finds solutions under 5% bound for all instances. In addition, it is also the winner in eight out of twelve cases looking the best obtained solution. We also observe that in short runs, the two algorithms based on fixed size neighborhood outperform the random size neighborhood iterative improvement. As expected, blind random search is not competitive on average, however curiously it is the winner on one instance.

Finally, comparing the speed of convergence we observe that all of the algorithms have a very steep convergence curve, a typical example is given in Figure 2.

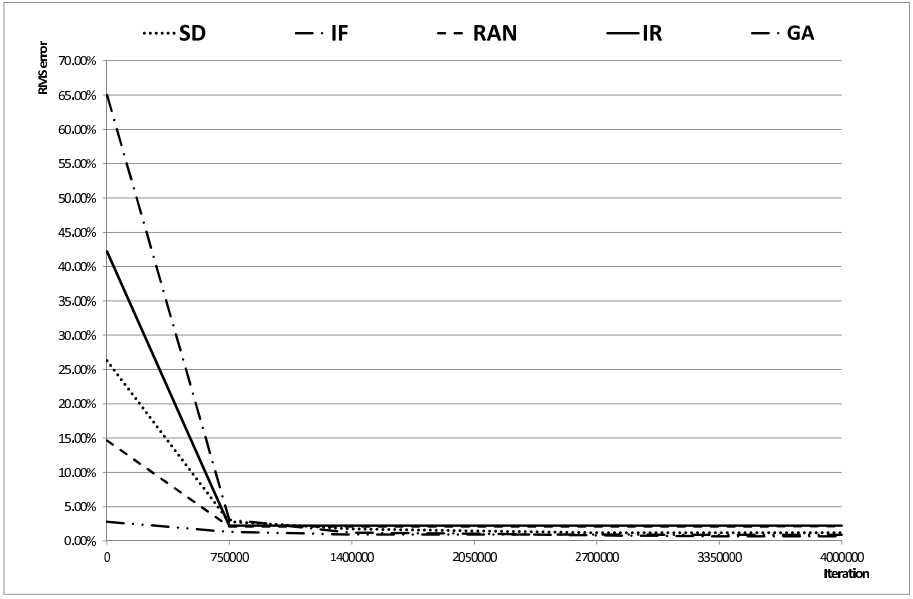



*Figure 2.* Linear interpolation of the approximation runtime process of CA13013 lens.

## 6. Conclusions

The goal of this part of the research was to design an efficient algorithm to fit an analytic model to the measured data of LED and secondary lens combination with symmetric spatial light distribution. We have designed several algorithms, and tested them on real lens data. The results of the test showed that, four of the algorithms produce approximations of acceptable quality. As the dataset used for testing includes a good variation of realistic LED and secondary lens combinations we can conclude that the practical approximation and design problem has been solved.

From theoretical viewpoint it is interesting to note that the genetic algorithm was very competitive, in short runs comparison even the best among the tested algorithms. Of course, it is well known that with fine tuning of parameters most of metaheuristics can be adopted to be very successful on a particular dataset. However, here we should add that our study started with testing the neighborhoods for local search and only

in the last part of the research we used the genetic algorithm. So in this case the parameter tuning of all algorithms took about the same effort. We thus believe that the comparison is fair also from this viewpoint.

The study presented here gave important information about the number of complexity of solving the general problem in the case of instances with symmetric spatial light distribution. Future work includes adaptation of the model to lenses with asymmetric spatial light distribution. Based on the new models, the heuristic data fitting that would lead to the construction of desired light engines, analogous to the work presented here may be possible. The general model will presumably include a larger number of parameters which in turn most probably means larger search spaces and more challenging optimization problems.

## Acknowledge

This work is supported in part by ARRS, the research agency of Slovenia.

## References

[1] Escaping the bulb culture: the future of LEDs in architectural illumination. *LEDs magazine*, (1):13–15, April 2005.

[2] I. Ashdown. Thinking Photometrically Part II. In *LIGHTFAIR 2001 Pre-Conference Workshop*, March 2001.

[3] Cree Inc. `http://www.cree.com/led-components-and-modules/products/xlamp/discrete-directional/xlamp-xpe`. Accessed 2014.

[4] R. L. Haupt and S. E. Haupt. *Practical Genetic Algorithms. 2nd Edition.* John Wiley & Sons, 2004.

[5] Ledil Oy. `http://www.ledil.com/`. Accessed 2014.

[6] M. Mitchell. *An Introduction to Genetic Algorithms. 5th Edition.* MIT Press, 1999.

[7] N. Mladenovic, P. Hansen, and J. Brimberg. Sequential clustering with radius and split criteria. *Cent. Eur. J. Oper. Res.*, 21(Suppl. 1):95–115, 2013.

[8] I. Moreno and C.-C. Sun. Modeling the radiation pattern of LEDs. *Opt. Express.*, 16(3):1808–1819, February 2008.

[9] D. Simon. *Evolutionary Optimization Algorithms.* John Wiley & Sons, 2013.

[10] C.-C. Sun, T.-X. Lee, S.-H. Ma, Y.-L. Lee, and S.-M. Huang. Precise optical modeling for led lighting verified by cross correlation in the midfield region. *Opt. Lett.*, 31:2193–2195, 2006.

[11] The Subcommittee on Photometry of the IESNA Computer Committee. IESNA standard file format for the electronic transfer of photometric data and related information. Technical Report ANSIDESNA LM-63-02, Illuminating Engineering Society of North America, 2002.

# PARALLEL CUDA IMPLEMENTATION OF THE DESIRABILITY-BASED SCALARIZATION APPROACH FOR MULTI-OBJECTIVE OPTIMIZATION PROBLEMS

Eren Akca
*HAVELSAN A.Ş., Ankara, Turkey*
erenakca88@gmail.com

Ökkes Tolga Altınöz
*Department of Electrical and Electronics Engineering*
*Faculty of Engineering and Architecture, TED University, Ankara, Turkey*
tolga.altinoz@tedu.edu.tr

Sadi Uçkun Emel
*HAVELSAN A.Ş., Ankara, Turkey*
semel@havelsan.com.tr

Asım Egemen Yilmaz
*Electrical and Electronics Engineering Department*
*Ankara University, Ankara, Turkey*
aeyilmaz@eng.ankara.edu.tr

Murat Efe
*Electrical and Electronics Engineering Department*
*Ankara University, Ankara, Turkey*
efe@eng.ankara.edu.tr

Tayfur Yaylagul
*HAVELSAN A.Ş., Ankara, Turkey*
tyaylagul@havelsan.com.tr

**Abstract** In this study, we present the results obtained for the parallel CUDA implementation of the previously proposed desirability-based scalarization approach for the solution of the multi-objective optimization problems. Our simulations show that compared to the sequential Java implementation, it is possible to find the same solutions (up to 16-time faster manner) by parallel CUDA implementation. We also try to outline our experiences of troubleshooting throughout the implementation as guidelines for upcoming researchers working in the same field.

**Keywords:** Aggregation, CUDA, Desirability functions, Genetic algorithm, GPGPU programming, Multi-objective optimization, Parallelization, Scalarization.

## 1. Introduction

The problem for determining the best possible solution set with respect to multiple objectives is referred to as a multi-objective (MO) optimization problem. There are many approaches for the solution of these kinds of problems. The most straightforward approach, the so-called "scalarization" or "aggregation" is nothing but to combine the objectives in order to obtain a single-objective [6]; and the most common method of this sort is the weighted sum [5, 7, 8].

The solution of the MO optimization problem is a set that contains trade-off solutions [11]. On the other hand, the problem instance defined via the scalarization technique yields a single solution. In order to find another trade-off solution, the parameters throughout the scalarization process shall be varied, and the resulting problem instance (which is different than the previous one) shall be solved. For the particular case in which the problem is bi-objective (with the objective functions $f_1$ and $f_2$) and the weighted-sum method is applied, the aggregated objective function is $w_1 f_1 + (1 - w_1) f_2$, where the weight $w_1$ is a real number between 0 and 1. By varying this weight, a new single-objective problem instance (and a new solution) is obtained. Obviously, for this simple case, the parameter setup of the "scalarization scheme" consists of only the weight $w_1$. For higher number of objectives and different scalarization techniques, the scalarization scheme might address a parameter set with multiple elements.

Scalarization techniques were popular in 1980s and early 1990s, prior to development of powerful multi-objective optimization algorithms such as the Non-Dominated Sorting Genetic Algorithm (NSGA) [10], NSGA-II [3] or Vector Evaluated Genetic Algorithm (VEGA) [9], etc. After the development of these powerful and successful multi-objective optimization algorithms, scalarization techniques were considered to be old-fashioned, and they were abandoned. By the time, especially af-

ter the evolution and rapid development of multi-core architectures in 2000s, researchers have started to reconsider and revisit the scalarization techniques since these techniques are usually suitable for parallelization when carefully implemented. In this study, with a similar motivation, we demonstrate how one of these techniques can be parallelized and implemented on the General Purpose Graphic Processing Units (GPGPUs) via the Compute Unified Device Architecture (CUDA) framework.

## 2. The Main Idea Beneath the Scalarization and the Weighted Sum Approach

As stated in the previous section, the main aim in a multi-objective optimization problem is to find a set of trade-off solutions. The optimality of a particular solution is determined via the definition of "domination" in the Pareto space. In Figure 1, the pictorial descriptions of the domination and the set of non-dominated solutions (i.e. the Pareto front) are given for a simple bi-objective problem.

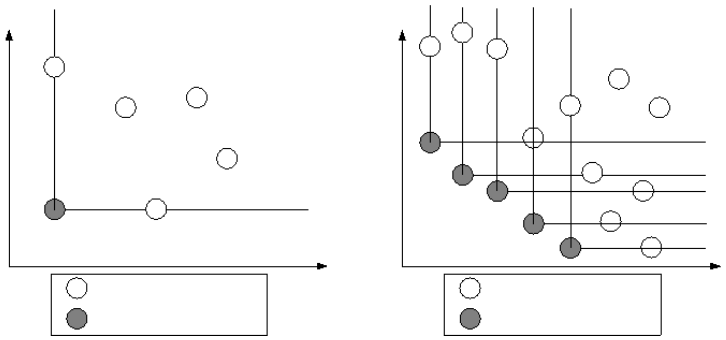


*Figure 1.* Pictorial descriptions of domination and the Pareto front.

In order to obtain a set of solutions for a bi-objective optimization problem via the weighted-sum method (for which the aggregated objective function $w_1 f_1 + (1 - w_1) f_2$ is constructed), the weight $w_1$ shall be varied between 0 and 1 in a systematical manner; and a new solution shall be found for each value of $w_1$. Even though this approach seems to be simple and well-working, it fails especially when the Pareto front is concave (totally or between two particular points in the Pareto space). In order to illustrate this, let us try to understand the main idea beneath the weighted-sum approach. As seen in Figure 2, the aggregated objective function $w_1 f_1 + (1 - w_1) f_2$ corresponds to a line in the Pareto space (particularly the $f_1 f_2$ plane). Throughout the optimization process, in which $c = w_1 f_1 + (1 - w_1) f_2$ is minimized, the corresponding

line is shifted by preserving its slope. Eventually, the solution obtained is nothing but the intersection of the corresponding line with the Pareto front curve (which is considered to be convex, for the time being) as seen in Figure 3.

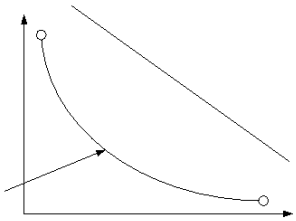


*Figure 2.* Pictorial description of the solution via the weighted sum approach for convex Pareto front.

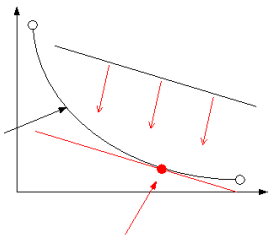


*Figure 3.* Pictorial description of how different solutions can be found by altering the weight in the weighted sum approach for convex Pareto front.

As seen in Figure 2, the slope of the line is determined by the weight, and altering this parameter will yield a different line. As seen in Figure 3, altering the weight would yield a different solution (in case the Pareto front is convex).

On the other hand, let us consider the case for which the Pareto front is concave between two points A and B as seen in Figure 4. In this case, the point B is found as a solution. In case the weight is altered as seen in Figure 5, the point A is found as an alternative solution. Unfortunately, in such a case, even if the weight is altered in its whole range, no points on the Pareto front other than A and B can be found by this approach [2]. Without a topological proof, this can be observed pictorially by intersecting different-slope lines with this concave Pareto front in Figures 4 and 5.

Since it is impossible to know whether the Pareto front is convex or concave in real life problems, the weighted-sum approach cannot be applied confidently. Hence, in this study we propose to apply the desirability-functions (by altering their shapes systematically) for scalarization as defined in [1]. The next section is devoted to description of this method.

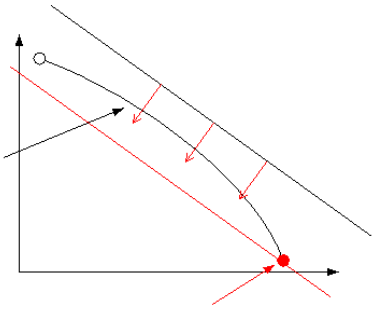


*Figure 4.* Pictorial description of the solution via the weighted sum approach for concave Pareto front (where the point B is found as the solution).

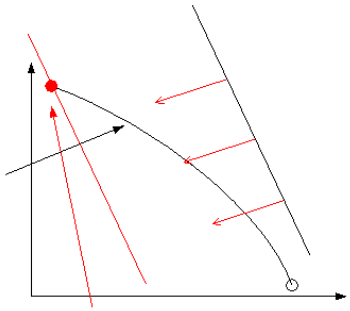


*Figure 5.* Pictorial description of the solution by altering the weight in the weighted sum approach for concave Pareto front (where the point A is found as the alternative solution).

## 3. Desirability Function-Based Scalarization

Previously in [1], an alternative scalarization method utilizing the so-called desirability functions was proposed. The concept of the desirability functions was first introduced by Harrington in 1965 for multi-objective industry quality control. After the proposition of the desirability function concept, Deringer and Suich [4] introduced different desirability function formulations. The main idea beneath the desirability functions is as follows:

- The desirability function is a mapping from the domain of real numbers to the range set $[0, 1]$.

- The domain of each desirability function is one of the objective functions; and it maps the values of the relevant objective function to the interval $[0, 1]$.

- Depending on the desire about minimization of each objective function (i.e. the minimum/maximum tolerable values), the relevant desirability function is constructed.
- The overall desirability value is defined as the geometric mean of all desirability functions; this value is to be maximized.

Particularly, for a bi-objective optimization problem in which the functions $f_1$ and $f_2$ are to be minimized, the relevant desirability functions $d_1(f_1)$ and $d_2(f_2)$ can be defined as in Figure 6. The desirability functions are not necessarily defined to be linear; certainly, non-linear definitions shall also be made as described in [1]. Throughout this study, we prefer the linear desirability functions.

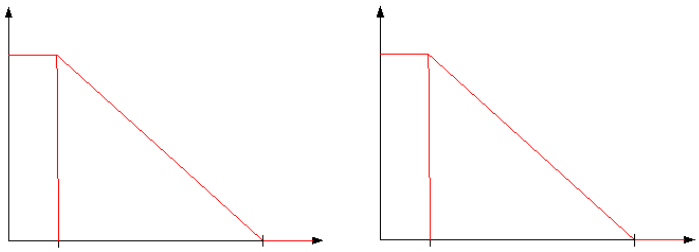


*Figure 6.* The linear desirability functions constructed for the bi-objective optimization problem.

In [1], a method for extraction of the Pareto front was proposed by altering the shapes of the desirability functions in a systematical manner. Particularly, by:

- Fixing the parameters $f_{1_{\text{max\_tol}}}$ and $f_{2_{\text{max\_to}}}$ seen in Figure 6 at infinity, and
- Varying the parameters $f_{1_{\text{min\_tol}}}$ and $f_{2_{\text{min\_tol}}}$ systematically,

it is possible to find the Pareto front regardless of its convexity or concavity. This claim can be illustrated for the bi-objective case as follows: as seen in Figure 7, the parameters $f_{1_{\text{min\_tol}}}$ and $f_{2_{\text{min\_tol}}}$ determine the sector which is traced throughout the solution. The obtained solution corresponds to a point for which the geometric mean of the two desirability values. As seen in Figure 8, even in the case of concave Pareto front, the solution can be found without loss of generality. In other words, unlike the weighted-sum approach, the method proposed in [4] does not suffer from the concave Pareto fronts.

In [1], the applicability and the efficiency of the proposed scalarization approach was demonstrated via some multi-objective benchmark functions. Each single-objective problem (i.e. the scalarization scheme) was

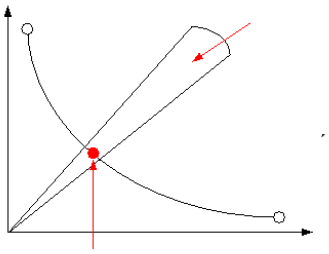


*Figure 7.* Pictorial description of the solution via the desirability-function based approach for convex Pareto front.

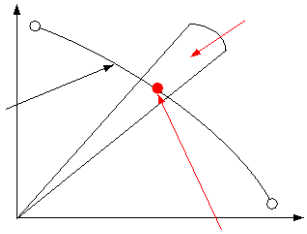


*Figure 8.* Pictorial description of the solution via the desirability-function based approach for concave Pareto front.

solved with Particle Swarm Optimization. Despite no explicit demonstration or proof, it was claimed that:

- There were no limitations about the usage of Particle Swarm Optimization; i.e. any other heuristic algorithm could be incorporated and implemented.
- The proposed method can be easily parallelizable.

In this study, we demonstrate the validity of these claims by incorporating the Genetic Algorithm in the proposed method, and performing a parallel implementation on GPGPUs via the CUDA framework. The next section is devoted to the implementation details.

## 4. Parallel CUDA Implementation of the Desirability Function-Based Scalarization

The main idea of our parallel implementation throughout this study is illustrated in Figure 9. Each scalarization scheme is handled in a separate thread; after the relevant solutions are obtained, they are gathered in a centralized manner to constitute the Pareto front from which the human decision maker picks a solution according to his/her needs. This

approach ensures that the number of solutions found that can be found in parallel is limited by the capability of the GPGPU card used. If the problem is a mission-planning problem as in our case, the two objectives might be "minimizing the overall mission completion time" and "minimizing the overall mission risk"; and the Pareto front is a plethora of various alternative mission plans with different overall completion time and risk values. In this case, the planner (or the commander) will pick up a solution (i.e. a mission plan to be executed) by using his/her own initiative. Regardless of the choice, it will be certain that the particular picked solution will be non-dominated.

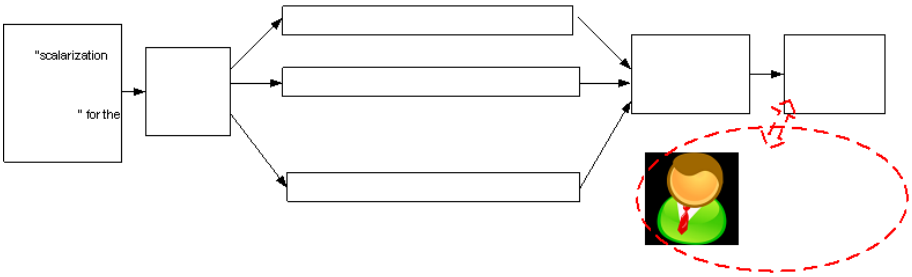



*Figure 9.* Pictorial description of the parallel CUDA implementation of the desirability-function based approach.

As stated before, we implemented an elitist Genetic Algorithm for verification of the aforementioned claims. The parallel CUDA implementation was compared to the sequential Java implementation in the environment seen in Table 1. The Genetic Algorithm parameters used throughout the simulations are given in Table 2.

*Table 1.* The environment in which the simulations are run.

| | |
|---|---|
| Device and Global Memory | Quadro K5000 and 4GB |
| CUDA Cores | 1536 |
| GPU and Memory Clock Rate | 706 MHz and 2700 MHz |
| Memory Bus Width | 256-bit |
| Maximum Number of Threads per MP and Block | 2048 and 1024 |
| CUDA Driver and Capability | 5.5 and 3.0 |

*Table 2.* Genetic Algorithm parameters throughout the simulations.

| | |
|---|---|
| Chromosome size | 16-bit |
| Population size | 100 |
| Number of generations (iterations) | 100 |
| Elitism Rate | 0.2 |
| Mutation Rate | 0.1 |
| Crossover Rate | 0.9 |

It was seen that both implementations (sequential Java and parallel CUDA) were able to find the same solutions but in different elapsed times. As seen in Figure 10, if the number of Pareto front solutions increase, the advantage of the parallel CUDA appears dramatically.

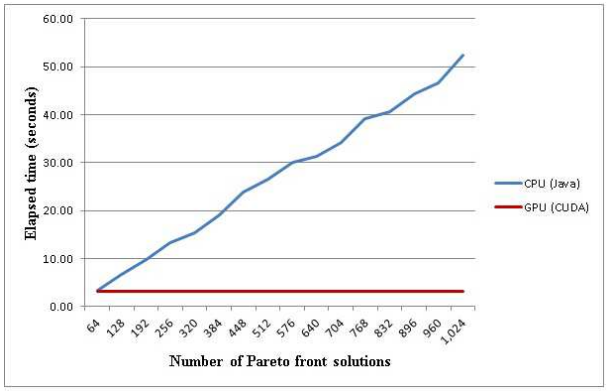



*Figure 10.* Comparison of the sequential Java and the parallel CUDA implementations.

These results show how efficient the parallel CUDA implementation will be in case numerous Pareto solutions are required by the decision maker in a multi-objective optimization problem.

## 5. Lessons Learned and Future Work

Throughout the implementation, we have experienced and observed the following:

- Memory allocations shall be made in advance in a bulk manner, since such operations deteriorate parallelism in case they are made

one by one. Bulk memory allocation and pointer assignment for usage of each thread is possible in our case, since it is quite straightforward to determine how much memory will be required by the Genetic Algorithm (since the parameters listed in Table 2 are definite).

- Random number generation is another issue. Various alternative approaches can be preferred:
  - To generate each random number at the CPU when required and to transfer it to the GPU: This is the least efficient and the slowest approach.
  - To generate all random numbers at the CPU prior to the solution and to transfer them to the GPU: This is better than the previous one.
  - To generate each random number directly at the GPU with no CPU-GPU communication need: This is the most efficient and the fastest approach. GPU generated random numbers demonstrate sufficient randomness features.
- Windows operating systems assigns a default 2-second time-out for the processes initiated at the peripheral devices. As seen in Figure 10, for our problem, the duration of the execution of each CUDA thread was about 3 seconds (which exceeds the 2-second time-out), causing the relevant process to be killed by the Windows operating system prior to completion. This problem was resolved by brute-force editing (and setting it to higher values) of the relevant key value in the registry.

The results shown in Figure 10 were obtained without any parallel/concurrent implementation of the single-objective optimization problem. As seen in Figure 11, some steps in the Genetic Algorithm can be executed in a parallel or concurrent manner.

More specifically, the computation of the objective and the fitness functions for the individuals in the Genetic Algorithm population can be performed concurrently on GPGPUs, in case the stream mechanism in CUDA is utilized as seen in Figure 12. The impact of such a modification in the implementation might not be drastic for the benchmark problems since they usually consist of computationally cheap functions. But in case of computationally expensive functions as in our mission-planning problem, the advantage of utilization of the streams is expected to be quite dramatic. As a future work, our aim is to perform the utilization of the streams as well as the shared memories in the GPGPUs, which are not being used in the current implementation.

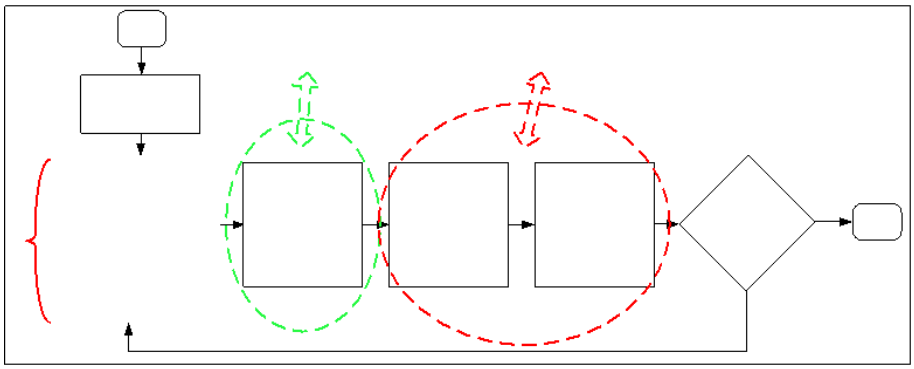


*Figure 11.* The steps of the Genetic Algorithm with indications of parallelization.

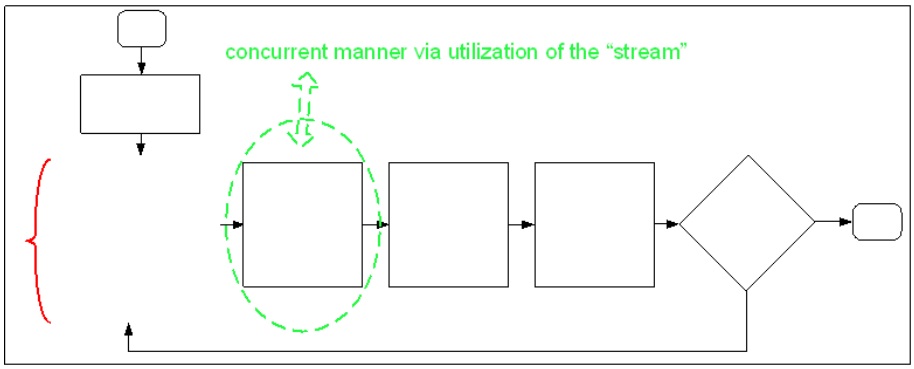



*Figure 12.* The step of the Genetic Algorithm which can be executed concurrently via utilization of the stream mechanism in CUDA.

In conclusion, in this study we have demonstrated that it is possible to achieve up to 16 times faster GPU implementations for the multi-objective problems via a careful and intelligent CUDA implementation. Further improvements in the implementation shall be made in the near future. Moreover, we have verified the claims in [4] and demonstrated the efficiency of the proposed approach.

## Acknowledgement

This study was made possible by grants from the Turkish Ministry of Science, Industry and Technology (Industrial Thesis - San-Tez Programme and HAVELSAN; with Grant Nr. 01568.STZ.2012-2) and the Scientific and Technological Research Council of Turkey – TUBITAK (with Grant Nr. 112E168). The authors would like to express their gratitude to these institutions for their support.

## References

[1] O. T. Altinoz, A. E. Yilmaz, and G. Ciuprina. A Multiobjective Optimization Approach via Systematical Modification of the Desirability Function Shapes. In *Proc. 8th International Symposium on Advanced Topics in Electrical Engineering*, pages 23–25, 2013.

[2] R. S. Burachik, C. Y. Kaya, and M. M. Rizvi. A new scalarization technique to approximate Pareto fronts of problems with disconnected feasible sets. *J. Optimiz. Theory App.*, appeared online, June 2013, DOI 10.1007/s10957-013-0346-0.

[3] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE T. Evolut. Comput.*, 2:182-197, 2002.

[4] G. Derringer and R. Suich. Simultaneous optimization of several response variables. *J. Qual. Technol.*, 12:214–219, 1980.

[5] J. Keski and R. Silvenneinen. Norm methods and partial weighting in multicriteria optimization of structures. *Int. J. Num. Meth. Eng.*, 24:1101–1121, 1987.

[6] R. Marler and S. Arora. Transformation methods for multiobjective optimization. *Eng. Optimiz.*, 37:551–569, 2009.

[7] R. Marler and S. Arora. The weighted sum method for multi-objective optimization: new insights. *Struct. Optimization*, 41:853–862, 2010.

[8] S. F. P. Saramago and V. Steffen, Jr. Optimization of trajectory planning of robot manipulators taking into account the dynamic of the system. *Mech. Mach. Theory*, 33:883–894, 1998.

[9] J. D. Schaffer. Multiple objective optimization with vector evaluated genetic algorithms. In *Proc. International Conference on Genetic Algorithm and their Applications*, 1985.

[10] N. Srinivas and K. Deb. Multi-Objective function optimization using non-dominated sorting genetic algorithms. *Evolutionary Computation*, 2:221-248, 1995.

[11] H. Trautmann and J. Mehmen. Preference-based pareto optimization in certain and noisy environments. *Eng. Optimiz.*, 41:23–38, 2009.

# DIFFERENTIAL EVOLUTION FOR SELF-ADAPTIVE TRIANGULAR BRUSHSTROKES

Uroš Mlakar
*Faculty of Electrical Engineering and Computer Science*
*University of Maribor, Slovenia*
uros.mlakar@um.si

Janez Brest
*Faculty of Electrical Engineering and Computer Science*
*University of Maribor, Slovenia*
janez.brest@um.si

Aleš Zamuda
*Faculty of Electrical Engineering and Computer Science*
*University of Maribor, Slovenia*
ales.zamuda@um.si

**Abstract** This paper proposes a lossy image representation where a reference image is approximated by an evolved image, constituted of variable number of triangular brushstrokes. The parameters of each triangle brush are evolved using differential evolution, which self-adapts the triangles to the reference image, and also self-adapts some of the control parameters of the optimization algorithm, including the number of triangles. Experimental results show the viability of the proposed encoding and optimization results on a few sample reference images.

**Keywords:** Differential evolution, Evolutionary computer vision, Evolutionary art, Image-based modeling, Self-adaptation, Triangular brushstrokes.

## 1. Introduction

In this paper, evolvable lossy image representation utilizing an image compared to its evolved generated counterpart image, is proposed. The

image is represented using a variable number of triangular brushstrokes [5], each consisting of triangle position and color parameters. These parameters for each triangle brush are evolved using differential evolution [3, 10], which self-adapts the control parameters, including the proposed self-adaptation for the number of triangles to be used. Experimental results show the viability of the proposed encoding and evolution convergence for lossy compression of sample images.

The approach presented is built upon and compared with [5], by addressing and also extending the original challenge. Namely, the challenge introduced in [5] uses triangles in trying to build an approximate model of an image [5]. The triangle is an efficient brush shape for this challenge, since it covers more pixels than a single point, and also allows overlaying and blending of colors over several regional surface pixels, which lines can not. Also, an arbitrary triangle shape is less constrained than any further point-approximated shape, and also other shapes can be built by combining several triangles.

Instead of genetic programming in [5], in this paper differential evolution is used with a fixed size tree-like chromosome vector, which is cut-off self-adaptively to form codon and anti-codon parts of the chromosome. Also, our approach uses a modified challenge, where we can reconstruct the model for the reference image solely using the evolved model without using the reference image, whereas the [5] needs the reference image when drawing pixels to the canvas in deciding which pixels match the reference image for accepting them into the evolved canvas. Also, in this paper the triangle brushstroke encoding differes and is proposed especially designed for an efficient DE encoding.

In the following section, related work is presented, then the proposed approach is defined. In Section 4, the experimental results are reported. Section 5 concludes the paper with propositions for future work.

## 2. Related Work

In this section, related work on evolutionary computer vision, evolutionary art, image representation, and evolutionary optimization using differential evolution, are presented. These topics are used in the proposed method, defined in the next section.

### 2.1 Image-Based Modeling, Evolutionary Computer Vision, and Evolutionary Art

Image-based approaches to modeling include processing of images, e.g. two-dimensional, from which after segmentation certain features are extracted and used to represent a geometrical model [7]. For art

drawings modeling, automatic evolutionary rendering has been applied [2, 9]. In [11] animated artwork is evolved using an evolutionary algorithm. Then, Izadi et al. [5] evolved triangular brushstrokes challenge using genetic programming for two-dimensional images, using unguided and guided searches on a three or four branch genetic program, where roughly 5% similarity with reference images was obtained on average per pixel. In this paper, we build upon and compare our new approach with [5], by addressing and also extending its challenge. After extending the challenge, we optimize it using DE, which is described in the next section.

### 2.2 Evolutionary Optimization Using Differential Evolution

Differential evolution (DE) [10] is a floating-point encoding evolutionary algorithm for continuous global optimization. It has been modified and extended several times with various versions being proposed [4]. DE has also been applied to remote sensing image subpixel mapping [14], image thresholding [8], and for image-based modeling using evolutionary computer vision to reconstruct a spatial procedural tree model from a limited set of two dimensional images [12,13]. Neri and Tirronen in their survey on DE [6] concluded that, compared to the other algorithms, a DE extension called jDE [3], is superior to the compared algorithms in terms of robustness and versatility over a diverse benchmark set used in the survey. Therefore, we choose to apply jDE in this approach.

The original DE has a main evolutionary loop where a population of vectors is computed within each generation. For one generation, counted as $g$, each vector $\mathbf{x}_i$, $\forall i \in \{1, ..., NP\}$ in the current population of size $NP$, undergoes DE evolutionary operators, namely the mutation, crossover, and selection. Using these operators, a trial vector (offspring) is produced and the vector with the best fitness value is selected for the next generation. For each corresponding population vector, mutation creates a mutant vector $\mathbf{v}_{i,g+1}$ (*'rand/1'* [10]):

$$\mathbf{v}_{i,g+1} = \mathbf{x}_{r_1,g} + F(\mathbf{x}_{r_2,g} - \mathbf{x}_{r_3,g}), \tag{1}$$

where the indexes $r_1$, $r_2$, and $r_3$ are random and mutually different integers generated in from set $\{1, ..., NP\}$, which are also different from $i$. $F$ is an amplification factor of the difference vector, mostly within the interval $[0, 1]$. The term $\mathbf{x}_{r_2,g} - \mathbf{x}_{r_3,g}$ denotes a difference vector, which is named the amplified difference vector after multiplication with $F$. The mutant vector $\mathbf{v}_{i,g+1}$ is then used for recombination, where with the target vector $\mathbf{x}_{i,g}$ a trial vector $u_{i,j,g+1}$ is created, e.g. using binary

crossover:

$$u_{i,j,g+1} = \begin{cases} v_{i,j,g+1}, & \text{if } rand(0,1) \leq CR \text{ or } j = j_{\text{rand}}, \\ x_{i,j,g} & \text{otherwise,} \end{cases} \tag{2}$$

where $CR$ denotes the crossover rate, $\forall j \in \{1, ..., D\}$ is a $j$-th search parameter of $D$-dimensional search space, $rand(0,1) \in [0,1]$ is a uniformly distributed random number, and $j_{\text{rand}}$ is a uniform randomly chosen index of the search parameter, which is always exchanged to prevent cloning of target vectors. Since the jDE self-adapts the $F$ and $CR$ control parameters to generate the vectors $\mathbf{v}_{i,g+1}$ and $\mathbf{u}_{i,g+1}$, corresponding values $F_i$ and $CR_i$, $\forall i \in \{1, ..., NP\}$ are updated prior to their use in the mutation and crossover mechanisms:

$$F_{i,g+1} = \begin{cases} F_{\text{l}} + rand_1 \times F_{\text{u}} & \text{if } rand_2 < \tau_1, \\ F_{i,g} & \text{otherwise,} \end{cases} \tag{3}$$

$$CR_{i,g+1} = \begin{cases} rand_3 & \text{if } rand_4 < \tau_2, \\ CR_{i,g} & \text{otherwise,} \end{cases} \tag{4}$$

where $\{rand_1, ..., rand_4\} \in [0,1]$ are uniform random floating-point numbers and $\tau_1 = \tau_2 = 0.1$. Finally, the selection operator evaluates and compares the trial to current vector and propagates the fittest:

$$\mathbf{x}_{i,g+1} = \begin{cases} \mathbf{u}_{i,g+1} & \text{if } f(\mathbf{u}_{i,g+1}) < f(\mathbf{x}_{i,g}), \\ \mathbf{x}_{i,g} & \text{otherwise.} \end{cases} \tag{5}$$

## 3. Differential Evolution for Self-Adaptive Triangular Brushstrokes

In this section, the encoding aspect, genotype-phenotpye rendering, and evaluation mechanisms of the proposed approach are defined.

### 3.1 Encoding Aspect

We encode an individual compressed image into a DE vector as follows. A DE vector $\mathbf{x} = (x_1, x_2, ..., x_{8T^{\max}}, F, CR, T^{\text{L}}, T^{\text{U}})$ is composed of floating-point scalar values packed sequentially as $\{x_j : \forall j \in \{1, ..., D + 4\}\}$, starting with a triangles-coding part of length $D = 8T^{\max}$, and the rest are the self-adaptive control parameters of the vector to be used during the DE. The self-adaptive control parameters part of the $\mathbf{x}$ vector encodes and uses the scaling factor $F$ and crossover rate $CR$ as in the jDE [3]; then the $T_i^{\text{L}}$, $T_i^{\text{U}} \in \{1, .., T^{\max}\}$ control parameters follow.

The self-adaptive $T_i^{\mathrm{L}}$ and $T_i^{\mathrm{U}}$ control parameters determine index-wise triangles encoded in the vector $\mathbf{x}$ to be used for rendering the evolved image, i.e. the portion of $\mathbf{x}$ to render an image is $\{x_j : \forall j \in \{T^{\mathrm{L}}, ..., T^{\mathrm{U}}\}\}$.

In this paper, we propose to have the whole vector represent a triangle set, organized similar to serializing a tree as a linear vector in visiting nodes by depth-first search. However, the leaf nodes are mostly exposed to being cut-off, whereas the root node is encoded in the middle of the vector and the near-root nodes are therefore more protected in being retained, since they are more anchored due to cut-offs mostly around the codon edges. After being included into a new trial vector, all nodes have an equal probability of having their triangle data changed.

In this way, the $T^{\mathrm{L}}$ and $T^{\mathrm{U}}$ allow us to render only a sub-portion of the triangles set, similarly to taking an inseparable portion of a GP tree traversal as in [5]. This gives us an arbitrary length render set, and keeps the crossover of anti-codon to help us find the number of triangles $T_i \in \{1, ..., T^{\max}\}$, which is more suitable for image approximation:

$$T_i = \begin{cases} T_i^{\mathrm{U}} - T_i^{\mathrm{L}} + 1 & \text{if } T_i^{\mathrm{L}} < T_i^{\mathrm{U}} \\ (T^{\max} - T_i^{\mathrm{L}}) + T_i^{\mathrm{U}} & \text{otherwise.} \end{cases} \tag{6}$$

The $T_i^{\mathrm{L}}$ and $T_i^{\mathrm{U}}$ are updated similarly to the $F_i$ control parameter:

$$T_{i,g+1}^{\mathrm{L}} = \begin{cases} \lfloor rand_1^{\mathrm{L}} \times T^{\max} \rfloor & \text{if } rand_2^{\mathrm{L}} < \tau^{\mathrm{L}}, \\ T_{i,g}^{\mathrm{L}} & \text{otherwise,} \end{cases} \tag{7}$$

$$T_{i,g+1}^{\mathrm{U}} = \begin{cases} \lfloor rand_1^{\mathrm{U}} \times T^{\max} \rfloor & \text{if } rand_2^{\mathrm{U}} < \tau^{\mathrm{U}}, \\ T_{i,g}^{\mathrm{U}} & \text{otherwise,} \end{cases} \tag{8}$$

where $\tau^{\mathrm{L}} = \tau^{\mathrm{U}} = \tau_1 = 0.1$ of the jDE.

### 3.2 Genotype-Phenotype Rendering

A DE vector $\mathbf{x}_i, \forall i \in \{1, ..., NP\}$ encoded using floating-point numbers $x_{i,j}, \forall j \in \{1, ..., D + 4\}$ constituting a genotype is rendered into a phenotype image $\mathbf{z}_i = \{\mathbf{z}_{i,x,y}\}$ of $R_{\mathrm{x}}$ width and $R_{\mathrm{y}}$ height in pixels, to be compared against a reference image $\mathbf{z}^*$ as follows.

The triangle brushstrokes (Figure 1) are represented as $(c_x, c_y, r, \alpha_1, \alpha_2, b^{\mathrm{Y}}, b^{\mathrm{Cb}}, b^{\mathrm{Cr}})$, where $c_x \in [0, ..., R_{\mathrm{x}})$, $c_y \in [0, ..., R_{\mathrm{y}})$, and $r \in [0, R_{\mathrm{x}}/\sqrt{T_{\max}}]$ define the circumscribed circle center and radius for the triangle to be rendered; $\alpha_1 \in [1°, 360°)$ and $\alpha_2 \in [1°, 180°)$ define the points of this triangle on its circumscribed circle; and $b^{\mathrm{Y}} \in [16, 236)$, $b^{\mathrm{Cb}} \in [16, 241)$, and $b^{\mathrm{Cr}} \in [16, 241)$ are the color components of the brush for the triangle contained pixels.

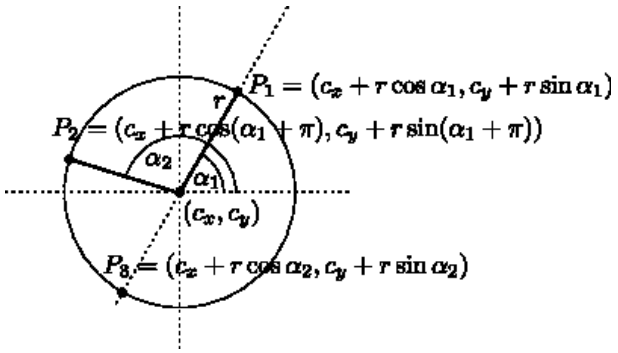



*Figure 1.* The triangle brush definition and the circumscribed circle.

The triangles' vertices encoded by $i$-th DE vector construct $T_i$ triangles, each triangle $\mathbf{T}_k = (c_{x,k}, c_{y,k}, r_k, \alpha_{1,k}, \alpha_{2,k}), \forall k \in \{1, ..., T_i\}$ ($\mathbf{T}_k$ being packed as $\mathbf{x}_i = \{x_{i,j}\}$, $j = 8k + m$, $m \in \{1, ..., 8\}$), defining the vertices of a triangle $P_{1,k}$, $P_{2,k}$, and $P_{3,k}$:

$$P_{1,k} = \lfloor (c_{x,k} + r_k \cos\alpha_{1,k}, c_{y,k} + r_k \sin\alpha_{1,k}) \rfloor , \tag{9}$$

$$P_{2,k} = \lfloor (c_{x,k} + r_k \cos(\alpha_{1,k} + \pi), c_{y,k} + r_k \sin(\alpha_{1,k} + \pi)) \rfloor , \tag{10}$$

$$P_{3,k} = \lfloor (c_{x,k} + r_k \cos\alpha_{2,k}, c_{y,k} + r_k \sin\alpha_{2,k}) \rfloor . \tag{11}$$

The brush color $\mathbf{b}_k^{\mathrm{YCbCr}} = (b_k^{\mathrm{Y}}, b_k^{\mathrm{Cb}}, b_k^{\mathrm{Cr}})$ is first transformed into RGB color model as $\mathbf{b}_k^{\mathrm{RGB}} = (b_k^{\mathrm{R}}, b_k^{\mathrm{G}}, b_k^{\mathrm{B}})$ $(b_k^{\mathrm{R}}, b_k^{\mathrm{G}}, b_k^{\mathrm{B}} \in [0, 255])$, where:

$$b_k^{\mathrm{R}} = \left\lfloor 1.164(b_k^{\mathrm{Y}} - 16) + 1.596(b_k^{\mathrm{Cr}} - 128) \right\rfloor \tag{12}$$

$$b_k^{\mathrm{G}} = \left\lfloor 1.164(b_k^{\mathrm{Y}} - 16) - 0.813(b_k^{\mathrm{Cr}} - 128) - 0.391(b_k^{\mathrm{Cb}} - 128) \right\rfloor \tag{13}$$

$$b_k^{\mathrm{B}} = \left\lfloor 1.164(b_k^{\mathrm{Y}} - 16) + 2.018(b_k^{\mathrm{Cb}} - 128) \right\rfloor \tag{14}$$

For each triangle $T_k$, a solid color is rendered without antialiasing over the triangle brush area rasterizing [1] with a transparency factor of $1/T_i$:

$$\mathbf{b}_k = \left\lfloor \frac{255}{T_i} \mathbf{b}_k^{\mathrm{RGB}} \right\rfloor . \tag{15}$$

This is analogous to blending the triangle as a part-transparent layer within the evolved image $\boldsymbol{Z}_i = \sum_k \mathbf{z}_{k,x,y}$ and computes R, G, and B color layers for the pixels of the $i$-th individual:

$$\mathbf{z}_{k,x,y} = \sum_{\mathbf{T}_k \text{ over } (x,y)} \mathbf{b}_{k,x,y} = \sum_{\mathbf{T}_k \text{ over } (x,y)} \left\lfloor \frac{255}{T_i} \mathbf{b}_{k,x,y}^{\mathrm{RGB}} \right\rfloor , \tag{16}$$

where $\mathbf{T}_k$ over $(x, y)$ denotes each triangle being rendered over the pixel $(x, y)$ such that $\mathbf{b}_{k,x,y}$ contains the rendered pixels of a brushstroke. Triangles defined possibly over the edges of image canvas are drawn by clipping away pixels outside of the canvas area.

The initialization of a genotype is such that the $c_x, c_y, \alpha_1, \alpha_2, b^{\mathrm{Y}}, b^{\mathrm{Cb}}$, $b^{\mathrm{Cr}}$, $T_i^{\mathrm{L}}$, and $T_i^{\mathrm{U}}$ are initialized uniform randomly to integer values within their respective definition intervals, while $r$ is kept as a floating-point. All parameters are however evolved as floating-point scalar values in DE.

### 3.3 Evaluation

Evaluation of the phenotype image $\boldsymbol{Z}_i$ to be compared against a reference image $\boldsymbol{Z}^*$ is as follows. A reference image $\boldsymbol{Z}^*$ is represented as RGB-encoded colored pixels integer values in layers $\boldsymbol{Z}^* = \{(z^{\mathrm{R}}_{x,y}, z^{\mathrm{G}}_{x,y}, z^{\mathrm{B}}_{x,y})\}$.

To obtain a difference assessment value, the following comparison metric is used for comparing an evolved image $\boldsymbol{Z} = \boldsymbol{Z}_i$ to $\boldsymbol{Z}^*$:

$$f(\boldsymbol{Z}) = 100 \times \frac{\sum_{y=0}^{R_{\mathrm{y}}-1} \sum_{x=0}^{R_{\mathrm{x}}-1} \mid z^{*\mathrm{R}}_{x,y} - z^{\mathrm{R}}_{x,y} \mid + \mid z^{*\mathrm{G}}_{x,y} - z^{\mathrm{G}}_{x,y} \mid + \mid z^{*\mathrm{B}}_{x,y} - z^{\mathrm{B}}_{x,y} \mid}{3 \times 255 \times R_{\mathrm{x}} R_{\mathrm{y}}}. \tag{17}$$

## 4. Experiments

The following experiments assess the viability of the approach on different control parameters, each with several independent runs. The parameter sets are as follows: the DE population size $NP = \{25, 50, 100\}$ and $T_{\max} = \{10, 20, ..., 150\}$, thereby for each run $RNi$={0,1,...,51} this counts for total of 45 parameter sets, i.e. 2340 independent runs. The maximum number of function evaluations (MAXFES) used is same as with [5], MAXFES is 1e+5. For image rendering, basic GDI+ is used.

### 4.1 Obtained Results

The obtained fitness values at the MAXFES termination of 1e+5, over different parameters of $T_{\max}$ and $NP$, are seen in Tables 1 and 2. The best values obtained overall for an image are marked in bold text font. The fitness convergence graphs for these best runs are seen in Figure 2, where after the initialization, the fitness is roughly below 40 (i.e. 40% similarity with reference), then drops below 15 for all test images and even further to slightly above 6 for two of them.

The convergent obtained results depend on the MAXFES used being same as with [5], but also $NP$ and $T_{\max}$, as reported below. From Ta-

*Table 1.* Obtained fitness over $T_{\max}$ and $NP$: test instances `Liberty` and `Palace`.

| | | `Liberty` | | | | `Palace` | | | |
|---|---|---|---|---|---|---|---|---|---|
| $NP$ | $T_{\max}$ | Best | Worst | Average | STD | Best | Worst | Average | STD |
| 25 | 10 | 8.29 | 11.99 | 9.93096 | 0.8233 | 8.69 | 13.69 | 10.1362 | 0.9655 |
| 25 | 20 | 8.03 | 13.14 | 10.0935 | 1.0845 | 7.83 | 11.5 | 9.12173 | 0.8092 |
| 25 | 30 | 8.41 | 13.74 | 10.0525 | 1.1712 | 7.52 | 11.1 | 8.97942 | 0.7992 |
| 25 | 40 | 8.13 | 12.81 | 10.4408 | 1.1416 | 7.34 | 11.36 | 8.91788 | 0.8922 |
| 25 | 50 | 8.49 | 13.37 | 10.6767 | 1.1768 | 7.65 | 12.53 | 8.87442 | 0.9788 |
| 25 | 60 | 7.95 | 14.65 | 10.9858 | 1.4284 | 7.9 | 11.88 | 8.99673 | 0.8761 |
| 25 | 70 | 8.28 | 14.21 | 11.4075 | 1.3630 | 7.79 | 13.17 | 9.50327 | 1.0482 |
| 25 | 80 | 8.72 | 15.89 | 11.7554 | 1.6330 | 7.97 | 12.34 | 9.43558 | 0.9765 |
| 25 | 90 | 8.84 | 16.24 | 12.1342 | 1.6608 | 8.41 | 13.54 | 9.82 | 1.2756 |
| 25 | 100 | 9.01 | 16.74 | 12.4798 | 1.7521 | 8.62 | 12.96 | 9.83635 | 0.8869 |
| 25 | 110 | 8.07 | 16.78 | 12.7412 | 1.7849 | 9.01 | 14.42 | 10.4119 | 1.2468 |
| 25 | 120 | 9.67 | 16.14 | 12.8467 | 1.7359 | 8.93 | 15.13 | 10.3858 | 1.3149 |
| 25 | 130 | 10.16 | 17.96 | 13.2692 | 1.7193 | 9.02 | 14.2 | 10.2858 | 1.0292 |
| 25 | 140 | 9.29 | 17.99 | 13.7029 | 1.7886 | 8.29 | 13.51 | 10.7779 | 1.0299 |
| 25 | 150 | 10.82 | 18.56 | 14.0373 | 1.6573 | 9.89 | 14.91 | 11.1206 | 1.0586 |
| 50 | 10 | 7.51 | 9.69 | 8.45077 | 0.4198 | 7.43 | 11.84 | 8.68058 | 0.8825 |
| 50 | 20 | 6.78 | 8.99 | 7.80173 | 0.4987 | **7.1** | 11.39 | 8.79173 | 0.9592 |
| 50 | 30 | 6.89 | 9.17 | 7.81788 | 0.5119 | 7.53 | 12.58 | 9.75654 | 1.1186 |
| 50 | 40 | **6.77** | 9.87 | 8.0375 | 0.6578 | 8.27 | 12.24 | 10.0575 | 0.9537 |
| 50 | 50 | 7.08 | 10.61 | 8.39923 | 0.7056 | 7.97 | 13.14 | 10.3338 | 1.1009 |
| 50 | 60 | 7.15 | 10.4 | 8.67115 | 0.7472 | 8.59 | 12.49 | 10.7817 | 1.0754 |
| 50 | 70 | 7.46 | 10.9 | 9.1025 | 0.8666 | 7.58 | 12.8 | 10.7744 | 1.1086 |
| 50 | 80 | 7.6 | 11.4 | 9.47981 | 0.8689 | 9.15 | 13.11 | 11.3802 | 1.0178 |
| 50 | 90 | 8.05 | 12.65 | 9.67346 | 0.9115 | 9.97 | 13.41 | 11.5227 | 0.9315 |
| 50 | 100 | 8.75 | 11.75 | 10.0152 | 0.7824 | 8.55 | 13.62 | 11.4356 | 0.9923 |
| 50 | 110 | 8.93 | 13.63 | 10.6356 | 0.9682 | 9.32 | 13.77 | 12.0712 | 0.9579 |
| 50 | 120 | 9.22 | 13.01 | 10.7502 | 0.9840 | 9.77 | 14.21 | 12.429 | 0.8972 |
| 50 | 130 | 9.42 | 12.59 | 11.0527 | 0.7707 | 11.37 | 14.07 | 12.7387 | 0.6134 |
| 50 | 140 | 9.99 | 13.39 | 11.5719 | 0.7815 | 9.69 | 15.5 | 12.9317 | 0.9708 |
| 50 | 150 | 10.2 | 14.56 | 12.2633 | 1.0702 | 9.58 | 15.36 | 12.8092 | 1.1717 |
| 100 | 10 | 7.1 | 9.12 | 7.98596 | 0.4241 | 7.91 | 13.88 | 10.9573 | 1.8019 |
| 100 | 20 | 6.85 | 9.77 | 7.83962 | 0.5360 | 8.86 | 14.59 | 12.1117 | 1.2862 |
| 100 | 30 | 7.15 | 11.8 | 8.49077 | 1.1563 | 9.59 | 16.15 | 12.9098 | 1.0589 |
| 100 | 40 | 7.22 | 13 | 8.86327 | 1.1092 | 9.65 | 14.97 | 13.2477 | 1.1543 |
| 100 | 50 | 7.41 | 12.75 | 9.34846 | 1.3939 | 11.01 | 15.52 | 13.8606 | 0.9750 |
| 100 | 60 | 8.06 | 12.97 | 9.77731 | 1.1539 | 11.5 | 16.14 | 14.1856 | 1.1234 |
| 100 | 70 | 8.67 | 13.28 | 10.1954 | 1.3722 | 10.77 | 16.32 | 14.3629 | 1.1713 |
| 100 | 80 | 8.73 | 14.48 | 11.0929 | 1.4093 | 10.98 | 17.06 | 14.9348 | 1.1679 |
| 100 | 90 | 9.04 | 14.92 | 11.3594 | 1.3483 | 11.1 | 16.8 | 15.104 | 1.2586 |
| 100 | 100 | 9.4 | 16.13 | 11.6604 | 1.4952 | 10.8 | 17.62 | 15.36 | 1.2330 |
| 100 | 110 | 10.17 | 15.68 | 12.3365 | 1.5685 | 13.01 | 17.86 | 16.0202 | 0.9744 |
| 100 | 120 | 10.26 | 15.45 | 12.3358 | 1.5076 | 11.07 | 17.99 | 15.6113 | 1.6455 |
| 100 | 130 | 10.22 | 16.19 | 13.2212 | 1.6108 | 12.33 | 18.37 | 16.4085 | 1.3168 |
| 100 | 140 | 11.42 | 16.65 | 13.7808 | 1.5502 | 11.64 | 18.35 | 16.1229 | 1.4990 |
| 100 | 150 | 11.35 | 18.68 | 14.6113 | 1.9726 | 10.11 | 18.34 | 16.2929 | 2.0056 |

*Table 2.* Obtained fitness over $T_{\max}$ and $NP$: test instances `Vegetables` and `Baboon`.

| | | `Vegetables` | | | | `Baboon` | | | |
|---|---|---|---|---|---|---|---|---|---|
| $NP$ | $T_{\max}$ | Best | Worst | Average | STD | Best | Worst | Average | STD |
| 25 | 10 | 14.13 | 17.21 | 15.7269 | 0.7148 | 15.02 | 18.59 | 16.38 | 0.7128 |
| 25 | 20 | 12.56 | 18.03 | 14.5658 | 0.9850 | 13.44 | 17.12 | 15.3815 | 0.8129 |
| 25 | 30 | 12.33 | 15.98 | 13.9215 | 0.8475 | 12.99 | 19.03 | 15.0204 | 1.1150 |
| 25 | 40 | 11.62 | 16.21 | 13.674 | 1.0436 | 11.99 | 16.85 | 14.4342 | 1.0135 |
| 25 | 50 | 12.16 | 17.08 | 13.88 | 1.0726 | 11.39 | 17.62 | 14.4573 | 1.2299 |
| 25 | 60 | 11.64 | 17.88 | 13.6438 | 1.2155 | 11.74 | 17.51 | 14.8038 | 1.2229 |
| 25 | 70 | 11.29 | 17.15 | 13.9056 | 1.3790 | 11.88 | 17.9 | 14.6267 | 1.3495 |
| 25 | 80 | 11.61 | 16.6 | 14.0871 | 1.3881 | 12.11 | 17.13 | 14.3606 | 1.2815 |
| 25 | 90 | 11.63 | 17.96 | 14.1062 | 1.4428 | 11.93 | 19.41 | 14.6644 | 1.5269 |
| 25 | 100 | 11.34 | 17 | 14.4533 | 1.4694 | 11.7 | 18.77 | 14.7642 | 1.7438 |
| 25 | 110 | 11.74 | 19.66 | 14.6085 | 1.7664 | 12.02 | 19.11 | 15.0046 | 1.7605 |
| 25 | 120 | 12.26 | 17.91 | 14.7737 | 1.5726 | 12.2 | 18.5 | 15.6467 | 1.6086 |
| 25 | 130 | 12.1 | 19.75 | 14.6338 | 1.9283 | 13.01 | 19.5 | 15.4254 | 1.5505 |
| 25 | 140 | 11.94 | 19.01 | 14.7635 | 1.6282 | 12.64 | 19.37 | 15.8235 | 1.8458 |
| 25 | 150 | 12.82 | 18.7 | 14.6487 | 1.3015 | 13.13 | 20.17 | 15.7952 | 1.6923 |
| 50 | 10 | 13.03 | 15 | 14.0723 | 0.4674 | 13.86 | 16.52 | 14.9192 | 0.5494 |
| 50 | 20 | 11.66 | 13.26 | 12.4644 | 0.3184 | 11.8 | 14.54 | 13.271 | 0.5569 |
| 50 | 30 | 11.12 | 13.59 | 12.2425 | 0.6528 | 11.59 | 13.62 | 12.5506 | 0.5732 |
| 50 | 40 | **10.94** | 14.1 | 12.1848 | 0.6656 | 11.1 | 13.84 | 12.3137 | 0.6090 |
| 50 | 50 | 11.04 | 13.92 | 12.2946 | 0.7609 | 11.34 | 14.36 | 12.4075 | 0.6304 |
| 50 | 60 | 11.29 | 15.86 | 12.5506 | 0.9222 | 11.25 | 14.1 | 12.3662 | 0.6161 |
| 50 | 70 | 11.18 | 15.21 | 12.6104 | 0.8682 | 11.54 | 14.57 | 12.5437 | 0.6510 |
| 50 | 80 | 11.32 | 15.26 | 12.8619 | 0.7658 | **11.07** | 15.56 | 12.9473 | 0.8087 |
| 50 | 90 | 11.84 | 15.28 | 13.0077 | 0.8038 | 11.32 | 16.2 | 12.857 | 1.0291 |
| 50 | 100 | 11.72 | 15.8 | 13.5058 | 0.9565 | 11.85 | 15.72 | 13.2658 | 0.7972 |
| 50 | 110 | 12.02 | 15.92 | 13.5204 | 0.8750 | 11.98 | 15.56 | 13.4275 | 0.7805 |
| 50 | 120 | 11.9 | 16.87 | 13.829 | 1.1151 | 12.43 | 15.66 | 13.5106 | 0.7265 |
| 50 | 130 | 12.51 | 15.97 | 14.094 | 0.8855 | 12.64 | 16.32 | 14.085 | 0.8259 |
| 50 | 140 | 12.16 | 17.07 | 14.8198 | 1.2154 | 12.54 | 16.31 | 14.15 | 0.8865 |
| 50 | 150 | 13.11 | 17.98 | 14.9838 | 1.2072 | 13.08 | 18 | 14.8765 | 1.0178 |
| 100 | 10 | 12.56 | 16.19 | 13.9815 | 0.8083 | 13.49 | 16.19 | 14.5367 | 0.5672 |
| 100 | 20 | 11.84 | 16.45 | 13.4704 | 1.0483 | 12.02 | 15.87 | 13.8244 | 0.8747 |
| 100 | 30 | 11.83 | 17.64 | 13.9133 | 1.3335 | 12 | 15.76 | 13.7206 | 0.9727 |
| 100 | 40 | 12.01 | 17.95 | 14.6354 | 1.3660 | 11.63 | 17.01 | 13.6467 | 1.3582 |
| 100 | 50 | 11.87 | 17.35 | 14.9156 | 1.4272 | 11.99 | 17.48 | 14.1658 | 1.5554 |
| 100 | 60 | 12.32 | 18 | 15.21 | 1.5119 | 12.12 | 17.46 | 14.5021 | 1.4517 |
| 100 | 70 | 12.13 | 18.05 | 15.6513 | 1.2457 | 12.12 | 17.16 | 14.3881 | 1.3782 |
| 100 | 80 | 12.9 | 18.86 | 16.2008 | 1.4121 | 12.13 | 17.56 | 14.8656 | 1.4214 |
| 100 | 90 | 12.32 | 20.04 | 16.3233 | 1.7789 | 12.25 | 18.66 | 15.2558 | 1.5144 |
| 100 | 100 | 12.98 | 20.55 | 16.7275 | 1.7119 | 13.09 | 18.42 | 15.5398 | 1.5064 |
| 100 | 110 | 13.76 | 20.18 | 17.2896 | 1.5242 | 13 | 19.62 | 15.84 | 1.6164 |
| 100 | 120 | 13.12 | 20.62 | 17.626 | 1.5807 | 13.34 | 19.58 | 16.4725 | 1.5223 |
| 100 | 130 | 13.52 | 20.12 | 17.9052 | 1.3516 | 13.84 | 19.6 | 16.9367 | 1.7362 |
| 100 | 140 | 14.08 | 20.52 | 18.216 | 1.6975 | 14.3 | 21 | 17.4387 | 1.7372 |
| 100 | 150 | 14.97 | 21.19 | 19.1221 | 1.2128 | 14.75 | 21.13 | 17.9488 | 1.6872 |

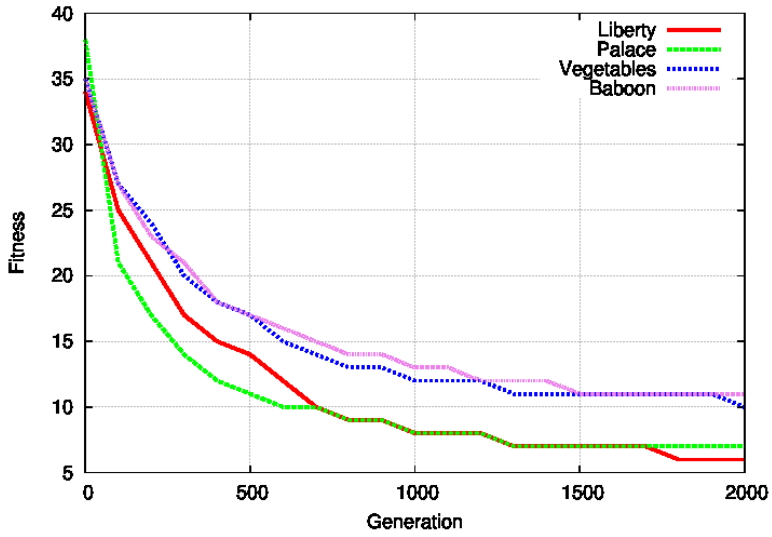



*Figure 2.* Fitness convergence during optimization, for best runs of each test image.

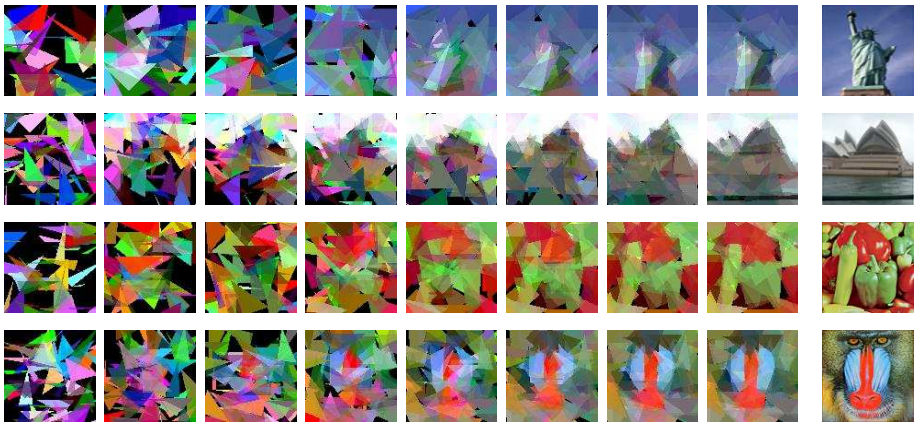


*Figure 3.* The evolved and the reference images.

bles 1 and 2, we choose to report further evolved images upto MAXFES of 1e+6 with all images. The best approximated images after MAXFES of 1e+6 are shown in the Figure 3 which shows the evolution of the four images. In each line of Figure 3, the best fitting vectors upto MAXFES of 1e+6 in generations $g = \{0, 100, 200, 400, 700, 1200, 2000\}$, and the final generation, are shown, then the rightmost the corresponding reference image. Figure 4 shows for each test image, dynamics of the number of triangle brushes in current best vector during generations, displaying varying convergent best $T_i$ values across images.

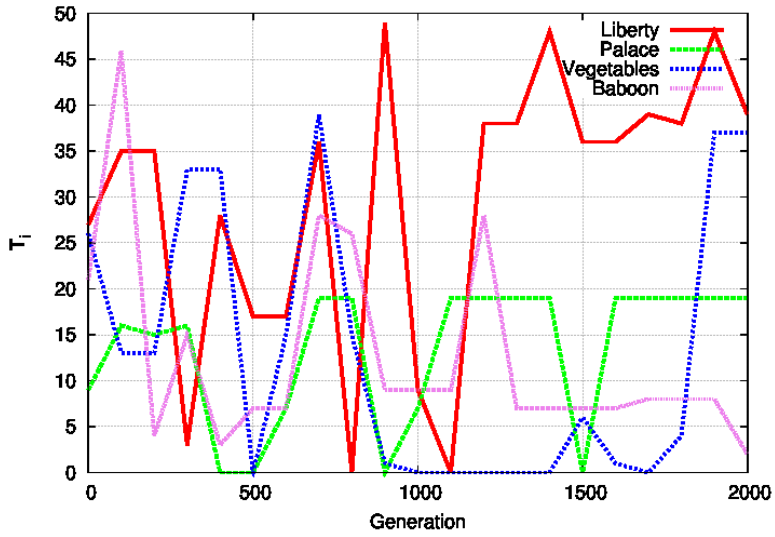



*Figure 4.* Number of best vector brushstrokes, for best runs of each test image.

Our approach searches for a representative image model and the values obtained such as 6.77, can roughly be compared to the 4.83 of [5]. Such representation of the problem also makes our $NP$ parameter have higher value, since we have no guided search and the problem is therefore more general. Also, our approach does not use a dynamically re-allocatable morphable variable-size tree structure as in genetic programming encoding, inspite it rather uses a fixed size vector and limits its brushstrokes set by two simple bounds, making the approach faster for execution.

## 5. Conclusion

This paper presents an evolvable lossy image representation, approximating an image by comparing it to its evolved generated counterpart image. The image is represented using a variable number of triangular brushstrokes, each consisting of a triangle position and color parameters. These parameters for each triangle brush are evolved using differential evolution, which self-adapts the control parameters for mutation and crossover. Also, the proposed DE extension splits the DE vector in the codon and anticodon parts, where the triangles material is used only from the codon part, adjusting the genetic tree center and its borders, together with the number of triangle brushstrokes to be rendered. Experimental results show the viability of the proposed encoding and evolution convergence for the lossy representation of reference images, where fitness is displayed dependent on the population size, maximal number of

function evaluations allowed, maximal number of triangles used in image representation, and different input reference images. Future work can include addressing different encoding aspects, evolutionary operators, control-parameters update, Euclidean distance for colors comparison, and more case studies on input images with different properties.

## References

[1] B. D. Ackland and N. H. Weste. The edge flag algorithm — a fill method for raster scan displays. *IEEE T. Comput.*, 30(1):41–48, 1981.

[2] P. Barile, V. Ciesielski, M. Berry, and K. Trist. Animated drawings rendered by genetic programming. In *Proc. 11th Annual Conference on Genetic and Evolutionary Computation (GECO)*, pages 939–946, 2009.

[3] 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.

[4] S. Das and P. N. Suganthan. Differential Evolution: A Survey of the State-of-the-art. *IEEE T. Evolut. Comput.*, 15(1):4–31, 2011.

[5] A. Izadi, V. Ciesielski, and M. Berry. Evolutionary non photo–realistic animations with triangular brushstrokes. *Lect. Notes Artif. Intell.*, 6464:283–292, 2010.

[6] F. Neri and V. Tirronen. Recent Advances in Differential Evolution: A Survey and Experimental Analysis. *Artif. Intell. Rev.*, 33(1-2):61–106, 2010.

[7] L. Quan. *Image-Based Modeling*. Springer, 2010.

[8] S. Rahnamayan and H. R. Tizhoosh. Image thresholding using micro opposition-based Differential Evolution (Micro-ODE). In *Proc. IEEE World Congress on Computational Intelligence (WCCI)*, pages 1409–1416, 2008.

[9] J. Riley and V. Ciesielski. Fitness landscape analysis for evolutionary non-photorealistic rendering. In *Proc. IEEE Congress on Evolutionary Computation (CEC)*, pages 1–9, 2010.

[10] 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.

[11] K. Trist, V. Ciesielski, and P. Barile, P. Can't see the forest: Using an evolutionary algorithm to produce an animated artwork. In *Arts and Technology*, Springer, Berlin, Heidelberg, 2010, pages 255–262.

[12] A. Zamuda and J. Brest. Vectorized procedural models for animated trees reconstruction using differential evolution. *Information Sciences*, 278:1–21. 2014.

[13] A. Zamuda, J. Brest, B. Bošković, and V. Žumer. Differential Evolution for Parameterized Procedural Woody Plant Models Reconstruction. *Appl. Soft Comput.*, 11(8):4904–4912, 2011.

[14] Y. Zhong and L. Zhang. Remote sensing image subpixel mapping based on adaptive differential evolution. *IEEE T. Syst. Man Cy. B*, 42(5):1306–1329, 2012.

# EXTENDED FINITE-STATE MACHINE INFERENCE WITH PARALLEL ANT COLONY BASED ALGORITHMS

Daniil Chivilikhin
*Computer Technologies Laboratory*
*St. Petersburg National Research University of Information Technologies, Mechanics and Optics, Saint-Petersburg, Russian Federation*
chivdan@rain.ifmo.ru

Vladimir Ulyantsev
*Computer Technologies Laboratory*
*St. Petersburg National Research University of Information Technologies, Mechanics and Optics, Saint-Petersburg, Russian Federation*
ulyantsev@rain.ifmo.ru

Anatoly Shalyto
*Computer Technologies Laboratory*
*St. Petersburg National Research University of Information Technologies, Mechanics and Optics, Saint-Petersburg, Russian Federation*
shalyto@mail.ifmo.ru

**Abstract** This paper addresses the problem of inferring extended finite-state machines (EFSMs) with parallel algorithms. We propose a number of parallel versions of a recent EFSM inference algorithm MuACO. Two of the proposed algorithms demonstrate super-linear speedup.

**Keywords:** Extended finite-state machine, Induction, Learning, Parallel algorithm, Synthesis learning

## 1. Introduction

The automata-based programming paradigm [6] allows to infer the logical structure of software automatically using behavior examples and temporal properties [8]. This logical structure is expressed in the form of

an extended finite-state machine (EFSM), which is quite appropriate for reactive (event-based) systems [9]. The EFSM acts as a controller in an *automated-controlled object*, which consists of an EFSM and a controlled object (software system). The EFSM receives input events from *event suppliers*, and sends commands to the controlled object, which processes them and makes corresponding actions (e.g. calls its methods). By using search-based optimization such as evolutionary algorithms [2] to automatically infer EFSMs from examples of desired behavior, and also by incorporating temporal properties that the target program should satisfy, we can get correct-by-design control programs automatically. This is in contrast with the traditional approach when the system is first designed and implemented and only then it is tested and verified.

The main issue in this scheme is that the problem of inferring EFSMs from behavior examples is extremely computationally hard. Even sophisticated algorithms require a long time to find EFSMs in quite simple cases [8]. Therefore, in order to bring automata-based programming closer to being practically applicable in industry, efficient parallel EFSM inference algorithms are needed.

An example of early work in parallel algorithms for FSM inference is [7], where a parallel genetic algorithm (GA) was used to synthesize FSMs from input/output sequences. Recent publications on EFSM inference such as [8] and [3] do not study parallel implementations.

In this work we study parallelization schemes for the Mutation-Based Ant Colony Optimization (MuACO) algorithm proposed in [3], which proved to be more efficient than GA for the problem of FSM inference. A number of parallel versions of MuACO are presented and experimentally evaluated on a shared-memory multi-core machine. Some of the presented parallel algorithms demonstrate super-linear speedup.

The rest of this paper is structured as follows. Section 2 gives an overview of the MuACO algorithm. In Section 3 the EFSM inference problem is described. Section 4 describes the proposed parallel algorithms. Experimental results are presented in Section 5 and Section 6 concludes.

## 2. MuACO: Mutation-Based Ant Colony Optimization

In this section we briefly describe MuACO. We only discuss details that are essential to understand what is done in this paper. For a full description of the algorithm please refer to [3].

MuACO works with a special graph called a *search graph*, where nodes correspond to FSMs and edges correspond to EFSM mutations. Each

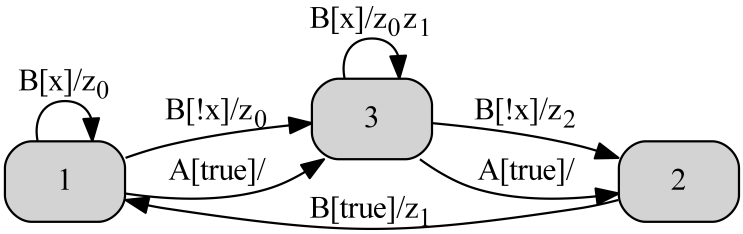



*Figure 1.* An example of an extended finite-state machine with three states. Each edge is marked with an input event and Boolean formula over the input variables (before the slash) and a sequence of output actions (after the slash).

search graph edge has associated pheromone and heuristic information values. The algorithm uses two types of EFSM mutation operators. The first mutation operator selects a random transition in the EFSM and changes its destination state to a new state, which is selected uniformly and randomly from the set of all states, excluding the old destination state of the transition. The second mutation operator with a certain probability modifies the set of transitions in each state. It can either delete a transition from a state, or add a new random transition to the state.

The algorithm starts off with a randomly generated solution (EFSM). Then, on each *colony iteration* a number of *ants* are launched to search for solutions. Each ant starts with the node of the search graph associated with the best-so-far solution. The ant has a limited number of steps. On each step it either:

- creates a number of new solutions by mutating its current solution and moves to the one having the largest fitness function value;
- or probabilistically selects the next node from the existing successors of the current node.

When all ants have finished searching for solutions, pheromone values are updated for all graph edges. If for a fixed number of colony iterations the best fitness value does not increase, the algorithm is restarted. In the original MuACO the algorithm was restarted from a randomly generated solution. However, in this work we found that it is more efficient to take the best-so-far solution as the initial one, manually decreasing its fitness value to prevent immediate stagnation.

## 3. Extended Finite-State Machine Inference

An *extended finite-state machine* is a septuple $\langle S, s_0, \Sigma, \Delta, Z, \delta, \lambda \rangle$, where $S$ is a set of states, $s_0 \in S$ is the initial state, $\Sigma$ is a set of

input events and $\Delta$ is a set of output actions. $Z$ is a set of Boolean *input variables*, $\delta\colon S \times \Sigma \times 2^Z \to S$ is the *transitions function* and $\lambda\colon S \times \Sigma \times 2^Z \to \Delta^*$ is the *actions function*. EFSMs in the inference algorithms are encoded in the form of full transition tables, where each cell corresponds to a transition in a certain state triggered by an event.

In the so-called specification-based EFSM inference the user provides some behavior samples that the target program should demonstrate. For instance, one could provide test examples [8], test scenarios [10] or negative scenarios. The user may also specify temporal properties that the EFSM should satisfy. In this work we use test scenarios and temporal properties expressed in Linear Temporal Logic (LTL).

A *test scenario* is a sequence of triples $\langle e, \varphi, O\rangle$ called *scenario elements*, where $e$ is an event, $\varphi$ is a Boolean formula over the input variables and $O$ is a sequence of output actions. An EFSM is said to be compliant with a scenario element in state $s$ if it has a transition from this state marked with $e$, $\varphi$ and $O$. Correspondingly, an EFSM is compliant with a test scenario if it is compliant with all scenario elements in the corresponding states when scenario elements are processed sequentially. For example, the EFSM in Figure 1 is compliant with scenario $\langle B, x, (z_0)\rangle\langle B, x, (z_0)\rangle\langle A, \text{true}, ()\rangle$ and is not compliant with scenario $\langle B, x, (z_0)\rangle\langle A, \text{true}, (z_1)\rangle$.

A LTL formula consists of problem-specific propositional variables **Prop**, Boolean logical operators, and a set of temporal operators, such as **G** (**G**lobally in the future), **X** (ne**X**t), **F** (in the **F**uture), etc. For the case of LTL formulae expressing properties of EFSMs, the set of propositional variables can include terms:

- $\forall e \in \Sigma$ : **wasEvent**$(e)$ – a transition marked with event $e$ has been triggered;

- $\forall a \in \Delta$ : **wasAction**$(a)$ – a transition marked with action $a$ has been triggered.

The problem is therefore to find an EFSM with a given number of states that is compliant with all given test scenarios and LTL formulae. This problem was previously tackled in [8] with a genetic algorithm. We adopt a fitness function $f$ proposed in that work to evaluate the degree of an EFSM's compliance with test scenarios and LTL formulae. The fitness function is based on string edit distance [4] and a specially developed EFSM verifier.

## 4. Parallel MuACO Algorithms

In this paper we propose a number of parallel versions of MuACO. These were implemented on a shared-memory machine but can be straightforwardly extended to work on a cluster of separate machines. All presented approaches implement the coarse-grained parallelization scheme, i.e. when the interaction between algorithms is rare.

### 4.1 Independent Parallel MuACO

In the first and simplest algorithm we call the Independent Parallel MuACO, a number of algorithm instances is launched in parallel for the same problem. That is, having $m$ processors, $m$ MuACO algorithms are launched in parallel, each from a separate randomly generated initial solution. An important detail is to provide each algorithm with a separate random number generator.

The motivation to this simple approach is that the problems MuACO deals with are combinatorial ones. In most cases a single fitness computation does not take much CPU time, however many solutions have to be evaluated in order to find the optimal one. By using a number of parallel independent search algorithms we increase the amount of different solutions explored during the same time period, thus, possibly, decreasing the amount of time needed to find the optimal one.

### 4.2 Parallel MuACO with Shared Best Solutions

The second step is to see if adding some interaction between independent MuACO instances can increase performance. The conventional interaction scheme in parallel evolutionary algorithms is to migrate some solutions between individual populations of algorithms running in parallel (see, for example, [1]). However, due to the nature of MuACO, there is no clear way of adopting this scheme in our case, since MuACO does not keep a population of solutions.

Therefore, one of the easiest ways of adding interaction is to keep a cache of best solutions found by each algorithm and share it among them. Suppose with have $m$ MuACO algorithms running in parallel. Let $K$ be an array of length $m$ for storing best found solutions, with $K_i$ reserved for the $i$-th algorithm (Figure 2). Each $i$-th algorithm updates the value of $K_i$ each time it finds a solution better than the one stored there. When a sequential MuACO comes to a stagnation state, it is restarted from its best found solution as described in Section 2. In this parallel variant the stagnated $i$-th algorithm will be restarted, taking

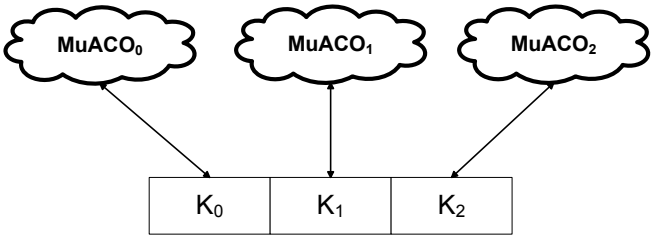



*Figure 2.* Parallel MuACO with shared best solutions.

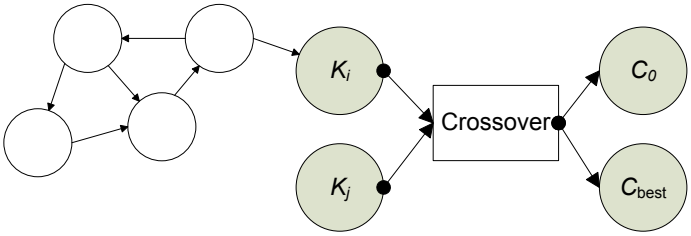



*Figure 3.* Parallel MuACO with crossover and shared best solutions: the best solution of the $i$-th thread $K_i$ is crossed over with a remote thread best solution $K_j$. White nodes at the left represent a part of the $i$-th algorithm's search graph.

$K_j$ as the initial solution, where $j$ is selected uniformly randomly from $\{0 \dots i-1, i+1 \dots m-1\}$.

## 4.3 Parallel MuACO with Crossover

As the algorithm described above, this third and final version of parallel MuACO uses a cache of shared best solutions, but also adopts a crossover operator as in genetic algorithms.

On each colony iteration of the $i$-th MuACO (before ants are launched) a random solution $K_j$ ($j \neq i$) is picked from the cache of shared best solutions $K$. A crossover operator is applied to this solution $K_j$ from a remote algorithm and the best solution $K_i$ found by the $i$-th algorithm. The offspring produced by the crossover operator are then evaluated with the fitness function $f$, and only one child solution $C_{\text{best}}$ with the best fitness value is left (Figure 3).

This solution $C_{\text{best}}$ is added to the search graph of the $i$-th algorithm as a child of the best-so-far solution. In the consequent colony iteration one ant will be launched from $C_{\text{best}}$, and the rest of the ants will be launched, as before, from the node associated with the best-so-far solution.

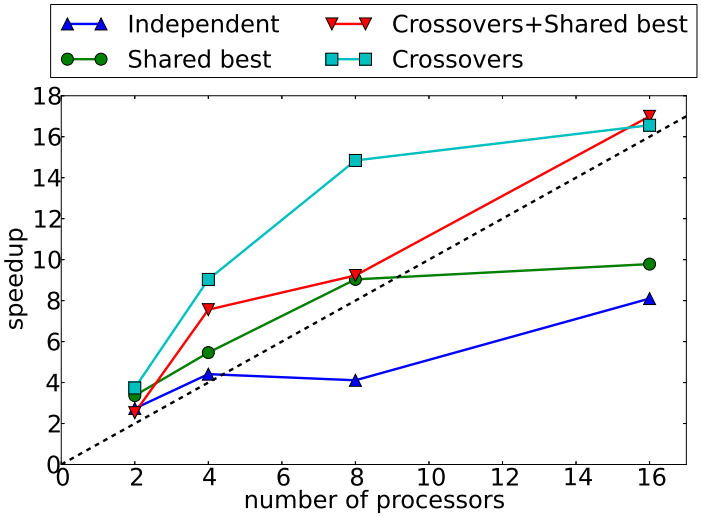



*Figure 4.* Speedup plots for different parallel MuACO algorithms.

Of course, the crossover operator applied here is problem-specific. Here we studied two crossover operators used in EFSM inference. The first crossover operator randomly mixes the transition tables of the two parent EFSMs and is thus called "Simple". The second crossover operator called "Test-based" proposed in [8] takes into account the transitions of the parent EFSMs the were used during fitness evaluation. The type of crossover is selected uniformly and randomly each time it has to be applied.

## 5. Experiments

Prior to conducting experiments the parameter values of sequential MuACO were automatically selected using the *irace* package [5]. This package implements a racing protocol and allows to derive appropriate parameter values that allow the algorithm to perform efficiently on a certain class of problem instances.

All algorithms were run on 50 randomly generated problem instances. Each of them was derived from a random EFSM with 10 states. For each problem instance each algorithm was run until finding an EFSM that satisfies all test scenarios and LTL formulae. To calculate speedup we measured the average wall clock running time of the sequential algorithm and divided it by the average running time of a particular parallel MuACO algorithm.

The following parallel algorithms were compared:

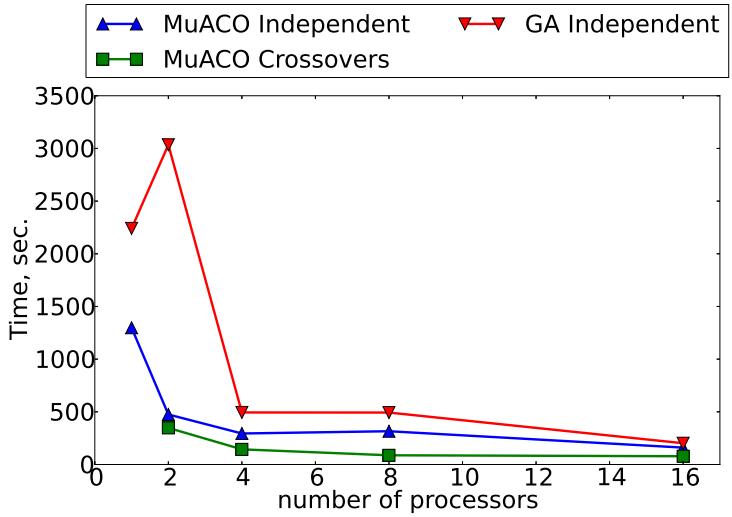



*Figure 5.* Execution time of independent and crossover parallel MuACO in comparison to independent parallel GA.

- independent parallel MuACO ("Independent");
- parallel MuACO with shared best solutions ("Shared best");
- parallel MuACO with crossovers, each algorithm is restarted from its own best solution ("Crossovers");
- parallel MuACO with crossovers and shared best solutions ("Crossovers + Shared best").

We used a machine with a 24-core AMD Opteron 6234 2.4 GHz processor. Experimental results in the form of speedup plots are shown in Figure 4. The mean runtime of the sequential MuACO was 1392 seconds.

As can be seen from Figure 4, all parallel MuACO algorithms demonstrate an increase of performance as the number of used processors grows. All algorithms are pretty much the same for the case of two processors. The "Independent" algorithm with no interaction is the worst among them, demonstrating a speedup of only about 8 for 16 processors. However, theoretically it is very robust, since for this algorithm performance will never decrease with the increase of the processors amount.

The "Shared best" algorithm scales well up until 8 processors, however its performance does not significantly change when 16 processors are used. And only two algorithms that use crossovers demonstrate super-linear speedup all the way, with the "Crossovers" algorithm be-

ing on average slightly better than "Crossovers+Shared best" for 2–8 processors.

The plot in Figure 4, however, only displays speedup measured from mean execution time of the algorithms. To verify the statistical significance of the differences between parallel algorithm performance, we used the Wilcoxon statistical test [11]. The test was applied to each pair of algorithms, comparing the distributions of running time for the case of 16 processors. The results of statistical tests are presented in Table 1.

As expected, the performance of all parallel algorithms (2–5) significantly differs from the performance of the single-thread MuACO (1), which is indicated by small $p$-values in the first row of the table. Algorithms with crossover (4) and (5) are significantly better than independent parallel MuACO (2). For the case of 16 processors, algorithms with crossover yield similar running time ($p$-value = 0.939).

We also implemented an independent parallel version of the GA from [8]. GA parameters were also tuned using *irace*. Plots showing mean execution time of this GA in comparison with independent and crossover parallel MuACO are shown in Figure 5. It can be seen that the independent GA is always slower than independent MuACO, which is in turn always slower than crossover MuACO. The independent GA is 2–9 times slower than crossover MuACO.

*Table 1.* Wilcoxon test $p$-values for algorithms Single (1), Independent (2), Shared best (3), Crossovers (4), Crossovers+Shared best (5).

| | (1) | (2) | (3) | (4) | (5) |
|---|---|---|---|---|---|
| (1) | – | $1.602 \times 10^{-8}$ | $1.946 \times 10^{-9}$ | $6.581 \times 10^{-14}$ | $4.067 \times 10^{-14}$ |
| (2) | – | – | 0.8582 | 0.103 | 0.128 |
| (3) | – | – | – | 0.026 | 0.025 |
| (4) | – | – | – | – | 0.939 |
| (5) | – | – | – | – | – |

## 6. Conclusion and Future Work

We have proposed several parallel versions of the MuACO EFSM inference algorithm. It has been shown that the use of crossover in parallel MuACO significanly improves performance. Parallel MuACO algorithms with crossover demostrated super-linear speedup.

Future work includes creating a hybrid GA–MuACO parallel algorithm where solutions will be exchanged between GA and MuACO instances.

## Acknowledgements

This work was financially supported by the Government of Russian Federation, Grant 074-U01, and also partially supported by RFBR, research project No. 14-01-00551 a.

## References

[1] E. Alba. Parallel evolutionary algorithms can achieve super-linear performance. *Information Processing Letters*, 82(1):7–13, 2002.

[2] T. Back, D. B. Fogel, and Z. Michalewicz. *Handbook of Evolutionary Computation. 1st edition.* IOP Publishing Ltd., 1997.

[3] D. Chivilikhin and V. Ulyantsev. MuACOsm: a new mutation-based ant colony optimization algorithm for learning finite-state machines. In *Proc. 15th Annual Conference Companion on Genetic and Evolutionary Computation (GECCO)*, pages 511–518, 2013.

[4] V. Levenshtein. Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics Doklady*, 10:707, 1966.

[5] M. López-Ibáñez, J. Dubois-Lacoste, T. Stützle, and M. Birattari. The `irace` package, iterated race for automatic algorithm configuration. Technical Report TR/IRIDIA/2011-004, IRIDIA, Université Libre de Bruxelles, Belgium, 2011.

[6] A. A. Shalyto. Software automation design: algoritmization and programming of problems of logical control. *J. Comput. Syst. Sc. Int.*, 6:899–916, 2000.

[7] S. Tongchim and P. Chongstitvatana. Parallel genetic algorithm for finite-state machine synthesis from input/output sequences. In *Proc. Conference on Genetic and Evolutionary Computation (GECCO)*, pages 20–24, 2000.

[8] F. Tsarev and K. Egorov. Finite state machine induction using genetic algorithm based on testing and model checking. In *Proc. 13th Annual Conference on Genetic and Evolutionary Computation (GECCO)*, pages 759–762, 2011.

[9] D. Harel and M. Politi. *Modeling Reactive Systems with Statecharts: The Statemate Approach. 1st edition* McGraw-Hill, 1998.

[10] V. Ulyantsev and F. Tsarev. Extended finite-state machine induction using SAT-solver. In *Proc. 10th International Conference on Machine Learning and Applications and Workshops (ICMLA)*, pages 346–349, 2011.

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

# EMPIRICAL CONVERGENCE ANALYSIS OF GENETIC ALGORITHM FOR SOLVING UNIT COMMITMENT PROBLEM

Domen Butala
*Financial Mathematics, Faculty of Mathematics and Physics*
*University of Ljubljana, Slovenia*
domen.butala@yahoo.com

Dejan Velušček
*Department of Mathematics, Faculty of Mathematics and Physics*
*University of Ljubljana, Slovenia*
dejan.veluscek@fmf.uni-lj.si

Gregor Papa
*Computer Systems Department*
*Jožef Stefan Institute, Ljubljana, Slovenia*
*and*
*Jožef Stefan International Postgraduate School, Ljubljana, Slovenia*
gregor.papa@ijs.si

**Abstract** This paper presents an implementation and empirical convergence analysis results of genetic algorithm for solving unit commitment problem in a power market. Various parameter settings are presented including an algorithm with a sequence of parameters, also called a variable-structure genetic algorithm. Implemented algorithm successfully solves both small and large scale problems and shows how much more efficient variable-structure genetic algorithm is in practice.

**Keywords:** Convergence analysis, Genetic algorithm, Empirical results, Unit commitment problem.

## 1. Introduction

In a power market total production has to meet the demand, net exports and system reserves over a given period of time, subject to the start-up and shut-down times of the generating units. The objective is to minimize the total costs of production while satisfying the start-up and shut-down time constraints. In reality power generating units are not only minimizing costs but also maximizing their profit.

The solution of a unit commitment is a complex optimization problem. It is one of the most widely studied problems in Electrical Engineering. A number of different techniques have been proposed to solve it as Mixed Integer Programming (MIP) [4] or an alternative Mixed Integer Linear Programming (MILP) [1]. Also Lagrangian Relaxation (LR) [4], Benders Decomposition [10], Evolutionary Programming (EP) [7] and Dynamic Programming (DP) are used.

Power generating units can reschedule their commitments (decisions) over and over again. They can do a reschedule when they believe that some parameters have changed enough to affect their decisions. This means that power plants can sell the energy in a futures market, later buy it back and finally commit their schedules on a day-ahead (or often called spot) power auction. If the market price of a certain product in futures market is higher than the expected spot price, power plants with lower marginal prices decide to sell (part of) the energy in advance. When this is not the case, they decide to wait and sell the energy on a day-ahead power auction.

In this paper we will focus not only on the implementation part of the algorithm but also on the convergence analysis. With empirical analysis we will briefly try to justify the relevance of theoretical convergence analysis demonstrated in the section 2.

## 2. Convergence Analysis

In this section we present three important theorems from [5] and [2]. The first theorem tells us something about an upper bound on the convergence speed and the last two tell us under which conditions we can expect a genetic algorithm (GA) to converge. Based on the presented theorems we do an empirical analysis and try to evaluate theoretical results in practice for a problem of unit commitment using the GA in a power market.

### 2.1 An Upper Bound on the Convergence Speed

Next theorem tells us an upper bound for convergence of the GA.

**Theorem 1** *[5] Let the size of population of the GA be $n \geq 1$, coding length $l > 1$, mutation probability $0 < p_m \leq \frac{1}{2}$ and let $\{\vec{X}_t, t \geq 0\}$ be the Markov chain population, $\pi^{(t)}$ distribution of $t^{th}$ generation of $\vec{X}_t$ and $\pi$ be the stationary distribution. Then it holds*

$$||\pi^{(k)} - \pi|| \leq (1 - (2p_m)^{nl})^k.$$

Theorem 1 identifies us next relationships.

- Bigger than the mutation probability, faster the convergence.
- Bigger than the population size and coding length, slower the convergence.

But on the other hand it is well known that algorithms with parameters set like this affect negatively on a long term convergence of the GAs. This was shown by studies [3], [8], [12] but also by many others. In section 2.2 let us take a closer look at two theorems which tells us that in case of a very big population with "small enough" mutation probability the GA converges in probability to a global optimum.

## 2.2 Convergence of Homogenous Algorithm

**Theorem 2** *[2] Let $a, b, c > 0$ be constants and $i$ intensity perturbations of algorithm. If it holds*

$$m > \frac{an + c(n-1)\Delta^{\otimes}}{\min(a, b/2, c\delta)}, \tag{1}$$

*then*

$$\forall x \in S^N : \qquad \lim_{i \to \infty} \lim_{t \to \infty} P([X_t^i] \subset f^* | X_0^i = x) = 1.$$

Theorem 2 tells us that we can, with a big enough population, solve an optimization problem. Additionally we can completely arbitrarily choose the parameters $a$, $b$ and $c$. If we choose very big $c$, then we rule out the importance of $\delta$ because it holds $\min(a, b/2) \leq c\delta$. If it is $\Delta^{\otimes}$ a constant and $N$ becomes very big, we can always successfully tackle with a problem. $\Delta^{\otimes}$ represents the difficulty of adopting new and better solutions. Condition 1 is quite rough regarding the population size limit $m$. It is important that perturbation mechanism lets the process visit the whole space, even though random perturbations could be very small. That is why the role of crossover is not crucial (algorithm without crossover corresponds to the case of $b = \infty$).

## 2.3 Convergence of Inhomogeneous Algorithm

In practice we will not wait for Markov chain to reach the equilibrium state before we would lower the intensity of perturbations. That is why we want to lower the intensity gradually depending on time. At the same time we want to maintain the properties of a limit law. Let us, from now on, assume that $i$ and $t$ are increasing simultaneously. So, let the $i$ be increasing function of time, where it holds $\lim_{t\to\infty} i(t) = \infty$.

That is how Markov chain becomes inhomogeneous and transition probabilities become time dependent. From now on let us define $X_t = X_t^i$. We can define convergence power of the increasing sequence $i(t)$ with $\lambda$, for which holds that

$$i(1)^{-\theta} + i(2)^{-\theta} + \cdots + i(t)^{-\theta} + \ldots$$

converges for $\theta > \lambda$ and diverges for $\theta < \lambda$ where $\lambda \geq 0 \in \mathbb{R}$

With $T_1, \ldots, T_t, \ldots$ we set times of successful visits of the chain $\{X_t\}$ in $S$, e.g. $T_t = \inf\{k : k > T_{t-1}, X_k \in S\}$. Let us focus on behavior of the chain $\{X_{T_t}\}$.

**Theorem 3** *[2]*

1 *That chain* $\{X_{T_t}\}$ *would reach* $f^*$ *after finite number of steps*

$$\forall x \in S \qquad P(\exists U, \forall u \geq U, [X_{T_u}] \subset f^* | X^{(0)} = x) = 1,$$

*convergence power of sequence* $i(u)$ *has to be positive constant from the set of real numbers;* $\theta_1, \theta_2 > 0 \in \mathbb{R}$ *have to exists to*

$$\sum_{u\geq 0} i(u)^{-\theta_1} = \infty \text{ and } \sum_{u\geq 0} i(u)^{-\theta_2} < \infty.$$

2 *"If convergence power* $\lambda$ *of the sequence* $i(t)$ *and the population size* $m$ *suffice the inequalities*

$$an + c(n-1)\Delta^{\otimes} < \lambda < \min(a, b/2, c\delta)m,$$

*then with probability 1 chain* $\{X_{T_t}\}$ *reaches* $f^*$ *after finite number of steps.*

3 *Let exist the constant* $t > 1 \in \mathbb{R}$ *that for* $\forall r \in \mathbb{N}$ *sequences* $i(\lfloor tu \rfloor + r)$ *and* $i(u)$ *are logarithmic equivalent. If convergence power* $\lambda$ *of the sequence* $i(u)$ *and population size* $m$ *suffice the inequalities*

$$an + c(n-1)\Delta^{\otimes} < \min(a, b/2, c\delta)m < \lambda,$$

*then*

$$\forall x \in S \qquad \lim_{t\to\infty} P([X_t] \subset f^* | X_0 = x) = 1.$$

### 2.4 Combination of Both Approaches

To be able to successfully use the results of both approaches we have to slightly correct an algorithm. Let us define a sequence of parameters $\{(n_t, p_m(t)), t \geq\}$ that it holds $n_t < n_{t+1}$ and $p_m(t) > p_m(t+1)$.

Thus, we run the GA with settings $(n_1, p_m(1))$ first. Let us define a solution close to the optimal with $\vec{X}_1(\infty)$. When it is reached, with predefined scheme, we rename the population $\vec{X}_1(\infty)$ to $\vec{X}_2(0)$ and use it as an initial population of the GA with settings $(n_2, p_m(2))$. Using this approach we are increasing the population size and decreasing the mutation probability thus increasing the algorithm's efficiency.

We can call such the GA a variable-structure GA [5]. We should note that the convergence is not a natural property of the GA but it is followed by elitism property in a selection operator [6].

## 3. Implementation and Model Description

In this section we will denote a mathematical formulation of the problem. Based on this we will describe an algorithm for the optimization problem.

### 3.1 Problem formulation

$$\min_{x_{i,t}^{type}} \Big\{ \sum_{t=1}^{T} \sum_{i=1}^{n} (mp_{i,t} x_{i,t}^{type} + \max\{s_{i,t} - s_{i,t-1}, 0\} sc_i) \Big\}$$

$$\sum_{i=1}^{n} x_{i,t}^{type} \geq PDP_t(price), \forall t \tag{2}$$

$$s_{i,t} = \left\{ \begin{array}{ll} 1, & \text{"if } x_{i,t}^{type} > 0, \\ 0, & \text{otherwise.} \end{array} \right\}, \forall t, i \tag{3}$$

$$st_{i,t} = (-1)^{1-s_{i,t}} \sum 1_{\left\{ \substack{I=[t-a,t+b] \wedge a,b \geq 0: \\ s_{i,t}=s_{i,\bar{t}} \forall \bar{t} \in I \wedge s_{i,t-a-1}=s_{i,t+b+1}=1-s_{i,t}} \right\}} \tag{4}$$

$$st_{i,t} \geq tup_i \vee st_{i,t} \leq -tdown_i, \forall t, i \tag{5}$$

$$x_{i,t} = xmax_{i,t} \tag{6}$$

Where $t = 1, \ldots, T$ is an observed time interval, $i = 1, \ldots, n$ is an unit index and *type* is a unit type. Furthermore, $x_{i,t}^{type} \in \mathbb{R}$ is the production of unit $i$ of type *type* in time $t$, $xmin_{i,t}$ is the technical minimum of unit $i$ in time $t$, $xmax_{i,t}$ is the installed capacity of unit $i$ in time $t$, $s_{i,t}$ is a status of unit $i$ in time $t$, $st_{i,t}$ represents a number of working or non-working hours of unit $i$ in time interval "near" $t$, $tup_i$ and $tdown_i$ is the

minimal time interval of unit $i$ in which unit cannot change its status decision due to the technical limitations, $PDP_t$ is a price dependent production, $mp_{i,t}$ a marginal price of unit $i$ in time $t$ and $sc_i$ are the starting costs of unit $i$.

**Goal.** A goal of the optimization is to find a solution with minimal costs to the system. Cost objective function describes costs of the system in a way that it calculates sum of product of production and marginal costs and total start-up costs on the whole time interval.

We have to meet the constraints, especially important are the following two. The first one (2) ensures that needed levels of thermal production are met and the second one (5) ensures that units are not violating their technical constraints.

At this stage it is important to note that the thermal production is dependent on the price and cross border commercial flows and vice-versa. Because both variables are dependent on neighboring markets, determination of both is a very complex problem. Also note that a more representative criterion function of a power generating unit is of a form $ax^2 + bx + c$ but in this case we have to find or know the parameters $a, b, c$. This is not a content of the article, so we will not discuss this problem.

## 3.2 Optimization

Let us write the pseudocode ofthe GA for our problem. This will serve as a starting point for later discussion.

**Algorithm 1** Genetic Algorithm

```
1: t = 0
2: P(t) = SetInitialPopulation(P)
3: Evaluate(P(t))
4: while not EndingCondition() do
5:     t+ = 1
6:     P(t) = Selection(P(t − 1))
7:     P(t) = Crossover(P(t))
8:     P(t) = Mutation(P(t))
9:     Evaluate(P(t))
10: end while
```

**Initial Solution.** For the initial solution we implemented an algorithm named priority list for a problem of unit commitment. Algorithm is deterministic, for this reason it returns the same value for the same

inputs every time. By using this algorithm we get an initial population consisting of only one solution, therefore we have to do a trick to get $n$ different solutions. To differentiate between $n$ initial solutions, we replicate the one as $n$-times and then mutate all except one.

**Mutation.** The idea of a mutation operator is very simple. With predefined probability, an individual from a population is chosen, on which a random change is made. But before we do this, we have to make sure that the change gives a feasible solution.

The operator mutation is built in a way that technical constraints of mutated units are satisfied first; e.g. minimal up or down times. After this step we have to check feasibility of constraint (2). If the solution is feasible we accept it otherwise we save it in the dictionary of rejected mutations. At the end of the mutation we loop through the dictionary and check for feasible solutions. We accept all that are feasible. After the step we empty the dictionary.

**Crossover.** A crossover operator is the most important operator in the GA. The goal is to successfully combine two different solutions to get two better offspring. This operator is the most complex in the implemented algorithm. In the operator mutation we have already realized that some minor corrections have to be done not to reject too many (infeasible) solutions. The operator crossover has to apply similar manipulations.

**One-Point Crossover.** Let us first present the one-point crossover operator. Later we will also present the multi-point crossover which is a partial generalization of this one.

The idea is very simple. First, we randomly determine pairs of populations for the crossover and then we randomly determine a crossover point $t$ which is the same for all units in a pair. Then we cross same the units within a pair of solutions. If needed we have to "correct" an infeasible solution to generate a feasible one.

A very important property of the operator is that it crosses only the same chromosomes (units) under different solutions. For example, in case of crossing two different units within one solution two units with different parameters (minimal up or down time, different installed capacities, maintenance schedules, ...) would be switched. This has to be treated differently, e.g. with another operator which would already be a joint operator of mutation and crossover.

At determination of a crossover point $t$ we have to check if the place is acceptable for all units in the population. If needed we have to shift the

time of crossover for a few time periods to left or to right for each unit. That is how we determine a vector of crossover places $T = (t_1, \ldots, t_n)$ for a pair of solutions. Because we did one additional operation, we avoided the violations of minimum up or down times. Now we just have to check the feasibility under constraint (2). If a solution is feasible and better, we accept it. Otherwise we correct it with probability $p$ and reject it with probability $1 - p$. If we decide to correct it, we have to turn additional units on to meet the constraint (2).

**Multi-Point Crossover.** One-point crossover is a special case of multi-point crossover. Therefore we can use our knowledge about the one-point crossover.

The main difference between both is already described by its name. Using multi-point crossover $m$ we split the coding length to $m + 1$ randomly determined intervals. But crossover operator basically stays the same. Instead of only two intervals $[0, t), [t, T]$ we have $m + 1$ intervals $[0, t_1), [t_1, t_2), \ldots, [t_m, T)$. Doing so we have to provide a feasible solution for the each interval.

The idea is to construct a feasible solution with the dealing of two adjacent intervals at the time and basically translating a multi-point crossover to a one-point crossover. So, we cross the $[0, t_1), [t_1, t_2)$ first, then the $[t_1, t_2), [t_2, t_3)$ until we finally get to the $[t_{m-1}, t_m), [t_m, T)$. With translation to a one-point crossover we get all needed instruments to deal with the problem.

**Selection.** The selection operator is the simplest operator in the implemented algorithm. There is no need to create or find new and better offspring but only to "shake" the population and randomly permute $n$ of the solutions where better solutions have higher probability to be chosen as offspring.

As we have seen in the section 2.4, one of the most important properties of selection operator is the property of elitism. Therefore we have chosen to implement it. Later we will see the algorithm's behavior without taking into account the elitism property.

## 4. Results

In this section we present results of the implemented algorithm of the actual data. To present and study results as unbiased as possible we compare them on the same data set but with different GA settings. The goal is to successfully distinguish between the worst and best solutions. The following parameters

- crossover and mutation probability,
- crossover operator,
- selection operator

are chosen based on the theoretical analysis which we try to confirm with empirical results.

## 4.1 Assumptions

First let us present the assumptions. The most important condition is to meet the demand or in our case needs of the thermal production. Let us call it a price dependent production. We observe two different time periods; length of the first is 72 hours and length of the second is 168 hours. In the first we analyze different settings of mutation and crossover probability, one-point crossover and selection. In the second we compare both, one-point and multi-point, crossover operators under the same settings. The number of power plants which appear in the optimization for specific problem is equal to $N$. Therefore the size of our matrices is equal to $72 \times N$ and $168 \times N$.

Each setting is run independently 30 times due to the stochastic properties of the algorithm. Only one would not suffice in providing unbiased results.

## 4.2 Empirical Analysis

Let us analyze impact of the mutation, crossover and selection of the convergence speed. First we observe a time interval of a length 72 hours and for crossover operator we use only the one-point crossover (OPC) to compare results under different parameters but the same conditions. In this optimization 145 power plants are appearing.

Later we compare one-point and multi-point crossover (MPC) operators. In this case we observe a time interval of a length 168 hours. The reason for this is that the results of the algorithm could be otherwise biased due to the time constraints in the optimization. In this optimization 167 power plants are appearing.

**Comparison of various settings on one point crossover.** In this subsection we analyze different settings of the algorithm and compare the results. Let us present settings $N_i$ for all $i$ in the table below. We will show 6 different settings which have been obtained by empirical experiments. Using the selected parameters we are able to show the impact on the GA's performance under different domains and, the most important, various conditions.

*Table 1.* Parameter settings.

| Parameters / Settings | $N_1$ | $N_2$ | $N_3$ | $N_4$ | $N_5$ | $N_6$ |
|---|---|---|---|---|---|---|
| Iterations | 10,000 | 10,000 | 10,000 | 10,000 | 10,000 | 10,000 |
| Population size | 30 | 30 | 30 | 60 | 30 | 60 |
| Elitism | 4 | 4 | 4 | 4 | 0 | 4 |
| Crossover | OPC | OPC | OPC | OPC | OPC | OPC |
| Crossover Probability | 50% | 75% | 50% | 50% | 50% | vary |
| Mutation Probability | | | | | | |
| - population | 25% | 25% | 40% | 25% | 25% | vary |
| - individual | 20% | 20% | 25% | 20% | 20% | vary |

The mutation probability is actually lower than it seems. We have to take a product of both mutation probabilities. In case of $N_1$ the mutation probability is equal to 10% (which is equal to 40% times 25%). Note that we have used 50% probability to correct infeasible solutions in the crossover operator.

All setting except the setting $N_6$ are constant over time. For the setting $N_6$ we have constructed a sequence of parameter settings to show how a sequence of settings affect the performance under different domains. The idea is to analyze the algorithm's performance which explore the space of feasible solutions more aggressive at the beginning and less later.

Thus let us define the sequence under different domains which we obtained by the empirical results. In the interval $[0, 1000)$ iterations we have set the crossover probability to 75% and mutation probability pair to (45%, 25%). In the next interval $[1000, 2000)$ we have decreased the mutation probability to (25%, 20%). In the interval $[2000, 5000)$ we have decreased crossover probability to 50%. And for the last interval $[5000, 10000)$ we have decreased mutation probability to (25%, 10%).

For a higher transparency let us present only the average values of a criterion function of 30 independent runs.

The most obvious thing we see is that the algorithm with the setting $N_5$ does not converge at all. This confirms that the convergence of the GA is a consequence of the elitism property.

Let us now compare the crossover's probability effect. Thus we limit ourselves to compare the results with parameter settings $N_1$ and $N_2$. We see that the algorithm with the parameter setting $N_2$ is around 20% more efficient at the beginning but after roughly 200 iterations we cannot significantly distinct between the efficiency of both. Due to the

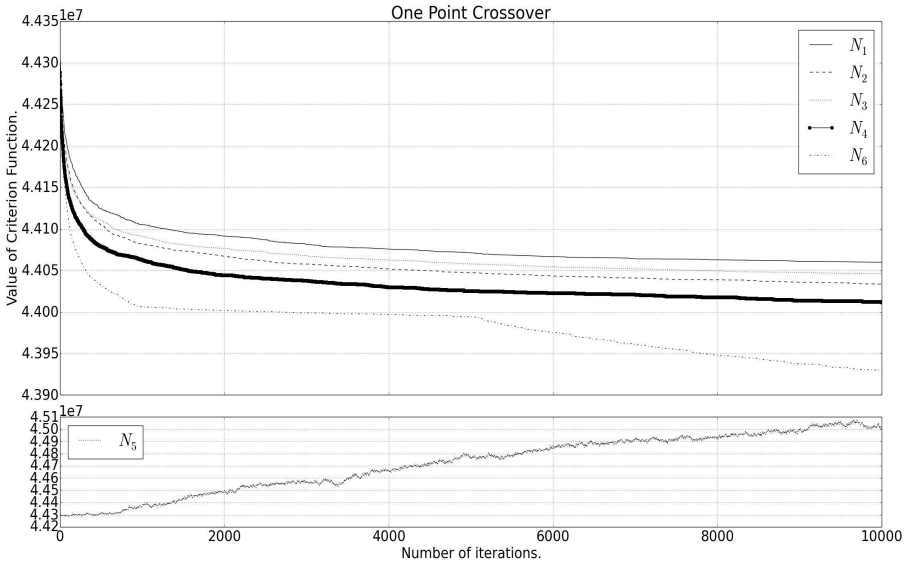



*Figure 1.* One point crossover vairous settings.

crossover's probability the algorithm with the parameter setting $N_2$ is around 35% slower than the algorithm with the parameter setting $N_1$.

If we only compare the mutation probability effect, we limit ourselves to parameter settings $N_1$ and $N_3$. Similar to the crossover probability effect, algorithms with higher probability are more efficient at the beginning, but after few iterations we see that the algorithm with the lower mutation probability becomes more efficient. For the first roughly 200 iterations the algorithm with the setting $N_3$ is around 20% more efficient than the algorithm with the setting $N_1$, but after that the algorithm $N_1$ becomes more efficient for about around 10% on average.

Algorithm with the parameter setting $N_4$ is one of the most efficient at the beginning. In comparison to the $N_2$ or $N_3$ it is more efficient for about 5%. After few iterations the efficiency becomes comparable to algorithms with parameter settings $N_1$, $N_2$ and $N_3$. But due to double the population size the algorithm with this setting needs on average twice as much time as with other settings for one iteration.

Based on these results we were able to construct a sequence of parameter settings. We clearly see that the algorithm with this sequence has the best result compared to other settings. Speed of the algorithm with setting $N_6$ is comparable to the algorithm with setting $N_4$, but because of the overall efficiency it is clearly significantly better. In addition to

this, we need only around 1,000 iterations to get at least as good a result as the algorithm with any other setting at 10,000th iteration.

**One-Point versus Multi-Point Crossover.** Let us now analyze and compare both crossover operators. We compare operators on the same data set with the same parameter settings. In this case we do the optimization on a time interval of 168 hours.

*Table 2.* Parameter settings.

| Parameters / Settings | $N_7$ | $N_8$ |
|---|---|---|
| Iterations | 10,000 | 10,000 |
| Population size | 30 | 30 |
| Elitism | 4 | 4 |
| Crossover | MPC | OPC |
| Crossover Probability | 50% | 50% |
| Mutation Probability | | |
| - population | 25% | 25% |
| - individual | 20% | 20% |

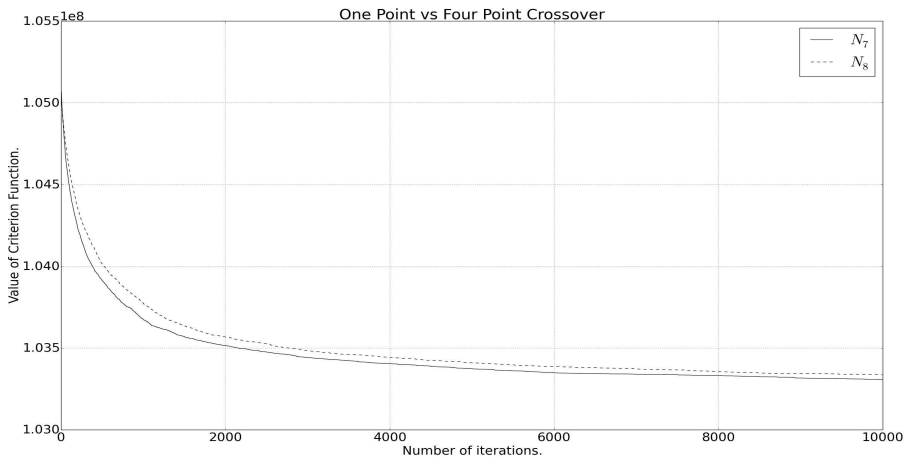



*Figure 2.* Multi point vs one point crossover.

We see a similar pattern than when comparing the algorithm with higher and lower crossover probabilities. At the beginning the algorithm with multi-point crossover is significantly (20%) more efficient, but later one-point crossover gets an advantage of 10% on average. This is due to the fact that closer we get to a sub-optimal solution, harder it is to find a better solution and therefore smaller steps should be taken to find it.

We see that the difference between best solutions of the algorithm with selected parameters is minimal, but the range between the best and worst solution of each setting at the 10,000th iteration is for 37% smaller in the $N_7$ case. This could tell us that with this setting we are able to find a better solution with a higher probability but it also could tell us that it is more likely to stay in the local optima. The algorithm with the setting $N_7$ maintains the advantage though due to the more efficient starting iterations and the fact that it is on average slower only for about 15-20%.

Thus if we would like to optimize a time interval of which length is appropriate for multi-point crossover, it is good to combine and apply both when constructing a sequence of parameter settings. First we should use multi-point crossover and later one-point crossover.

## 5. Conclusion

We have seen that the parameter settings of the algorithm can strongly affect on the performance properties and the efficiency. Empirical results have shown theoretical analysis as valid for the unit commitment problem solved by the GA. Regardless of that, detailed analysis of the algorithm settings on real data should be done for each of the goals to get the best results possible. This can be a time consuming, however worth the effort due to the performance and efficiency gains. Further projects in the future could be an implementation of a self-adaptive GA [11]. That is how we would be able to find a good enough (or even optimal) parameter settings. On the other hand one has to be aware of the difficulty of finding better feasible solutions. This means that we will, sooner or later, reach the performance limit.

In this article we have successfully analyzed and compared different algorithm settings and were able to differentiate between them. We did not focus on finding a global optimum on a chosen data set. We have also shown successfully that a variable-structure GA can drastically improve the efficiency of the algorithm.

## References

[1] M. Carrion and J. Arroyo. A computationally efficient mixed-integer linear formulation for the thermal unit commitment problem. *IEEE T. Power Syst.*, 21(3):1371–1378, 2006.

[2] R. Cerf. Asymptotic Convergence of Genetic Algorithms. *Adv. Appl. Probab.*, 30(2):521–550, 1998.

[3] K. A. De Jong. *An analysis of the behavior of a class of genetic adaptive systems.* The University of Michigan, Ann Arbor, 1975.

[4] A. Fragioni, C. Gentile, and F. Lacalandra. Tighter Approximated MILP Formulations for Unit Commitment Problems. *IEEE T. Power Syst.*, 24(1):105–113, 2009.

[5] Y. Gao. An Upper Bound on the Convergence Rates of Canonical Genetic Algorithms. *Complexity International*, Vol. 5, 1998.

[6] M. Iosifescu. *Finite Markov Processes and Their Applications.* Wiley, Chichester, 1980.

[7] K. A. Juste, H. Kita, and E. Tanaka. An evolutionary programming solution to the unit commitment problem. *IEEE T. Power Syst.*, 14(4):1452–1459, 1999.

[8] Y. Leung, Y. Gao, and Z. B. Xu. Degree of population diversity: A perspective on premature convergence in genetic algorithms and its Markov chain analysis. *IEEE T. Neural Networ.*, 8(5), 1165–1176, 1996.

[9] J. A. López, J. L. Ceciliano-Meza, I. Guillén, and R. N. Gómez. A Heuristic algorithm to solve the unit commitment problem for real-life large-scale power systems. *International Journal of Electrical Power & Energy Systems*, 49:287-295, 2013.

[10] T. Niknam, A. Khodaei, and F. Fallahi. A new decomposition approach for the thermal unit commitment problem. *Appl. Energy*, 86(9):1667–1674, 2009.

[11] G. Papa. Parameter-less algorithm for evolutionary-based optimization. *Comput. Optim. Appl.*, 56 (1):209–229, 2013.

[12] G. G. Robertson. Population size in classifier systems. In *Proc. 5th International Conference on Machine Learning*, pages 142–152, 1988.

[13] W. Snyder, H. Powell, and J. Rayburn. *Dynamic programming approach to unit commitment. IEEE T. Power Syst.*, 2(2):465–473, 1987.