# BIOINSPIRED OPTIMIZATION METHODS AND THEIR APPLICATIONS

# BIOINSPIRED OPTIMIZATION METHODS AND THEIR APPLICATIONS

## Proceedings of the Third International Conference on Bioinspired Optimization Methods and their Applications, BIOMA 2008

## 13–14 October 2008, Ljubljana, Slovenia

Edited by
BOGDAN FILIPIČ
JURIJ ŠILC

Jožef Stefan Institute, Ljubljana

**Jožef Stefan Institute**
Ljubljana, Slovenia

Editors: Bogdan Filipič, Jurij Šilc

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

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

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

004.02(063)(082)
005.519.1(063)(082)
510.5(063)(082)

INTERNATIONAL Conference on Bioinspired Optimization Methods and their Applications (3 ; 2008 ; Ljubljana)
Bioinspired optimization methods and their applications : proceedings of the Third International Conference on Bioinspired Optimization Methods and their Applications - BIOMA 2008, 13-14 October 2008, Ljubljana, Slovenia / edited by Bogdan Filipič, Jurij Šilc. - Ljubljana : Jožef Stefan Institute, 2008

ISBN 978-961-264-002-6
1. Gl. stv. nasl. 2. Filipič, Bogdan

241035264

BIOMA 2008 is part of the Information Society multiconference.
The IS 2008 multiconference is partially financed by the Slovenian Research Agency and Jožef Stefan Institute.

ISSN-1581-9973

# Contents

# Preface

Computational techniques inspired by natural phenomena are nowadays being studied and employed in practice at an increasing rate. Their robustness, ability of providing multiple solutions to a problem at hand, and suitability for implementation in distributed computing environments make them both a challenging research issue and a powerful problem solving tool. The area of computer optimization algorithms is no exception to this trend. Evolutionary algorithms, ant colony optimization, and particle swarm optimizers are examples of methods that overcome many shortcomings of traditional algorithms in application domains where little is known about the properties of the underlying problems.

These proceedings contain some of the recent theoretical and practical contributions to the field of bioinspired optimization. The papers were presented at the Third International Conference on Bioinspired Optimization Methods and their Applications (BIOMA 2008), held in Ljubljana, Slovenia, on 13 and 14 October 2008. Encouraged by the success of the BIOMA conferences in 2004 and 2006, we organized the conference again to bring together theoreticians and practitioners to present their new achievements in a single stream of talks, and exchange the ideas in informal discussions. After the review process, 14 papers were accepted for publication, contributed by 40 (co)authors coming from 13 countries.

Professor Kalyanmoy Deb from the Indian Institute of Technology in Kanpur, India, currently visiting the Helsinki School of Economics in Finland, a pioneer and one of the leading researchers in evolutionary multiobjective optimization (EMO) has contributed the BIOMA 2008 invited talk on EMO principles, algorithms, and current research and application activities in multiple criteria decision making. Theoretical and algorithmic studies address specialized topics in bioinspired optimization: coevolutionary particle swarm optimization to improve the performance of game playing and market strategy analyses, hybridization of fast evolution strategies and particle swarm optimization, distributed implementation of the multilevel ant colony optimization meta-

heuristic, experimental evaluation of a self-adaptive differential evolution incorporating SQP local search, a statistical nonparametric comparison procedure for assessing the relative performance of evolutionary algorithms, testing procedures for global search algorithms used in space trajectory design, and clustering short-text corpora with a discrete particle swarm optimizer. Presentations of applied work come from a variety of domains: solving constrained engineering optimization problems, protein structure prediction, computer-based jazz harmony evolution, transportation scheduling, protein function prediction, and scheduling electrical power generation.

BIOMA 2008 was sponsored by the Slovenian Research Agency. It was organized as part of the 11th International Multiconference Information Society (IS 2008) held at the Jožef Stefan Institute, Ljubjana, from 13 to 17 October 2008.

Our thanks go to the conference sponsors, members of the program and organizing committees, the invited speaker, paper presenters and other participants for contributing their parts to the conference. We wish you would find the event both beneficial and enjoyable, and this collection of papers inspiring for your work.

*Ljubljana, 3 October 2008*

Bogdan Filipič and Jurij Šilc

## Program Committee

Bogdan Filipič, *Chair*, Jožef Stefan Institute, Ljubljana, Slovenia
Jurij Šilc, *Chair*, Jožef Stefan Institute, Ljubljana, Slovenia
Janez Brest, University of Maribor, Slovenia
Dirk Büche, University of Applied Sciences Northwestern Switzerland, Switzerland
Carlos A. Coello Coello, CINVESTAV-IPN, Mexico
Marco Dorigo, Université Libre de Bruxelles, Belgium
Rolf Drechsler, University of Bremen, Germany
Gary B. Fogel, Natural Selection, Inc., San Diego, USA
Jürgen Jakumeit, ACCESS e.V., Aachen, Germany
Peter Korošec, Jožef Stefan Institute, Ljubljana, Slovenia
Barbara Koroušić Seljak, Jožef Stefan Institute, Ljubljana, Slovenia
Marjan Mernik, University of Maribor, Slovenia
Zbigniew Michalewicz, University of Adelaide, Australia
Edmondo Minisci, University of Glasgow, UK
Nalini N, Siddaganga Institute of Technology, Karnataka, India
Gregor Papa, Jožef Stefan Institute, Ljubljana, Slovenia
Mario Pavone, University of Catania, Italy
Günter Rudolph, University of Dortmund, Germany
Jim Tørresen, University of Oslo, Norway

## Organizing Committee

Gregor Papa, *Chair*, Jožef Stefan Institute, Ljubljana, Slovenia
Peter Korošec, Jožef Stefan Institute, Ljubljana, Slovenia
Barbara Koroušić Seljak, Jožef Stefan Institute, Ljubljana, Slovenia
Katerina Taškova, Jožef Stefan Institute, Ljubljana, Slovenia
Tea Tušar, Jožef Stefan Institute, Ljubljana, Slovenia

# Contributing Authors

**Yusri S. Akil** received his Master degree in Electrical Engineering from the University of Hasanuddin, Indonesia in 2005. He is currently a lecturer at the University of Hasanuddin, Faculty of Engineering. His research interests include electrical power system, power system economics. He is a Member of IATKI in Indonesia.

**Kjell Bäckman** received his masters degree in Art & Technology from the Chalmers University of Gothenburg, Sweden, in 2006. He is currently a Ph.D. student at the IT University of Gothenburg, Sweden, and teacher in programming at the West University of Trollhttan, Sweden. He has been piano teacher in improvisation at the Musical Academy of Gothenburg, Sweden. His research interests include computer generated music in general and evolutionary algorithms applied to jazz improvisation in specific.

**Mathieu Barrette** received his master degree in electrical engineering from École de technologie supérieure of Montreal in 2008. His is currently an application engineer in robotic systems integration.

**Mohammad Ehsan Basiri** was born in Firoozabad, Iran on May 20, 1982. He received his B.S. degree in Software Engineering from Shiraz University in 2005. He is now with Isfahan University, Isfahan, Iran as M.Sc. student in Artificial Intelligence. His research interests include swarm intelligence, data mining, and bioinformatics.

**Andrew J. Bennett** received his M.Sc. degree in Chemistry from the University of Birmingham in 2005. He is currently a Ph.D. Student in Computational Chemistry at the University of Birmingham. His research interests include natural computation. He is an Associate Member of the Royal Society of Chemistry.

**Borko Bošković** received his B.S. degree in Computer Science from the University of Maribor in 2004. He is currently a teaching assistant at the University of Maribor, Faculty of Electrical Engineering and Computer Science. His research interests include evolutionary algorithms. 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 has been with the Laboratory for Computer Architecture and Programming Languages, University of Maribor, since 1993. He is currently an Associate Professor. His research interests include evolutionary computing, artificial intelligence, and optimization. His fields of expertise embrace programming languages, web-oriented programming, and parallel and distributed computing research. He is a member of IEEE, and ACM.

**Leticia Cecilia Cagnina** received her Master degree in Computer Science from Universidad Nacional de San Luis, Argentina in 2007. She is currently a Ph.D. student at the same University. Her research interests include bio-inspired metaheuristics, optimization mono and multi-objective.

**Carlos A. Coello Coello** received his Ph.D. degree in Computer Science from Tulane University in 1996. He is currently a professor at CINVESTAV-IPN, Faculty of the Computer Science Department. His research interests include evolutionary optimization (both single- and multi-objective). He is a member of IEEE CIS and ACM.

**Shuang Cong** Shuang Cong received her B.S. degree in Automatic Control from Beijing University of Aeronautics and Astronautics in 1982, and Ph.D. degree in system engineering from University of Rome "La Sapienza", Rome, Italy, in 1995. She is currently a Professor with the Department of Automation in University of Science and Technology of China. Her research interests include advanced control strategies for motion control, fuzzy logic control, neural networks design and applications, robotic coordination control and quantum system control.

**Pascal Côté** received his master degree in manufacturing engineering from École de technologie supérieure of Montreal. His is currently a Ph.D. candidate at École Polytechnique of Montréal researching power systems distribution problems.

**Bruno de Kelper** received his Ph.D. in electrical engineering from École de technologie supérieure of Montreal. He is currently professor at the same university, department of electrical engineering. His research interests include realtime simulation and computing.

**Kalyanmoy Deb** is the Deva Raj Chair Professor of Mechanical Engineering at Indian Institute of Technology Kanpur in India. Currently he is visiting Helsinki School of Economics in Finland as a Finland Distinguished Professor. He is the recipient of the prestigious Shanti Swarup Bhatnagar Prize in Engineering Sciences for the year 2005. He has also received the "Thomson Citation Laureate Award" from Thompson Scientific for having highest number of citations in Computer Science during the past ten years in India. He is a fellow of Indian National Academy of Engineering (INAE), Indian National Academy of Sciences, and International Society of Genetic and Evolutionary Computation (ISGEC). He has received Fredrick Wilhelm Bessel Research award from Alexander von Humboldt Foundation in 2003. His main research interests are in the area of computational optimization, modeling and design, and evolutionary algorithms. He has written two text books on optimization and more than 200 international journal and conference research papers. He has pioneered and is a leader in the field of evolutionary multi-objective optimization. He is associate editor and in the editorial board or a number of major international journals. More information about his research can be found from `http://www.iitk.ac.in/kangal/deb.htm`.

**Marcelo Luis Errecalde** received his Dr. degree in Computer Science from the Universidad Nacional del Sur, Baha Blanca, Argentina in 2004. He is currently a professor at the Universidad Nacional de San Luis, Argentina, in the Department of Computer Science. His research interests include: Clustering narrow domain short texts, semantic text categorization, cognitive robotic, argumentation and decision making.

**Susana C. Esquivel** is a professor at the Universidad Nacional de San Luis (Argentina), Faculty of Ciencias Físico-Matemáticas y Naturales. Her research interests include bio-inspired metaheuristics, optimization mono and multi-objective.

**Chunshi Feng** received his B.Sc. degree in Automatic from University of Science and Technology of China in 2004, and is currently a Ph.D. candidate student with Department of Automation, University of Sci-

ence and Technology of China. He is also a visiting researcher at Faculty of Textile Science and Technology, Shinshu University. His research interests include optimization and its applications in robotics.

**Nasser Ghasem-Aghaee** received his Ph.D. degree in Computer Science from the University of Bradford, Bradford, UK in 1987. He is currently a professor at the University of Isfahan, Faculty of Engineering. His research interests include computer simulation, object oriented analysis and design, artificial intelligence, expert systems, AI in software engineering and AI in simulation.

**Sašo Greiner** received his M.Sc. degree in computer science from the University of Maribor in 2004. He is currently a teaching assistant at the Faculty of Electrical Engineering and Computer Science at the University of Maribor. His research areas include design and implementation of programming languages, virtual machines, and reflective architectures.

**Takahiro Hamashima** received his B.S. degree in textile science and technology from Shinshu University in 2008. He is currently a master course student at Shinshu University. His research interests include evolutionary computation.

**Nadjamuddin Harun** received his Ph.D. degree in Electrical Engineering from the University of Hasanuddin and Technische Universiteit Technology Berlin in 1999. He is currently a professor at the University of Hasanuddin, Faculty of Engineering. His research interests include electrical power system. He is a member of APEI in Indonesia.

**Jun Q. He** received his Ph.D. degree in Computer Science from Wuhan University, China in 1995. He is currently a Research Fellow at the University of Aberystwyth, in the Faculty of Computer Science. His research interests include computational intelligence, knowledge-discovery and data mining, network security and parallel computing.

**Diego Alejandro Ingaramo** is currently a Ph.D. student at the Universidad Nacional de San Luis. His research interests include: Clustering narrow domain short texts, semantic text categorization, cognitive robotic, argumentation and decision making.

**Roy L. Johnston** received his Ph.D. degree in Chemistry from the University of Oxford in 1986. He is currently Professor of Computational Chemistry at the University of Birmingham, U.K. His research interests include theoretical chemistry, computational nanoscience and natural computation. He is a Fellow of the Royal Society of Chemistry.

**Andrew Koh** received his Bachelor of Business in Economics (with honours) from Nanyang Technological University in Singapore (1998) and a Masters of Science in Transport Planning and Engineering at the University of Leeds, UK in 1999. With over 10 years of experience in highway transport planning working for private consultancies, he is currently a Research Officer and a Ph.D. student at the same University. His research interest is in the application of evolutionary algorithms for bilevel programming, game theoretical models and congestion control. Andrew is a Member of IEEE, USA as well as a Chartered Member of the Institute of Logistics and Transport, UK.

**Peter Korošec** received B.S. and M.Sc. degrees in computer science from the Faculty of Computer and Information Science, University of Ljubljana, Slovenia, in 2001 and 2004, respectively. In 2006, he received his Ph.D. from the Jožef Stefan International Postgraduate School, Ljubljana, Slovenia. Since winter 2002 he has been a researcher at the Jožef Stefan Institute, Ljubljana, Slovenia. He is presently a researcher at the Computer Systems Department and an assistant professor at the University of Primorska, Faculty of Mathematics, Science and Informational Technologies Koper, Koper, Slovenia. His current areas of research include combinatorial and numerical optimization with ant-based metaheuristics, and distributed computing. More information about his work can be found at `http://csd.ijs.si/korosec/`.

**Marco Locatelli** received his Master and Ph.D. degrees from the Universita Statale di Milano, Italy, in 1992 and 1996 respectively. He is currently an Associate Professor at the Universita di Torino, Department of Computer Science. His main research interests are related to problems in continuous global optimization: multistart like algorithms, simulated annealing algorithms for continuous global optimization, one-dimensional global optimization, concave optimization over polytopes, a special all-quadratic problem: packing of equal circles into the unit square, molecular conformation problems. He is Member of the editorial board of Computational Optimization and Applications.

**Yoshiyuki Matsumura** received his Ph.D. degree in system function engineering from Kobe University in 2002. He is currently a Senior Assistant Professor at Shinshu University, Faculty of Textile Science and Technology. He is also a honorary research fellow at the University of Birmingham, School of Computer Science. His research interests include evolutionary computation, evolutionary robotics and grid computing.

**Edmondo Minisci** received his Master and Ph.D. degrees from the Department of Aerospace Engineering of Politecnico di Torino, Italy, in 1998 and 2004 respectively. He is currently a research fellow at the University of Glasgow, Department of Aerospace Engineering. His research interests include global single- and multi-objective optimization, robust optimization, distributed computing, concurrent design, space mission design.

**Shahla Nemati** was born in Arsanjan, Iran on May 24, 1982. She received her B.S. degree in Hardware Engineering from Shiraz University in 2004 and M.Sc. degree in Hardware Engineering from Isfahan University of Technology in 2008. Her research interests include swarm intelligence, data mining, and speech recognition.

**Kazuhiro Ohkura** received Ph.D. degree in computer science from Hokkaido University in 1997. He is currently a Professor at Hiroshima University, Faculty of mechanical engineering. His research interests include evolutionary computation, reinforcement learning, artificial life, multi-robot systems and manufacturing systems.

**Gregor Papa** is research assistant at the Computer Systems Department, Jožef Stefan Institute, Ljubljana, Slovenia. He received his M.Sc. and Ph.D. degrees in Electrical Engineering from the University of Ljubljana, Slovenia, in 2000 and 2002, respectively. His research interests include optimization techniques, meta-heuristic algorithms, high-level synthesis of integrated circuits, hardware implementations of high-complexity algorithms, and industrial product improvements. His work is published in several international journals.

**Paolo Rosso** received his Ph.D. degree in Computer Science from the University of Trinity College Dublin, Ireland in 1999. He is currently a professor at the Universidad Politcnica of Valencia, Spain (Faculty of Computer Science) and the head of the Natural Language Engineering

Lab. His research interests include: Clustering narrow domain short texts, Question Answering, toponym disambiguation and Geographical Information Retrieval, plagiarism detection, and the use of the Web as lexical resource.

**Jurij Šilc** received his Ph.D. in Electrical Engineering from the University of Ljubljana in 1992. In 1980 he joined the Jožef Stefan Institute, where he is now a researcher. At the Institute, he served as the head of the Computer Architecture Laboratory from 1986 to 1994. He is presently the deputy head of the Computer Systems Department and an assistant professor at the Jožef Stefan International Postgraduate School. He has published over 150 research papers on subjects including magnetic bubble memories, dataflow computing, algorithm mapping, parallel processing, high-level synthesis, combinatorial and numerical optimization, and processor architecture. He has been involved in a number of conferences and professional activities concerned with programming languages, parallel processing, computer architectures, and bio-inspired optimization algorithms. He is a Member of the IEEE. More information about his work can be found at `http://csd.ijs.si/silc/`.

**Mirjam Sepesy Maučec** received her Ph.D. degrees in Computer Science from the Faculty of Electrical Engineering and Computer Science at the University of Maribor in 2001. She is currently a researcher at the same faculty. Her research interests include language modelling, statistical machine translation, computational linguistics, and bio-inspired optimization algorithms.

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

**Eleanor Turpin** received her M.Sc. degree in Computer Science from the University of Birmingham in 2008.

**Massimiliano Vasile** received his Master and Ph.D. degrees from the Department of Aerospace Engineering of Politecnico di Milano, Italy, in

1996 and 2000 respectively. He is currently a Senior Lecturer in Space System Engineering at the Department of Aerospace Engineering and Head of Research for the Space Advanced Research Team at the University of Glasgow (`http://www-legacy.aero.gla.ac.uk/Research/SpaceArt/index.html`). His main research interests are: space mission analysis and design, global and multiobjective optimization, bio-inspired optimization methods, asteroids, evolutionary computation, concurrent engineering, multidisciplinary design, swarm intelligence, formation flying, autonomous robotic systems. He is Member of the IEEE, AIAA and AIRO.

**Tony Wong** received his Ph.D. in computer engineering from École Polytechnique of Montreal. He is currently professor at the École de technologie supérieure, department of automated manufacturing engineering. His research interests include metaheuristic optimization and parallel computing.

**Aleš Zamuda** received M.Sc. degree in computer science from University of Maribor in 2008. He is currently a Ph.D. candidate and affiliated with the Faculty of Electrical Engineering and Computer Science at the University of Maribor. His research interests include self adaptive techniques for differential evolution, multicriterion optimization, evolutionary algorithms, multicriterion optimization, artificial life, and computer animation. He is a Member of IEEE and IEEE Computational Intelligence Society.

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

I

# INVITED CONTRIBUTION

# EVOLUTIONARY MULTI-OBJECTIVE OPTIMIZATION AND DECISION MAKING

Kalyanmoy Deb
*Department of Business Technology, Helsinki School of Economics*
*Helsinki, Finland*
kalyanmoy.deb@hse.fi

**Abstract** Evolutionary multi-objective optimization (EMO) has now established itself as a mature field of research and application with an extensive literature, many commercial softwares, numerous freely downloadable codes, a dedicated biannual conference running successfully, special sessions and workshops held at all major evolutionary computing conferences, and full-time researchers from universities and industries from all around the globe. In this paper, we make a brief outline of EMO principles, some EMO algorithms, and focus on current research and application activities of EMO, particularly in the area of multiple criteria decision making.

**Keywords:** Constrained handling, Decision making, Evolutionary algorithms, Multi-objective optimization

## 1. Introduction

For the past three decades, evolutionary optimization (EO) methodologies have been popularly used for various optimization problem solving tasks and have found their niches in handling nonlinearities, large dimensionalities, non-differentiabilities in functions, non-convexities, multiplicity in optima, multiplicity in objectives, uncertainties in decision and problem parameters, problems having large computational overheads and various other complexities for which the classical optimization methodologies are known to be vulnerable. For the past 15 years or so, EO methodologies have been suitably extended to solve multi-objective optimization problems. In a recent survey conducted before the World Congress on Computational Intelligence (WCCI) in 2006, the evolutionary multi-objective optimization (EMO) field was judged as one of the

three fastest growing field of research and application among all computational intelligence areas. As the name suggests, multi-objective optimization problems handle multiple conflicting objectives and, by nature, give rise to a set of *Pareto-optimal* solutions which need a further processing to arrive at a single preferred solution. To achieve the first task of finding a set of trade-off Pareto-optimal solutions, it becomes a natural proposition to use a modified EO, because the use of population in an iteration helps an EO to simultaneously find multiple Pareto-optimal solutions in a single simulation run.

In this paper, we briefly describe the principles of EMO and then discuss a few well-known procedures. Thereafter, we highlight the current research and application of EMO in handling the second aspect of preferring a single Pareto-optimal solution using multiple criteria decision making (MCDM) approaches. This paper should motivate interested readers to look into the extensive EMO literature indicated in the reference section for more details.

## 2. Evolutionary Multi-Objective Optimization (EMO)

As in the single-objective optimization problem, the multi-objective optimization problem usually has a number of constraints which any feasible solution (including the optimal solution) must satisfy:

$$\left.\begin{array}{rll} \text{Minimize/Maximize} & f_m(\mathbf{x}), & m = 1, 2, \ldots, M; \\ \text{subject to} & g_j(\mathbf{x}) \geq 0, & j = 1, 2, \ldots, J; \\ & h_k(\mathbf{x}) = 0, & k = 1, 2, \ldots, K; \\ & x_i^{(L)} \leq x_i \leq x_i^{(U)}, & i = 1, 2, \ldots, n. \end{array}\right\} \quad (1)$$

A solution $\mathbf{x}$ is a vector of $n$ decision variables: $\mathbf{x} = (x_1, x_2, \ldots, x_n)^T$. The last set of constraints are called variable bounds, restricting each decision variable $x_i$ to take a value within a lower $x_i^{(L)}$ and an upper $x_i^{(U)}$ bound.

The optimal solutions in multi-objective optimization can be defined from a mathematical concept of *partial ordering*. In the parlance of multi-objective optimization, the term *domination* is used for this purpose. The domination between two solutions is defined as follows [7, 24]:

**Definition 1** *A solution $\mathbf{x}^{(1)}$ is said to dominate the other solution $\mathbf{x}^{(2)}$, if both the following conditions are true:*

1 *The solution $\mathbf{x}^{(1)}$ is no worse than $\mathbf{x}^{(2)}$ in all objectives. Thus, the solutions are compared based on their objective function values (or*

*location of the corresponding points ($\mathbf{z}^{(1)}$ and $\mathbf{z}^{(2)}$) on the objective space).*

2 *The solution $\mathbf{x}^{(1)}$ is strictly better than $\mathbf{x}^{(2)}$ in at least one objective.*

For a given set of solutions, a pair-wise comparison can be made using the above definition and whether one point dominates the other can be established. All points which are not dominated by any other member of the set are called the non-dominated points. With the above concept, now it is easier to define the *Pareto-optimal solutions* in a multi-objective optimization problem. If the given set of points for the above task contain all points in the search space, the points lying on the non-domination front, by definition, do not get dominated by any other point in the objective space, hence are Pareto-optimal points.

## 2.1 EMO Principles

Evolutionary multi-objective optimization (EMO) algorithms are designed to have following two properties:

1 Find a set of solutions close to the optimal solutions, and

2 Find a set of solutions which are diverse enough to represent the spread of the true Pareto-optimal solutions.

However, finding a set of solutions has a difficulty. From a practical standpoint, a user needs only one solution, no matter whether the associated optimization problem has a single objective or multiple objectives. In the case of multi-objective optimization, the user is now in a dilemma. Since a number of solutions are optimal, the obvious question arises: Which of these optimal solutions must one choose? This is not an easy question to answer. It involves a number of higher-level information which are often non-technical, qualitative and experience-driven. However, if a set of many trade-off solutions are already worked out or available, one can evaluate the pros and cons of each of these solutions based on all such non-technical and qualitative, yet still important, considerations and compare them to make a choice. Thus, in an approach to solve multi-objective optimization problems, the effort may be put in finding a set of trade-off optimal solutions by considering all objectives to be important. After a set of such trade-off solutions are found, a user can then use higher-level qualitative decision making considerations to make a choice.

## 3. State-of-the-Art EMO Methodologies

A number of *non-elitist* EMO methodologies [28, 15, 17] gave a good head-start to the research and application of EMO, but suffered from the fact that they did not use an important operator – an elite-preservation mechanism which ensures survival of better solutions from one generation to the next – in their procedures. An addition of this property in a single-objective EO was found to be important to have an asymptotic convergence property of an EO algorithm to the global optimum [26]. The next-level EMO algorithms implemented an elite-preserving operator in different ways and gave birth to elitist EMO procedures, some of which we describe in the following subsections.

### 3.1 Elitist Non-dominated Sorting GA or NSGA-II

The NSGA-II procedure [9] for finding multiple Pareto-optimal solutions has the following three features: (i) it uses an elitist principle, (ii) it uses an explicit diversity preserving mechanism, and (iii) it emphasizes non-dominated solutions. In NSGA-II, the offspring population $Q_t$ is first created by using the parent population $P_t$ and the usual genetic operators. Thereafter, the two populations are combined together to form $R_t$ of size $2N$. Then, a non-dominated sorting is used to classify the entire population $R_t$. Once the non-dominated sorting is over, the new population is filled by solutions of different non-dominated fronts, one at a time. The filling starts with the best non-dominated front and continues with solutions of the second non-dominated front, followed by the third non-dominated front, and so on. Since the overall population size of $R_t$ is $2N$, not all fronts may be accommodated in $N$ slots available in the new population. All fronts which could not be accommodated are simply deleted. When the last allowed front is being considered, there may exist more solutions in the last front than the remaining slots in the new population. This scenario is illustrated in Figure 1. Instead of arbitrarily discarding some members from the last front, the solutions which will make the diversity of the selected solutions the highest are chosen. Due to the simplicity and efficient usage of its operators and without the need of any additional parameter setting, NSGA-II procedure is probably the most popular EMO procedure today. A C code implementing the procedure is available from author's web site `http://www.iitk.ac.in/kangal/soft.htm`.

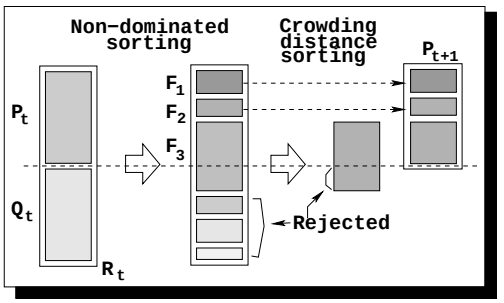



*Figure 1.* Schematic of the NSGA-II procedure.

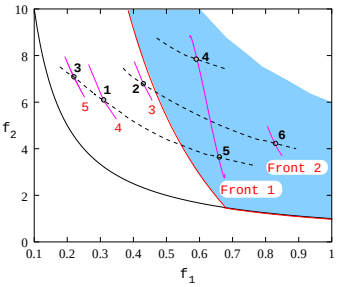



*Figure 2.* Non-constrained-dominated fronts.

## 3.2 Strength Pareto EA (SPEA) and SPEA2

Zitzler and Thiele [33] suggested an elitist multi-criterion EA with the concept of non-domination in their strength Pareto EA (SPEA). They suggested maintaining an external population at every generation storing all non-dominated solutions discovered so far beginning from the initial population. This external population participates in genetic operations. At each generation, a combined population with the external and the current population is first constructed. All non-dominated solutions in the combined population are assigned a fitness based on the number of solutions they dominate. To maintain diversity and in the context of minimizing the fitness function, they assigned a higher fitness value to a non-dominated solution having more dominated solutions in the combined population. On the other hand, a higher fitness is also assigned to solutions dominated by more solutions in the combined population. Care is taken to assign no non-dominated solution a fitness worse than that of the best dominated solution. This assignment of fitness makes sure that the search is directed towards the non-dominated solutions and simultaneously diversity among dominated and non-dominated solution are maintained. On knapsack problems, they have reported better results than other methods used in that study. In their subsequent improved version (SPEA2) [32], three changes have been made. First, the archive size is always kept fixed (thus if there are fewer non-dominated solutions in the archive than the predefined archive size, dominated solutions from the EA population are copied to the archive). Second, a fine-grained fitness assignment strategy is used in which fitness assignment to the dominated solutions are slightly different and a density information is used to resolve the tie between solutions having identical fitness values. Third, a modified clustering algorithm is used with $k$-th nearest neighbor

distance measure and special attention is made to preserve the boundary elements.

### 3.3 Pareto Archived ES (PAES) and Pareto Envelope based Selection Algorithms (PESA and PESA2)

Knowles and Corne [18] suggested a simple possible EMO using evolution strategy (ES). In their Pareto-archived ES (PAES) with one parent and one child, the child is compared with respect to the parent. If the child dominates the parent, the child is accepted as the next parent and the iteration continues. On the other hand, if the parent dominates the child, the child is discarded and a new mutated solution (a new child) is found. However, if the child and the parent do not dominate each other, the choice of child or a parent considers the second task of keeping diversity among obtained solutions using a crowding procedure. To maintain diversity, an archive of non-dominated solutions found so far is maintained. The child is compared with the archive to check if it dominates any member of the archive. If yes, the child is accepted as the new parent and the dominated solution is eliminated from the archive. If the child does not dominate any member of the archive, both parent and child are checked for their *nearness* with the solutions of the archive. If the child resides in a least crowded region in the parameter space among the members of the archive, it is accepted as a parent and a copy of added to the archive. It is interesting to note that both features of (i) emphasizing non-dominated solutions, and (ii) maintaining diversity among non-dominated solutions are present in this simple algorithm. Later, they suggested a multi-parent PAES with similar principles as above. In their subsequent version, they called Pareto Envelope based Selection Algorithm (PESA) [5], they combined good aspects of SPEA and PAES. Like SPEA, PESA carries two populations (a smaller EA population and a larger archive population). Non-domination and the PAES's crowding concept is used to update the archive with the newly created child solutions.

In an extended version of PESA (or PESA2) [6], instead of applying the selection procedure on population members, hyperboxes in the objective space are selected based on the number of residing solutions in the hyperboxes. After hyperboxes are selected, a random solution from the chosen hyperboxes is kept. This region-based selection procedure has shown to perform better than individual-based selection procedure of PESA. In some sense, PESA2 selection scheme is similar in concept to the $\epsilon$-dominance in which predefined $\epsilon$ values determine the hyper-

box dimensions. Other $\epsilon$-dominance based EMO procedures [13, 20] have shown computationally faster and better distributed solutions than NSGA-II or SPEA2.

There also exist other competent EMOs, such as multi-objective messy GA (MOMGA) [30], multi-objective micro-GA [3], neighborhood constraint GA [21], ARMOGA [27], and others. Besides, there exists other EA based methodologies, such as particle swarm EMO [4, 25], ant-based EMO [23, 16], and differential evolution based EMO [1].

## 4. Constraint Handling in EMO

One way to handle constraints is to modify the domination principle. In the presence of constraints, each solution can be either feasible or infeasible. Thus, there may be at most three situations: (i) both solutions are feasible, (ii) one is feasible and other is not, and (iii) both are infeasible. We consider each case by simply redefining the domination principle as follows. A solution $\mathbf{x}^{(i)}$ is said to 'constrained-dominate' a solution $\mathbf{x}^{(j)}$, if any of the following conditions are true: (i) solution $\mathbf{x}^{(i)}$ is feasible and solution $\mathbf{x}^{(j)}$ is not, or (ii) solutions $\mathbf{x}^{(i)}$ and $\mathbf{x}^{(j)}$ are both infeasible, but solution $\mathbf{x}^{(i)}$ has a smaller constraint violation, or (iii) solutions $\mathbf{x}^{(i)}$ and $\mathbf{x}^{(j)}$ are feasible and solution $\mathbf{x}^{(i)}$ dominates solution $\mathbf{x}^{(j)}$ in the usual sense. The above change in the definition requires a minimal change in the NSGA-II or other EMO procedures described earlier. Figure 2 shows the non-dominated fronts on a six-membered population due to the introduction of two constraints. In the absence of the constraints, the non-dominated fronts (shown by dashed lines) would have been `((1,3,5), (2,4,6))`, but in their presence, the new fronts are `((4,5), (6), (2), (1), (3))`. The first non-dominated front is constituted with the best feasible solutions in the population and any feasible solution lies on a better non-dominated front than an feasible solution. This simple modification in domination principle allows to form an appropriate hierarchy among solutions in the population for them to move towards the true constrained Pareto-optimal front.

## 5. EMO and Decision Making

It is important here to note that finding a set of Pareto-optimal solutions by using an EMO is only a part of multi-objective optimization, as choosing a particular solution for implementation is the remaining decision making task which is also an equally important task. Research and application using decision making concepts in the context of EMO are in their infancy and further efforts are needed. Hybrid ideas can be borrowed from the MCDM field for this purpose. The decision making

task of choosing a single preferred solution can be considered from two main aspects:

## 5.1 Generic Consideration

There are some aspects which most practical users would like to consider in narrowing down their choice. For example, in the presence of uncertainties in decision variables and/or problem parameters, the users are usually interested in finding *robust* solutions which are relatively insensitive to the variation in decision variables or parameters. In the presence of such variations, no one would be interested in truly Pareto-optimal but sensitive solutions. In such scenarios, the obtained solutions can be ranked based on their sensitivity to the uncertainties in variables or parameters, and the preference can be narrowed to the ones having smaller sensitivities. A recent study on an electric power dispatch problem has demonstrated such a generic consideration in the process of choosing a single preferred solution [8].

In addition, if some other special points, such as a *knee* point which demands a large sacrifice in at least one objective to achieve a small gain in another objective, exist in the set of obtained solutions, they can be preferred. Other generic considerations are as follows: preference of points having correlated relationship between objectives to decision variables, points having multiplicity (finding Pareto-optimal solutions corresponding to multiple (say at least two or more) decision variable vectors but each having identical objective values), points for which decision variable values are well within their allowable bounds and not near their lower or upper boundaries, points having some theoretical aspects such as all Lagrange multipliers having more or less same absolute values (condition for having equal importance to each constraint), and others. These considerations are motivated from the fundamental and practical aspects of optimization and may be utilized to narrow down the choice.

## 5.2 Subjective Consideration

In this category, any problem-specific information can be used to narrow down the choices and the process may even lead to a single preferred solution at the end. Most decision making procedures use some preference information (utility functions, reference point approaches [31], reference direction approaches [19], and a host of other considerations [24]) to select a subset of Pareto-optimal solutions. A recent book is dedicated to the discussions of many such multiple criteria decision analysis (MCDA) tools and collaborative suggestions with EMO [2]. Some hy-

brid EMO and MCDA algorithms are also suggested in the recent past [14, 12, 11, 29, 22] for this purpose.

### 5.3 A Case Study

A recent study [10] suggested an interactive multi-objective optimization procedure, which first finds a nondominated front using NSGA-II and then allows focusing on a preferred region by using a number of generic and subjective MCDA approaches. Figure 3 shows nondominated solutions obtained by NSGA-II (with a plus). Robust solutions are then searched using The front is then searched for solutions having a 2% perturbances in decision variables and the obtained robust frontier is shown in the same figure with circles. Thereafter, to reduce our focus further, we now use a subjective decision-making procedure of surrogate worth trade-off. Of the robust solutions, we are interested in solutions for which a 100% sacrifice in the cost value (first objective), at-least 150% improvement in deflection (second objective) occurs. That is, from a solution if we double the cost value, we are interested in solutions which reduces the deflection 2.5 times. Simultaneously, we would also like to ensure that a saving of at-least 25% cost for a 100% sacrifice in deflection. Resulting solutions are found to lie within the rectangular box.

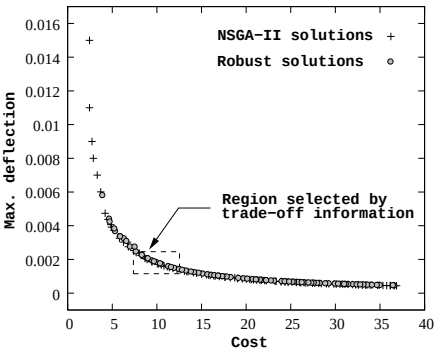



*Figure 3.* NSGA-II solutions are modified for handling uncertainties in variables and then for a user-defined trade-off relationship in the welded-beam design problem.

To narrow down the preferred region, next we consider a subjective decision-making tool with reference points. Say, we are interested in solutions towards two extreme regions of the remaining trade-off front and specify following two reference (aspiration) points in the objective space: $(8.7, 0.00195)^T$ and $(12.0, 0.0014)^T$. Figure 4 shows the final solutions obtained by the reference NSGA-II run on both reference points simultaneously. Reference points are also shown in the figure.

Finally, we decide to use another subjective decision-making tool based on the utility function approach. We decide to use the following utility function:

$$\text{Minimize} \quad U(f_1, f_2) = f_1 \times f_2. \tag{2}$$

Since cost and deflection are conflicting to each other, a product of the two objective values in the regions of our interest may be thought as a combined utility measure, minimizing which may result a solution having small values of both objectives. Figure 5 shows the contour plot of the above utility function and reference point based NSGA-II solutions. The utility function is tangential with the reference NSGA-II solutions at point $A$, thereby meaning that the solution A is the most preferred solution with respect to the chosen utility.

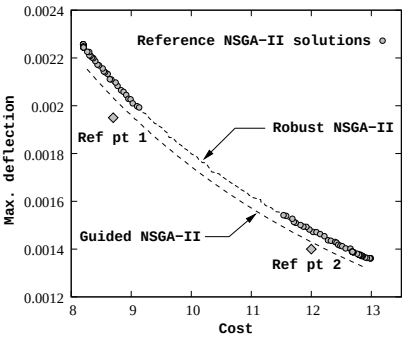



*Figure 4.* Robust Pareto-optimal solutions based on the target values.

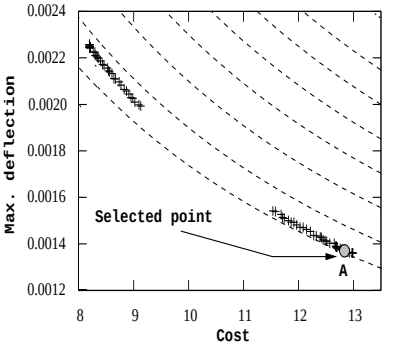



*Figure 5.* Final solution selected based on utility function.

## 6. Conclusions

This paper has provided a brief introduction to a fast-growing field of multi-objective optimization based on evolutionary algorithms. The EMO principle of solving multi-objective optimization had been to first find a set of Pareto-optimal solutions and then choose a preferred solution. Since an EO uses a population of solutions in each iteration, EO procedures are potentially viable techniques to find and capture a number of Pareto-optimal solutions in a single simulation run. Besides their routine applications in solving multi-objective optimization problems, EMO has spread its wings in aiding other types of optimization problems, such as single-objective constrained optimization, clustering problems etc. EMO has been used to unveil important hidden knowledge about what makes a solution optimal. EMO techniques are increasingly being found to have tremendous potential to be used in conjunction with multiple criteria decision making tasks in not only finding a set of optimal solutions but also to aid in selecting a preferred solution at the end.

## References

[1] B.V. Babu and M.M.L. Jehan. Differential Evolution for Multi-Objective Optimization. In *Proc. IEEE Congress on Evolutionary Computation (CEC 2003)*, pages 2696–2703, Canberra, Australia, 2003.

[2] J. Branke, K. Deb, K. Mietinnen, and R. Slowinski. *Multiobjective Optimization: Interactive and Evolutionary Approaches*. Springer-Verlag, in press.

[3] C.A. Coello Coello and G. Toscano. A micro-genetic algorithm for multiobjective optimization. Technical Report, Lania-RI-2000-06, Laboratoria Nacional de Informatica Avanzada, Xalapa, Veracruz, Mexico, 2000.

[4] C.A. Coello Coello and M.S. Lechuga. MOPSO: A Proposal for Multiple Objective Particle Swarm Optimization. In *Proc. IEEE Congress on Evolutionary Computation (CEC 2002)*, pages 1051–1056, Honolulu, HI, 2002.

[5] D. Corne, J. Knowles, and M. Oates. The Pareto envelope-based selection algorithm for multiobjective optimization. In *Proc. Sixth International Conference on Parallel Problem Solving from Nature VI (PPSN-VI)*, pages 839–848, Paris, France, 2000.

[6] D.W. Corne, N.R. Jerram, J.D. Knowles, and M.J. Oates. PESA-II: Region-based selection in evolutionary multiobjective optimization. In *Proc. Genetic and Evolutionary Computation Conference (GECCO-2001)*, pages 283–290, San Francisco, CA, 2001.

[7] K. Deb. *Multi-objective optimization using evolutionary algorithms*. Chichester, UK, Wiley, 2001.

[8] K. Deb. Scope of stationary multi-objective evolutionary optimization: A case study on a hydro-thermal power dispatch problem. *J. Global Optim.*, 2008. (DOI 10.1007/s10898-007-9261-y).

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

[10] K. Deb and S. Chaudhuri. I-MODE: An interactive multi-objective optimization and decision-making using evolutionary methods. In *Proc. Fourth International Conference on Evolutionary Multi-Criteria Optimization (EMO 2007)*, pages 788–802, Matsushima/Sendai, Japan, 2007.

[11] K. Deb and A. Kumar. Interactive evolutionary multi-objective optimization and decision-making using reference direction method. In *Proc. Genetic and Evolutionary Computation Conference (GECCO-2007)*, pages 781–788, London, UK, 2007.

[12] K. Deb and A. Kumar. Light beam search based multi-objective optimization using evolutionary algorithms. Technical Report, KanGAL Report No. 2007005, Indian Institute of Technology Kanpur, India, 2007.

[13] K. Deb, M. Mohan, and S. Mishra. Towards a quick computation of well-spread pareto-optimal solutions. In *Proc. Second Evolutionary Multi-Criterion Optimization (EMO 2003) Conference*, pages 222–236, Faro, Portugal, 2003.

[14] K. Deb, J. Sundar, N. Uday, and S. Chaudhuri. Reference point based multi-objective optimization using evolutionary algorithms. *International Journal of Computational Intelligence Research (IJCIR)*, 2(6):273–286, 2006.

[15] C.M. Fonseca and P.J. Fleming. Genetic algorithms for multiobjective optimization: Formulation, discussion, and generalization. In *Proc. Fifth International Conference on Genetic Algorithms (ICGA93)*, pages 416–423, University of Illinois at Urbana-Champaign, IL, 1993.

[16] M. Gravel, W.L. Price, and c. Gagné. Scheduling continuous casting of aluminum using a multiple objective ant colony optimization metaheuristic. *Eur. J. Oper. Res.*, 143(1):218–229, 2002.

[17] J. Horn, N. Nafploitis, and D.E. Goldberg. A niched Pareto genetic algorithm for multi-objective optimization. In *Proc. First IEEE Conference on Evolutionary Computation*, pages 82–87, Orlando, FL, 1994.

[18] J.D. Knowles and D.W. Corne. Approximating the non-dominated front using the Pareto archived evolution strategy. *Evol. Comput.*, 8(2):149–172, 2000.

[19] P. Korhonen and J. Laakso. A visual interactive method for solving the multiple criteria problem. *Eur. J. Oper. Res.*, 24:277–287, 1986.

[20] M. Laumanns, L. Thiele, K. Deb, and E. Zitzler. Combining convergence and diversity in evolutionary multi-objective optimization. *Evol. Comput.*, 10(3):263–282, 2002.

[21] D.H. Loughlin and S. Ranjithan. The neighborhood constraint method: A multiobjective optimization technique. In *Proc. Seventh International Conference on Genetic Algorithms*, pages 666–673, East Lansing, MI, 1997.

[22] M. Luque, K. Miettinen, P. Eskelinen, and F. Ruiz. Three different ways for incorporating preference information in interactive reference point based methods. Technical Report W-410, Helsinki School of Economics, Helsinki, Finland, 2006.

[23] P.R. McMullen. An ant colony optimization approach to addessing a JIT sequencing problem with multiple objectives. *Artif. Intell. Eng.*, 15(3):309–317, 2001.

[24] K. Miettinen. *Nonlinear Multiobjective Optimization*. Kluwer, Boston, 1999.

[25] S. Mostaghim and J. Teich. Strategies for Finding Good Local Guides in Multiobjective Particle Swarm Optimization (MOPSO). In *Proc. IEEE Swarm Intelligence Symposium*, pages 26–33, Indianapolis, IN, 2003.

[26] G. Rudolph. Convergence analysis of canonical genetic algorithms. *IEEE T. Neural Networ.*, 5(1):96–101, 1994.

[27] D. Sasaki, M. Morikawa, S. Obayashi, and K. Nakahashi. Aerodynamic shape optimization of supersonic wings by adaptive range multiobjective genetic algorithms. In *Proc. First International Conference on Evolutionary Multi-Criterion Optimization (EMO 2001)*, pages 639–652, Zurich, Switzerland, 2001.

[28] N. Srinivas and K. Deb. Multi-objective function optimization using non-dominated sorting genetic algorithms. *Evol. Comput.*, 2(3):221–248, 1994.

[29] L. Thiele, K. Miettinen, P. Korhonen, and J. Molina. A preference-based interactive evolutionary algorithm for multiobjective optimization. Technical Report, Working Paper Number W-412, Helsingin School of Economics, Helsingin Kauppakorkeakoulu, Finland, 2007.

[30] D. Van Veldhuizen and G.B. Lamont. Multiobjective evolutionary algorithms: Analyzing the state-of-the-art. *Evol. Comput.*, 8(2):125–148, 2000.

[31] A.P. Wierzbicki. The use of reference objectives in multiobjective optimization. In G. Fandel and T. Gal, editors, *Multiple Criteria Decision Making Theory and Applications*, pages 468–486. Berlin, Springer-Verlag, 1980.

[32] E. Zitzler, M. Laumanns, and L. Thiele. SPEA2: Improving the strength pareto evolutionary algorithm for multiobjective optimization. In K.C. Giannakoglou, D.T. Tsahalis, J. Périaux, K.D. Papailiou, and T. Fogarty, editors, *Evolutionary Methods for Design Optimization and Control with Applications to Industrial Problems*, pages 95–100, Athens, Greece, 2001. International Center for Numerical Methods in Engineering (Cmine).

[33] E. Zitzler and L. Thiele. Multiobjective optimization using evolutionary algorithms – A comparative case study. In *Proc. Parallel Problem Solving from Nature V (PPSN-V)*, pages 292–301, Amsterdam, The Netherlands, 1998.

II

# THEORY AND ALGORITHMS

# NASH EQUILIBRIA, COLLUSION IN GAMES AND THE COEVOLUTIONARY PARTICLE SWARM ALGORITHM

Andrew Koh
*Institute for Transport Studies, University of Leeds*
*Leeds, United Kingdom*
a.koh@its.leeds.ac.uk

**Abstract** In recent work, we presented a deterministic algorithm to investigate collusion between players in a game where the players payoff functions are subject to a variational inequality describing the equilibrium of a transportation system. In investigating the potential for collusion between players, a fixed point iterative algorithm returned a local optimum. In this paper, we apply a coevolutionary particle swarm optimization (PSO) algorithm developed in earlier research in an attempt to return the global maximum. A numerical experiment is used to verify the performance of the algorithm in overcoming local optimum.

**Keywords:** Bilevel variational inequality, Diagonalisation, Equilibrium problems with equilibrium constraints, Nash equilibrium

## 1. Introduction

This paper discusses the determination of Nash Equilibrium (NE) subject to variational inequality constraints. This is an emerging area of research within transportation network analysis and has particular significance in an environment of deregulated infrastructure provision. When these private sector participants compete in a market in simultaneously and non-cooperatively deciding their strategic variables to offer to consumers, the concept of the Cournot-Nash game can be used to model the equilibrium variables offered by each firm to the consumers. When the firms's actions are constrained by the a variational inequality describing system equilibrium, we obtain an Equilibrium Problem with Equilibrium Constraints (EPEC) [11]. Throughout this paper, the terms "firm" and "players" are used interchangeably as the subject matter transcends both game theory and market structures.

In this paper, we focus on this problem to specifically consider implicit collusion and how this leads to equilibrium strategies that can be beneficial to the players. In doing so, we make use of the concept of local NE introduced in [16] to distinguish that from a global NE.

This paper is organized as follows. In the next section, we introduce the EPEC and define associated concepts of NE taken from [16]. A currently available deterministic optimization algorithm available is also given. In Section 3, we will outline the essential concepts in a transportation network setting which is our application area. We then describe the coevolutionary particle swarm optimization algorithm (CoPSONash) [7] in Section 4 to this problem and by way of a numerical example in Section 5 show that the CoPSONash obtains a global optimum for this problem overcoming the local NE trap. Section 6 summarizes.

## 2. EPECs and Nash Equilibria

An Equilibrium Problem with Equilibrium Constraints (EPEC) [11, 12] seeks to find equilibrium points in a game when the constraints describe a variational inequality that defines an overall system equilibrium. For the purposes of this paper, the system equilibrium is the equilibrium in route choices in a (highway) transportation setting. The study of EPECs has only just recently surfaced as an important research area within mathematics and optimization theory with significant practical applications e.g. in deregulated electricity markets (e.g. [5]).

### 2.1 Nash Equilibrium

In a single shot normal form game with N players, indexed by $i, j \in \{1, 2, ..., N\}$ with $i \neq j$, each player can play a strategy $u_i \in U_i$ which all players are assumed to announce simultaneously. Let $u = (u_1, u_2, ..., u_N) \in U$ be the combined strategy space of all players in this game and let $\psi_i(u)$ be some payoff or profit function to $i \in$ {1,2,...,$N$} player if the combined strategy is played. Then the combined strategy tuple $u^* = (u_1^*, u_2^*, ..., u_N^*) \in U$ is a Nash Equilibrium (NE) for the game if the following holds

$$\psi_i(u_i^*, u_j^*) \geq \psi_i(u_i, u_j^*) \; \forall u_i \in U_i, \forall i, j \in \{1, 2, ...N\}, i \neq j \qquad (1)$$

Equation 1 states that a NE is attained when no player has an incentive to deviate from his current strategy, based on Nash in [13]. We now consider two refinements from [16].

## 2.2 Local Nash Equilibrium and NE Trap

**Local Nash Equilibrium ([16], Definition 2, p306).** The combined strategy tuple $u^*$(as above) is a local NE if: $\exists \omega > 0$ such that $\forall i, \forall u_i \in B_i^{\omega}(u_i^*)$, $B_i^{\omega}(\hat{u}_i) = \{u_i \in U_i \,|\, \|u_i - \hat{u}_i\| < \omega\}$ the following holds:

$$\psi_i(u_i^*, u_j^*) \geq \psi_i(u_i, u_j^*) \; \forall u_i \in U_i, \forall i, j \in \{1, 2, ...N\}, i \neq j \qquad (2)$$

Each NE that satisfies the above definition given in Eq. 1 clearly also satisfies the definition of local NE given by Eq. 2. But the converse is not generally true. In essence this means that a strategy is only a Nash equilibrium within some ball radius in strategy space; but it may not be necessarily so globally. Hence we define the notion of a local NE trap.

**Local NE Trap ([16], Definition 3, p306).** The combined strategy $u^*$is a local NE trap if: It is a local NE as defined above in *and* in addition: $\exists i$ such that $\exists u_i^{**} \in U_i$the following holds:

$$\psi_i(u_i^{**}, u_j^*) \geq \psi_i(u_i^*, u_j^*) \; \forall u_i \in U_i, \forall i, j \in \{1, 2, ...N\}, i \neq j \qquad (3)$$

## 2.3 A Deterministic Algorithm for EPECs

While novel deterministic algorithms have been recently proposed for EPECs [12], their use has not been widely adopted. Instead, we describe a simple and well known deterministic (gradient based) solution method for this problem.

This algorithm decomposes the problem into a series of interrelated optimization problems, one for each player and then solving each in turn. This is a Gauss-Jacobi/Gauss-Seidel fixed point iteration (FPI) type algorithm. Harker [4] used this algorithm for solving the classical Cournot Nash game from economics. Similarly EPECs in the deregulated electricity markets were solved thus in [5]. The algorithm is presented below (see the Algorithm 1).

The drawback with the above algorithm is that it could terminate at the local NE and fall prey to the local NE trap; an outcome dependent on the choice of the initial strategy of each player (cf Step 1). It has been shown [16] that iterative search algorithms such as the FPI cannot differentiate the real NE from a local NE trap.

**Algorithm 1** Gauss-Jacobi Fixed Point Iteration FPI

1: Set iteration counter $k = 0$. Select a convergence tolerance parameter, $\varepsilon(\varepsilon > 0)$. Choose a strategy for each player. Let the initial strategy set be $u^k = (u_1^k, u_2^k, ..., u_N^k)$. Set $k = k + 1$ and go to Step 2,

2: For the $i^{\text{th}}$ player $i \in \{1, 2, ..., N\}$, solve the following optimization problem: $u_i^{k+1} = \max_{u_i \in U} \psi_i(u_i, u_j^k) \quad i, j \in \{1, 2, ..., N\}, i \neq j,$

3: Convergence Check: If $\sum_{i=1}^{N} \left\| u_i^{k+1} - u_i^k \right\| \leq \varepsilon$ terminate, else return to Step 2.

## 3. Problem Definition

We now describe in more detail the optimization problem at Step 2 of the above algorithm in the context of highway transportation equilibrium.

### 3.1 Notation

Define:

$A$: the set of directed links in a traffic network,

$B$: the set of links which have their tolls optimised, $B \subset A$

$K$: the set of origin destination (O-D) pairs in the network

$\mathbf{v}$: the vector of link flows $\mathbf{v} = [v_a],\ a \in A$

$\mathbf{x}$: the vector of link tolls $\mathbf{x} = [x_a],\ a \in B$

$\mathbf{c}(\mathbf{v})$: the vector of monotonically non decreasing travel costs as a function of link flows $\mathbf{c}(\mathbf{v}) = [c_a(v_a)],\ a \in A$

$\mathbf{d}$: the continuous and monotonically decreasing demand function for each O-D pair as a function of the generalized travel cost between OD pair$k$alone, $\mathbf{d} = [d_k],\ k \in K$ and

$\mathbf{D^{-1}}$: the inverse demand function

$\mathbf{\Omega}$: feasible region of flow vectors,(defined by a linear equation system of flow conservation constraints).

### 3.2 Optimization Problem for Individual Players

This is to find an optimal equilibrium toll (level of road user charge per vehicle) for each firm who separately controls[1] a predefined link on the traffic network under consideration. We can consider this problem to be a Cournot Nash game between these individual players. Therefore the equilibrium decision variables can be determined using the concepts of NE as defined above.

If we assume that each player controls only a single link in the network then, the optimization problem for each player, with the objective being maximizing the revenue[2] is as follows:

$$\underset{x_i}{\text{Max}}\ \psi_i(\mathbf{x}) = v_i(\mathbf{x})x_i, \forall i \in N \tag{4}$$

Where $v_i$ is obtained by solving the variational inequality in Eq. 5 (see [2, 14])

$$\mathbf{c}\left(\mathbf{v}^*, \mathbf{x}\right)^T \cdot \left(\mathbf{v} - \mathbf{v}^*\right) - \mathbf{D^{-1}}\left(\mathbf{d}^*, \mathbf{x}\right)^T \cdot \left(\mathbf{d} - \mathbf{d}^*\right) \geq 0 \text{for} \forall \left(\mathbf{v}, \mathbf{d}\right) \in \mathbf{\Omega} \tag{5}$$

It is important to stress that the vector of link flows can only be obtained by solving the variational inequality given by Eq. 5. This variational inequality represents Wardrop's user equilibrium condition where user equilibrium in route choice is attained when no user can decrease his travel costs by unilaterally changing routes [15]. If we further assume that the travel cost of any link in the network is dependent only on flow on the link itself, the above variational inequality in , for given $\mathbf{x}$, can be solved by means of a convex optimization problem [1] (this is the known as the Traffic Assignment Problem or TAP). Details can be found in [8].

In practice, if this problem is to be solved by the FPI algorithm outlined in Section 2.3, then the optimization problem in Step 2 at each iteration involves using a gradient based optimization method (here we use the Cutting Constraint Algorithm (CCA) [10] to solve the problem for each player using the objective given by Eq. 4. In this case, the variational inequality constraint is implicitly handled within CCA[3].

### 3.3 Considering Collusion

In [8], for a model with 2 players, we introduced a collusion parameter $\alpha$ $(0 \leq \alpha \leq 1)$ to model the possibility of players colluding. With $\alpha$ we can consider a more general form of the expression for the payoff function given in Eq. 4. In this case, at each iteration, the optimization problem to be solved at Step 2 by the FPI algorithm becomes that as given by Eq. 6

$$\underset{x_i}{\text{Max}}\ \psi_i(\mathbf{x}) = v_i(\mathbf{x})x_i + \alpha(v_j(\mathbf{x})x_j), \forall i, j \in N, i \neq j \tag{6}$$

Where $v_i$ is obtained by solving the variational inequality in Eq. 5.

Equation 6 reduces to Eq. 4 when $\alpha = 0$; similarly when $\alpha = 1$, the objective of each player is to maximize the total toll revenue of both players. However, bear in mind that player $i, i \neq j$ can only change tolls on the link under his control and still continues to take the other player's

toll as an exogenous input in his optimization process. Thus whilst the $i$th player is in the process of optimizing his revenue, he takes into account a proportion represented by $\alpha$ of the $j$th player's toll revenue. In doing so, he is "signaling" to his competitor that he wishes to "collude" to maximize the total revenue, not just his own. Thus $\alpha$ represents some intuitive level of collusion between players. We also assume throughout that players reciprocate the actions of the competitors and would do likewise.

The interesting question therefore is whether it is possible to "perturb" the FPI algorithm at each iterate with the intent of simulating this implicit signaling to each other an alternative objective and in so doing collude to raise overall revenues. This problem was not considered in the development of the original PSO based algorithm used for this problem [7] and discussed in the next section.

## 4. CoEvolutionary PSO Algorithm

The PSO algorithm [6] forms the basis of the coevolutionary PSO algorithm (CoPSONash) we developed in [7] as an alternative to the FPI algorithm. For a game with $N$ players, each sub-population represents particles comprising the strategic decision variables (tolls, $x$) for each of these players. If each players' strategies are encoded in a swarm with $H$particles, then the steps of CoPSONash are as follows in the Algorithm 2.

During initialization, particle positions and associated velocities are randomly generated. One strategy from each subpopulation is randomly selected as the initial Nash strategy for that player. Each subpopulation is evaluated separately, by solving the traffic assignment problem , to determine the objective for each player , given the Nash strategy of the other players. Hence, the personal bests and global best particle for each player can be identified. With all subpopulations evaluated, each player's global best particle is announced to the whole group during the key synchronization phase of the algorithm. This synchronization intrinsically embodies coevolution as the fitness of a particular strategy is dependent on that of others in the game. This process continues for a maximum number of user defined iterations. The aim of the algorithm is to evolve a swarm of strategy vectors for each player robust to the strategies of others which would then satisfy the Nash equilibrium condition as defined by Eq. 1. For more details of the algorithm, the reader is referred to [7].

Our numerical example in the next section shows that by harnessing the global search capabilities of PSO, the pitfall of falling into the local

**Algorithm 2** CPSONash

1: Generate $N$ subpopulations (1 for each player) of particles ($x$) and velocities randomly
2: Randomly select one particle from each player as its Nash strategy
3: Evaluate each subpopulation by solving a TAP for fixed $x$ (and compute Eq. 6) given the Nash strategy (from Step 2).
4: Identify the personal best (pb) and global best (gb) of each particle from each subpopulation
5: Set gb as the new Nash strategy for each population
6: **repeat**
7: Synchronization: Announce Nash strategy to all players.
8: **for** each subpopulation $i = 1$ to $N$ **do**
9: Re-evaluate i$^{th}$ subpopulation given the announced Nash strategy of all other players to obtain new pb and gb
10: **for** each particle $j = 1$ to $H$ **do**
11: Fly $j^{th}$ particle using PSO velocity update equation.
12: Update $j^{th}$ particle position using PSO position update equation.
13: Solve TAP (cf Eq. 5) with new $x$ to compute Eq. 6 and obtain the objective.
14: Update pb if fitter than previous pb.
15: Update gb if fitter than fittest discovered by $i^{th}$ subpopulation so far.
16: next $j$
17: **end for**
18: Identify gb particle and set this as the Nash strategy for subpopulation $i$
19: next $i$
20: **end for**
21: **until** Termination Criteria is met (e.g. after a given maximum number of iterations)

NE trap can be obviated. In addition, any variant of PSO (see [3] for a full review) can be employed in the search process in Steps 11 to 18 of the above.

## 5. Numerical Example

The numerical example used here is a network shown in Fig. 1 and taken from [9]. The link parameters and the elastic demand functions can be found therein. This network has 18 one way links with 6 O-D

pairs (1 to 5, 1 to 7, 5 to 1, 5 to 7, 7 to 1 and 7 to 5). Links 7 and 10, shown as dashed lines in Fig. 1, are the only links in the network subject to tolls. The maximum allowable toll for each link was set to be 1000 seconds.

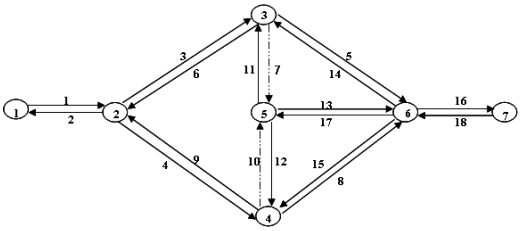



*Figure 1.* Traffic Network for Numerical Example.

Our numerical example focuses only on the case when $\alpha$, the collusion parameter, for each player, equals 1. In this case, the solution of the EPEC should be the similar to assuming that 1 player has control over both links 7 and 10 in the network. This is termed the monopoly solution which represents the maximum total possible revenue arising from tolls on these two links and serves as a benchmark in terms of the total revenues received. The results of the FPI algorithm (with CCA) are contrasted with that obtained by CPSONash and the benchmark and are shown in Table 1. This table also shows the best solution from 30 runs of the CPSONash algorithm (with 200 iterations per run).

*Table 1.* Comparing FPA with CPSONash $\alpha = 1$

| | MONOPOLY | | FPA-CCA | | CPSONash | |
|---|---|---|---|---|---|---|
| | Toll (secs) | Revenue (secs/hr) | Toll (secs) | Revenue (secs/hr) | Toll (secs) | Revenue (secs/hr) |
| Link7 | 713.19 | 280,255 | 189.76 | 116,186 | 713.17 | 282,291 |
| Link10 | 709.53 | 266,465 | 186.58 | 111,216 | 709.55 | 264,427 |
| Total | | 546,720 | | 227,402 | | 546,719 |

Figure 2 plots the revenue surface (i.e. the revenue obtained by simultaneously varying tolls on Links 7 and 10) and illustrates that the solution obtained by the FPI algorithm is in fact a local optimum of this function. From Fig. 2, it is evident that this algorithm fell into a local

NE trap defined in Section 2.2 while the CoPSONash converged to the global optimum for this problem.

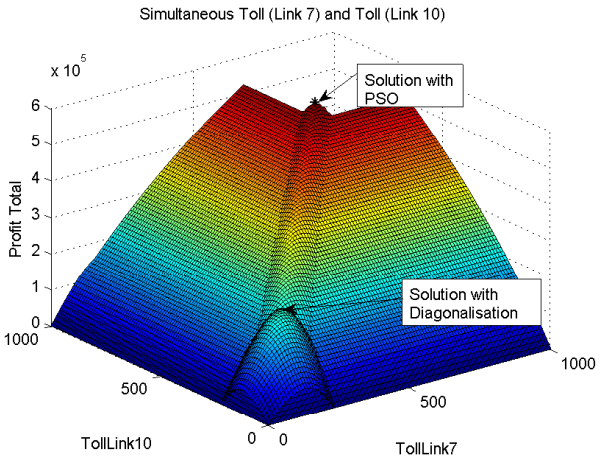



*Figure 2.* Total Revenue Surface as Tolls on Link 7 and Link 10 vary.

## 6. Conclusions and Further Research

In this paper, we applied a coevolutionary particle swarm algorithm to overcome a local NE trap defined by [16]. Our particular application showed that it is possible for players to collude by taking into account a modified objective function. Using a coevolutionary PSO algorithm, we demonstrated that it was possible to bypass the NE trap, attain the global optimum and thereby increase toll revenues for both players. A limitation of this work is the problem considered here is a game with only two players and a single strategy variable (tolls). Nevertheless there appears to be potential in applying the proposed algorithm to more difficult EPECs with increased dimension in both strategies and players. Further work on this topic is currently underway.

## Acknowledgement

This work is supported by UK EPSRC.

## Notes

1. "Control" is used as a short hand to imply that the firm has been awarded some franchise for collection of the tolls.

2. Costs of toll collection could easily be accounted for in the model but ignored here for simplicity.

3. Details of the implementation of CCA for each individual player's optimization problem can be found in [9]

## References

[1] M. Beckmann, C.B. McGuire, and C.B. Winsten. *Studies in the Economics of Transportation*. Yale University Press, New Haven, Connecticut, 1956.

[2] S.C. Dafermos. Traffic Equilibrium and Variational Inequalities. *Transport. Sci.*, 14(1):42–54, 1980.

[3] A.P. Engelbrecht. *Computational Intelligence: An Introduction*. John Wiley, New York, 2002.

[4] P.T. Harker. A variational inequality approach for the determination of Oligopolistic Market Equilibrium. *Math. Program.*, 30(1):105–111, 1984.

[5] B.F. Hobbs, C.B. Metzler, and J.S. Pang. Strategic gaming analysis for electric power networks: An MPEC approach. *IEEE T. Power Sys.*, 15(2):638-645, 2000.

[6] J. Kennedy and R.C. Eberhart. Particle swarm optimization. In *Proc. IEEE International Conference on Neural Networks*, IEEE Press, Piscataway, NJ, pages 1942–1948, 1995.

[7] A. Koh. Coevolutionary Particle Swarm Algorithm for Cournot-Nash Games. In M. Middendorf and A.P. Engelbrecht, editors, *Applied Swarm Intelligence*, Springer, submitted.

[8] A. Koh and S.P. Shepherd. Tolling, Capacity Selection and Equilibrium Problems with Equilibrium Constraints. In *Electronic Proc. 3rd Kuhmo-Nectar Conference*, Free University of Amsterdam, July 3-4, 2008.

[9] A. Koh, S.P. Shepherd, and A.S. Sumalee. Second Best Toll and Capacity Optimisation in Networks. In *Proc. 13th Conference of the Hong Kong Society for Transportation Studies*, Hong Kong, December 2007.

[10] S. Lawphongpanich and D.W. Hearn. An MPEC Approach to Second-Best Toll Pricing. *Math. Program.*, 101(1):33–55, 2004.

[11] B.S. Mordukhovich. Optimization and equilibrium problems with equilibrium constraints. *Omega-Int. J. Manage. S.*, 33(5):379–384, 2005.

[12] B.S. Mordukhovich. *Variational Analysis and Generalized Differentiation. I: Basic Theory*. Grundlehren Series in Fundamental Principles of Mathematical Sciences, Vol. 330, Springer-Verlag, Berlin, 2006.

[13] J. Nash. Equilibrium points in N-person games. *P. Natl Acad. Sci. USA*, 36(1):48–49, 1950.

[14] M.J. Smith. The existence, uniqueness and stability of traffic equilibria. *Transport. Res. B-Meth.*, 13(4):295–304, 1979.

[15] J.G. Wardrop. Some theoretical aspects of road traffic research. *P. I. Civil. Eng.-Transp.*, 1(36):325–378, 1952.

[16] S. You and R. Baldick. Hybrid coevolutionary programming for Nash equilibrium search in games with local optima. *IEEE T. Evol. Comput.*, 8(4):305–315, 2004.

# HYBRID OPTIMIZATION BASED ON FAST EVOLUTION STRATEGIES AND PARTICLE SWARM OPTIMIZATION

Chunshi Feng
*Faculty of Textile Science and Technology, Shinshu University*
*Ueda, Japan*
*and*
*Department of Automation, University of Science and Technology of China*
*Hefei, Anhui, China*

Yoshiyuki Matsumura, Takahiro Hamashima
*Faculty of Textile Science and Technology, Shinshu University*
*Ueda, Japan*
matsumu@shinshu-u.ac.jp

Kazuhiro Ohkura
*Graduate School of Engineering, Hiroshima University*
*Hiroshima, Japan*

Shuang Cong
*Department of Automation, University of Science and Technology of China*
*Hefei, Anhui, China*

**Abstract** In this paper, an empirical study on the search step size of the fast evolution strategies (FES) and particle swarm optimization (PSO) is first carried out. The results show that FES tends to generate search individuals around the neighborhoods of the current ones, and PSO is superior generate individuals in a broader search area. Inspired by the empirical conclusions, hybrid optimizations based on FES and PSO (FESPSO) are proposed. In the new hybrid algorithm, individuals of each population are divided into two groups by fitness. The first (and better) group is based on FES, and the second group is based on PSO, which are chosen mainly to take the advantage of the balance between

exploration and exploitation. Experiments are done on a set of standard benchmark functions. In order to find out the performance of the hybrid algorithm, different offspring division ratios are tested. Experimental results show that the hybrid algorithms outperform FES and PSO on part of the test function set.

**Keywords:** Fast evolution strategies (FES), FESPSO, Hybrid optimization, Particle swarm optimization (PSO)

## 1. Introduction

Evolution strategies (ESs) [2] are a class of optimization algorithms which were inspired by the natural evolution theory. There have been several variants of ESs, such as $(1+1)$-ES, $(\mu+1)$-ES, $(\mu+\lambda)$-ES, $(\mu, \lambda)$-ES, and so forth. In this paper, we focus on the $(\mu, \lambda)$-ES. Mutation is the primary evolutionary operator in $(\mu, \lambda)$-ES [1], where $\lambda > \mu \geq 1$. $(\mu, \lambda)$ means $\mu$ parents generate $\lambda$ offspring through mutation for each generation. The best $\mu$ offspring are selected and enter the next generation. Classical $(\mu, \lambda)$-ES (CES) uses Gaussian mutation. Yao and Liu introduced Cauchy mutation into $(\mu, \lambda)$-ES, called Fast Evolution Strategies (FES) [9]. FES is demonstrated to be more efficient than CES on most of the test functions.

In recent years, swarm intelligence is drawing more and more attentions in different research areas, especially in numerical optimization, multi-agent systems, and so on. It makes use of information of the whole swarm and performs search through the information sharing [3]. Although there is no centralized control dictating the individuals, information sharing often causes a global pattern to emerge. Particle swarm optimization (PSO) [5] is one of the most well-known swarm algorithms. It imitates the behavior of a flying bird flock. Individuals are called particles. They are moving in the search space, bearing locations and velocities. A global best location is shared by all the particles and every particle itself bears a local best location. A particle updates velocity according to the distance from these two locations.

Basically, natural evolution and swarm intelligence are two systems with different methodologies. But hybrid system often causes new performance to appear. As a result, hybrids based on natural evolution and swarm intelligence have been studied and proposed recently. Hsieh et al. [4] developed a particle swarm guided evolution strategies. In their method, ES adopted a new mutation operator, called guided mutation, which is inspired by the particle swarm optimization. The experiments showed some good results. Mo et al. [6] introduced a new hybrid algorithm called particle swarm assisted incremental evolution strategies.

In their study, search space is sliced by cutting planes and hyperplanes along different dimensions. PSO is used to globally adjust the planes, and ES is used to locally search the optima in those planes. In our paper, we will first study the search step size of FES and PSO. The empirical results show that FES tends to generate local search individuals around the neighborhoods of the current ones, while PSO can generate far away individuals. According to the basic optimization theory, in a generation, the best ones are suitable for exploitation, while the worst ones need further exploration. Since FES concentrates on local neighborhood, we can use it to guide the exploitation of the best individuals. On the other hand, PSO concentrates on larger neighborhood search, we can use it to lead the worst individuals to explore. Based on this methodology, we propose our new hybrid method.

The rest of this paper is organized as follows. Section 2 briefly introduces FES, PSO and their implementations. In Section 3, empirical study on the search step size are carried out. Based on the conclusion of the search step size study, our hybrid algorithm is proposed and described in Section 4. Numerical experiments on standard benchmark functions are done in Section 5 to test the performance of the hybrid algorithm, followed by some discussions. Finally, conclusions are drawn in Section 6.

## 2. Brief Introduction to FES and PSO

In this section, we give a brief background of FES and PSO, which is related to the proposed hybrid algorithms.

### 2.1 FES

Compared to classical $(\mu, \lambda)$-ES [1], FES only introduces Cauchy mutation instead of Gaussian mutation. The success of FES is explained as a result of a larger probability of escaping from local optima. It is usually implemented as follows.

1 Generate an initial population of $\mu$ individuals. Each individual is taken as a pair of real-valued vectors $(\boldsymbol{x}_i, \boldsymbol{\eta}_i), \forall i \in \{1, \cdots, \mu\}$, where $\boldsymbol{x}_i$ and $\boldsymbol{\eta}_i$ are the $i$-th coordinate values of the solution and the strategies parameters (larger than zero), respectively.

2 Evaluate the fitness for each individual $(\boldsymbol{x}_i, \boldsymbol{\eta}_i)$, $\forall i \in \{1, \cdots, \mu\}$ in the population, based on the objective function $f(\boldsymbol{x}_i)$.

3 Each parent $(\boldsymbol{x}_i, \boldsymbol{\eta}_i), i = 1, \cdots, \mu$, creates $\lambda/\mu$ offspring on average, so that a total of $\lambda$ offspring are generated. The offspring are gen-

erated as follows: for $i = 1, \ldots, \mu$, $j = 1, \ldots, n$, and $p = 1, \ldots, \lambda$,

$$\eta'_p(j) = \eta_i(j) exp\{\tau' N(0,1) + \tau N_j(0,1)\} \quad (1)$$
$$x'_p(j) = x_i(j) + \eta'_p(j)\delta_j \quad (2)$$

where $x_i(j), x'_p(j), \eta_i(j)$ and $\eta'_p(j)$ denote the $j$-th component of the vectors $\boldsymbol{x}_i, \boldsymbol{x'}_p, \boldsymbol{\eta}_i$ and $\boldsymbol{\eta'}_p$, respectively. $N(0,1)$ denotes a normally distributed one-dimensional random number with mean zero and standard deviation one. $N_j(0,1)$ indicates that the random number is generated anew for each value of $j$. $\delta_j$ is a Cauchy random number, which is generated anew for each value of $j$, too. The factors $\tau$ and $\tau'$ are commonly set to $\left(\sqrt{2\sqrt{n}}\right)^{-1}$ and $\left(\sqrt{2n}\right)^{-1}$.

4 Calculate the fitness of each offspring $(\boldsymbol{x'}_i, \boldsymbol{\eta'}_i)$, $\forall i \in \{1, \cdots, \lambda\}$, according to $f(\boldsymbol{x'}_i)$.

5 Sort offspring $(\boldsymbol{x'}_i, \boldsymbol{\eta'}_i)$, $\forall i \in \{1, \cdots, \lambda\}$ according to their fitness values, and select the $\mu$ best offspring out of $\lambda$ to be parents of the next generation.

6 If stop condition is not reached, go back to step 3.

## 2.2 PSO

In PSO, a particle moves towards two locations, one is the best location found by the particle itself, and the other is the best location found by the whole swarm. In each generation, the particle modifies its velocity and locations expecting to find the optimum. The procedure is usually implemented as follows [7].

1 Initialize the locations $x_i$, velocities $v_i$, local best locations $x_{ibest}$ and global best location $x_{gbest}$ of the particles.

2 The particles move in the search space according to:

$$v_i(k+1) = w * v_i(k) + c_1 * rand() * (x_{ibest} - x_i(k)) + c_2 * rand() * (x_{gbest} - x_i(k)) \quad (3)$$
$$x_i(k+1) = x_i(k) + v_i(k+1) \quad (4)$$

where $x_{ibest}$ is the best found ever by the $i$-th particle, $x_{gbest}$ is the best found ever by the whole swarm, $c_1$, $c_2$ are positive constants, $w$ is the inertia weight, $rand()$ is random functions in the range of $[0, 1]$. If $v_i < -V_{max}$, set $v_i = -V_{max}$, if $v_i > V_{max}$, set $v_i = V_{max}$.

3 Evaluate the fitness of the particles $f(x_i)$, and update $x_{ibest}$ and $x_{gbest}$.

4 If stop condition is not reached, go back to step 2.

## 3. Empirical Study on the Search Step Size

In [10], the authors showed the relationship between the search step size and the probability of finding a global optimum. They pointed out that when the global optimum is sufficiently far away from the current search point, i.e., when the distance between the current point and the global optimum is larger than the step size of the search, large search step size is beneficial. In order to identify the differences of the search step size of FES and PSO, we carried out an empirical study based on benchmark functions.

### 3.1 Benchmark Functions

In this empirical study, we adopt the first 13 high dimensional benchmark functions of the paper [10]. The dimension are all set to 30. Functions $f_1$ to $f_7$ are unimodal functions. $f_8$ to $f_{13}$ are multimodal functions, which have numerous local minima. The global optima are all 0, except for $f_8$, which has a global optimum of $-12569.5$. The details of these functions can be found in the appendix of [10].

### 3.2 Parameter Settings

In our study, for FES, $\mu = 30$, $\lambda = 200$. For PSO, $c_1 = c_2 = 2.0$, $w = 0.8$. $V_{max} = X_{max}$. These parameters are selected based on [1] and [8]. For the convenience of comparing, we set $population = 200$ in PSO. The search stops after 2000 generations.

Since the search space in these problems are all real number space, the Euclidean distance is a natural choice. Then the step size of individual $x$ between generation $g$ and generation $g-1$ can be defined as follows.

$$d(g) = |x(g) - x(g-1)| = \sqrt{\sum_{i=1}^{n} (x_i(g) - x_i(g-1))^2} \tag{5}$$

The average step size of generation $g$ is defined as

$$D(g) = \frac{\sum_{j=1}^{pop} d_j(g)}{pop} \tag{6}$$

*Table 1.* Average step size for the first 1000 generations

| | $f_1$ | $f_2$ | $f_3$ | $f_4$ | $f_5$ | $f_6$ | $f_7$ |
|---|---|---|---|---|---|---|---|
| $FES$ | 7.1209 | 1.7907 | 10.4878 | 22.2938 | 3.6709 | 7.8472 | 0.5727 |
| $PSO$ | 81.3812 | 8.8756 | 91.7065 | 85.0926 | 25.0895 | 89.6132 | 1.5929 |
| | $f_8$ | $f_9$ | $f_{10}$ | $f_{11}$ | $f_{12}$ | $f_{13}$ | |
| $FES$ | 34.3084 | 1.3993 | 3.7717 | 33.9971 | 6.3242 | 5.3480 | |
| $PSO$ | 1283.6025 | 4.8418 | 25.8612 | 472.5660 | 46.3836 | 41.5835 | |

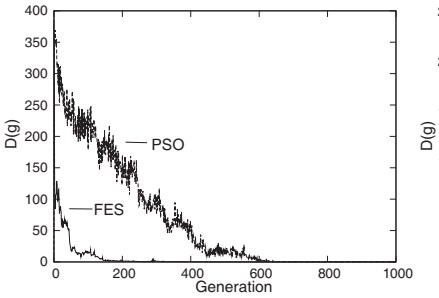



*Figure 1.* Search step size on $f_1$.

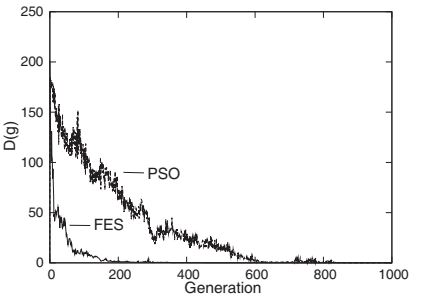



*Figure 2.* Search step size on $f_{13}$.

### 3.3 Results

After analysis, we found that the algorithms almost converge after 1000 generations, i.e., the step size almost gets down to 0. We calculate the average step size of the first 1000 generations (see *Table 1*). The step size on $f_1$ (unimodal) and $f_{13}$ (multimodal) are also shown in *Figure 1* and *Figure 2*. From *Table 1* and the figures, we can see that the search step size of PSO are larger than FES before convergence. This characteristic helps PSO be more efficient in finding global optimum, especially in the beginning of the search, where PSO can very quickly find a near-optimum area. This is especially beneficial for the unimodal functions. While for multimodal functions, PSO might fall in some local optima, because the diversity of the population decreases very quickly due to the fast convergence. As for FES, the step size is smaller than in PSO. This is inferior when the individuals are sufficiently far away from the global optimum, but when the individuals come to the very near neighborhood of the global optimum, this becomes beneficial.

From the point view of exploration and exploitation, we can say that PSO is strong at exploring, while FES is strong at exploiting. In order to achieve a balance between exploration and exploitation, we can hy-

bridize FES with PSO. In the next section, our hybrid algorithm will be described in detail.

## 4. FESPSO

There is a trade-off between exploration and exploitation, to which a good optimization algorithm should give enough attention. As for a population of individuals in an optimization algorithm, the best ones are usually exploited to search for better points in the near areas, and the worst ones need to explore to escape from local areas. Based on FES and PSO, hybrid FESPSO is proposed and implemented as follows.

1 Initialize the first generation, the individuals bear locations $x_i$, velocities $v_i$, mutation factor $\eta_i$, local best location $x_{ibest}$, global best location $x_{gbest}$.

2 Evaluate the fitness of the individuals according to $f(x_i)$, and sort the individuals by fitness.

3 Carry out particle swarm update for the last $P_{pso}$ individuals to produce $C_{pso} = P_{pso}$ offspring, and maintain their $\eta_i$ property.

4 Create $C_{fes} = \lambda - C_{pso} = \lambda - P_{pso}$ offspring out of $P_{fes} = \mu - P_{pso}$ parents. Carry out FES mutation to the offspring. Maintain their $v_i$.

5 Evaluate the fitness of the offspring, and sort the offspring by fitness. Select the best $\mu$ individuals into the next generation. Update $x_{ibest}$ and $x_{gbest}$.

6 If stop condition is not reached, go back to step 3.

For a schematic view of FESPSO one can also refer to *Figure 3*. Compared with FES and PSO, the hybrid algorithm does not increase the computational complexity.

## 5. Experiments on Standard Benchmark Functions

### 5.1 Benchmark functions

In order to test the performance of the hybrid algorithm, we adopted the same 13 high dimensional benchmark functions used in Section 3 to carry out numerical optimization experiments.

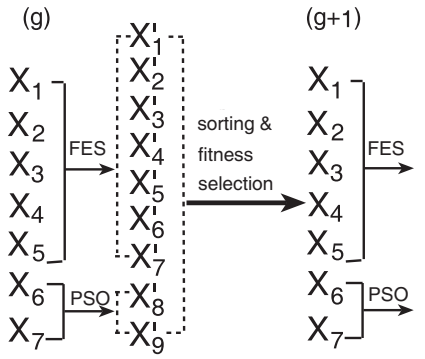



*Figure 3.* The hybrid algorithm, taking $P_{pso} = 2$, $\mu = 7$ ($P_{fes} = 5$), $\lambda = 9$ ($C_{fes} = 7$) for instance.

## 5.2 Experimental Setup

In our experiments, $\lambda = 200$, $c_1 = c_2 = 2.0$, $w = 0.8$. $V_{max} = X_{max}$. Besides, to better understand the effect of the hybrid algorithm, we tested 6 groups of offspring division ratios, $C_{fes} : C_{pso} = 200 : 00$ (FES), $C_{fes} : C_{pso} = 160 : 40$, $C_{fes} : C_{pso} = 120 : 80$, $C_{fes} : C_{pso} = 80 : 120$, $C_{fes} : C_{pso} = 40 : 160$ and $C_{fes} : C_{pso} = 00 : 200$ (PSO). When carrying out FES, we adopt $P_{fes} : C_{fes} = 3 : 20$, then $\mu$ would be different for different groups, which should be noticed. The optimization procedure stopped when the number of function evaluations reached $4.0e5$.

## 5.3 Results and Discussion

*Table 2.* Average best after $4.0e5$ function evaluations for FESPSO (average over 20 runs)

| | 200 : 00 | 160 : 40 | 120 : 80 | 80 : 120 | 40 : 160 | 00 : 200 |
|---|---|---|---|---|---|---|
| $f_1$ | 1.9552*e*–5 | 2.0929*e*–5 | 2.0724*e*–5 | 5.3193*e*–7 | 6.1640*e*–8 | 0 |
| $f_2$ | 1.8670*e*–2 | 1.9287*e*–2 | 1.9908*e*–2 | 4.4139*e*–3 | 1.2715*e*–3 | 0 |
| $f_3$ | 1.2736*e*+0 | 1.8028*e*+0 | 1.0922*e*+0 | 1.0875*e*–5 | 5.2874*e*–6 | 0 |
| $f_4$ | 2.0741*e*–1 | 4.2971*e*–1 | 9.5368*e*–1 | 1.7437*e*–1 | 3.6654*e*–2 | 0 |
| $f_5$ | 3.8773*e*+1 | 2.9178*e*+1 | 3.8976*e*+1 | 1.1200*e*+0 | 9.5771*e*–1 | 2.4572*e*+1 |
| $f_6$ | 0 | 0 | 0 | 0 | 0 | 0 |
| $f_7$ | 1.6994*e*–2 | 2.3609*e*–2 | 4.4250*e*–3 | 2.5258*e*–3 | 2.2405*e*–3 | 2.2943*e*–3 |
| $f_8$ | −1.0894*e*+4 | −1.0675*e*+4 | −1.0474*e*+4 | −1.0621*e*+4 | −1.0345*e*+4 | −8.2933*e*+3 |
| $f_9$ | 1.0495*e*+0 | 1.2023*e*+0 | 2.3566*e*+0 | 1.2969*e*+0 | 2.0620*e*+0 | 0 |
| $f_{10}$ | 3.2709*e*–3 | 3.3211*e*–3 | 3.4999*e*–3 | 5.5410*e*–4 | 1.8170*e*–4 | 7.0735*e*–5 |
| $f_{11}$ | 6.0257*e*–2 | 7.1351*e*–2 | 6.5669*e*–2 | 2.4650*e*–2 | 2.3913*e*–2 | 0 |
| $f_{12}$ | 2.5915*e*–2 | 2.0073*e*–7 | 1.5551*e*–2 | 5.1835*e*–3 | 2.0734*e*–2 | 1.1594*e*–4 |
| $f_{13}$ | 2.8010*e*–6 | 3.0405*e*–6 | 3.1736*e*–6 | 8.2011*e*–8 | 1.0024*e*–8 | 5.4824*e*–1 |

The FESPSO experimental results are shown in *Table 2*. One can see that in 5 out of 7 unimodal functions (except $f_5$ and $f_7$), PSO can detect the global optimum in every run. In $f_6$, FES, FESPSO and PSO can all find the global optimum. And in $f_5$ and $f_7$, FESPSO with the offspring division ratio of 40 : 160 wins, even though without finding the global optimum. From *Figure 4*, we can see that in the beginning of the search, PSO has a faster convergence rate than FES and FESPSO, and quickly approaches the neighborhood of the global optimum. But when the number of function evaluations reaches $8e4$, FESPSO outperforms PSO. After $1e5$ function evaluations, PSO almost cannot improve further, while FESPSO can still find better solutions. As for the multimodal functions, PSO can detect global optimum in every run in $f_9$ and $f_{11}$. FES performs best on $f_8$ and PSO on $f_{10}$, and FESPSO on $f_{12}$ (160 : 40) and $f_{13}$ (40 : 160). From *Figure 5*, one can see that hybrid algorithm converges fastest from the beginning of the search, and keeps leading until the search stops. On the other hand, PSO shows its weak point in multimodal problems where it is apt to fall in the local optimum and cannot escape.

For comparison, an experiment based on another type of hybrid, namely PSOFES, is carried out, where the best ones are dealt with PSO, and the worst ones are dealt with FES, with $C_{pso} : C_{fes} = 200 : 00/160 : 40/120 : 80/80 : 120/40 : 160/00 : 200$ (see *Table 3*). Even though the best hybrid of PSOFES outperforms the best hybrid of FESPSO on some functions, PSOFES exhibits no advantages of hybridization. It just resembles PSO, and does not improve the algorithm performance.

*Table 3.* Average best after $4.0e5$ function evaluations for PSOFES (average over 20 runs)

| | 200 : 00 | 160 : 40 | 120 : 80 | 80 : 120 | 40 : 160 | 00 : 200 |
|---|---|---|---|---|---|---|
| $f_1$ | 0 | $1.6646e{+}0$ | $2.6259e{-}1$ | $3.0809e{-}121$ | $8.8981e{-}82$ | $1.9552e{-}5$ |
| $f_2$ | 0 | $1.0847e{+}0$ | $2.9300e{+}0$ | $1.1831e{-}54$ | $5.6591e{-}43$ | $1.8670e{-}2$ |
| $f_3$ | 0 | $1.3793e{+}1$ | $7.9403e{+}0$ | $4.6423e{-}116$ | $1.6242e{-}84$ | $1.2736e{+}0$ |
| $f_4$ | 0 | $6.8629e{+}0$ | $3.7538e{+}0$ | $2.3992e{-}54$ | $5.8331e{-}42$ | $2.0741e{-}1$ |
| $f_5$ | $2.4572e{+}1$ | $1.2205e{+}2$ | $6.2856e{+}1$ | $2.6305e{+}1$ | $2.6769e{+}1$ | $3.8773e{+}1$ |
| $f_6$ | 0 | $3.9400e{+}1$ | $5.6050e{+}1$ | 0 | 0 | 0 |
| $f_7$ | $2.2943e{-}3$ | $1.3940e{-}2$ | $1.8444e{-}2$ | $4.0754e{-}6$ | $5.4167e{-}6$ | $1.6994e{-}2$ |
| $f_8$ | $-8.2933e{+}3$ | $-7.0151e{+}3$ | $-6.4353e{+}3$ | $-5.7059e{+}3$ | $-5.1792e{+}3$ | $-1.0894e{+}4$ |
| $f_9$ | 0 | $9.2976e{+}0$ | $1.7351e{+}1$ | 0 | 0 | $1.0495e{+}0$ |
| $f_{10}$ | $7.0735e{-}5$ | $8.8066e{-}1$ | $3.7739e{+}0$ | 0 | 0 | $3.2709e{-}3$ |
| $f_{11}$ | 0 | $8.4898e{-}1$ | $4.1185e{-}1$ | 0 | 0 | $6.0257e{-}2$ |
| $f_{12}$ | $1.1594e{-}4$ | $2.6147e{+}0$ | $3.8332e{+}0$ | $8.5986e{-}3$ | $4.0718e{-}2$ | $2.5915e{-}2$ |
| $f_{13}$ | $5.4824e{-}1$ | $8.6615e{+}0$ | $2.2785e{+}1$ | $2.8094e{+}0$ | $2.8453e{+}0$ | $2.8010e{-}6$ |

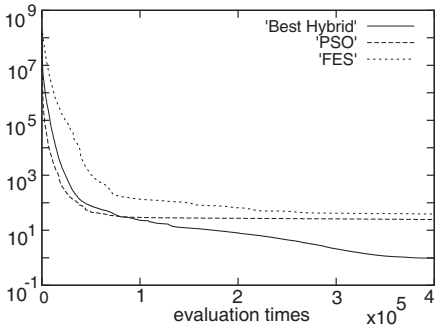



*Figure 4.* Performance on $f_5$.

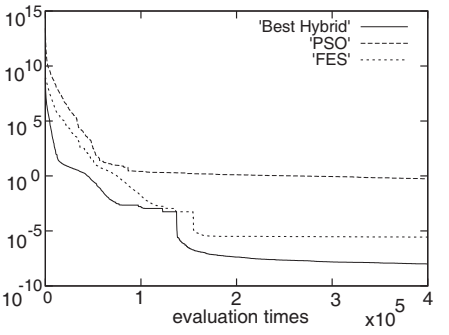



*Figure 5.* Performance on $f_{13}$.

We can conclude that PSO is the most superior algorithm for unimodal functions. But for multimodal functions, FESPSO shows superior performance and promising properties in the balance between exploration and exploitation.

## 6. Conclusions

The study of the search step size of FES and PSO demonstrates that FES is good at exploiting and PSO is strong in exploring. In order to introduce a balance between exploitation and exploration, hybrid algorithms based on FES and PSO are proposed, namely FESPSO. Experimental results indicate that the hybrid algorithms outperform on part of the benchmark functions. Especially on multimodal functions, FESPSO shows good balance between exploitation and exploration.

In our future work, the performances on other benchmark function sets will be studied, and comparisons between our hybrid algorithm and other state-of-art hybrid algorithms would be done.

## Acknowledgment

The authors acknowledge financial support in part through Grant-in-Aid for Scientific Research (19500192) and Global COE Program from the Ministry of Education, Culture, Sports, Science and Technology of Japan.

## References

[1] T. Bäck and H. Schwefel. An overview of evolutionary algorithms for parameter optimization. *Evol. Comput.*, 1(1):1–23, 1993.

[2] H. Beyer and H. Schwefel. Evolution strategies: a comprehensive introduction. *Natural Computing*, 1(1):3–52, 2002.

[3] E. Bonabeau, M. Dorigo, and G. Theraulaz. *Swarm Intelligence, From Natural to Artificial Systems*. Oxford University Press, 1999.

[4] C. Hsieh, C. Chen and Y. Chen. Particle swarm guided evolution strategy. In *Proc. 9th Annual Conference on Genetic and Evolutionary Computation*, pages 650–657, 2007.

[5] J. Kennedy and R. Eberhart. Particle swarm optimization. In *Proc. IEEE International Conference of Neural Networks*, pages 1942–1948, 1995.

[6] W. Mo, S. Guan, and S. Puthusserypady. A novel hybrid algorithm for function optimization: particle swarm assisted incremental evolution strategy. *Studies in Computational Intelligence*, Springer Berlin, 2007, pp. 101–125.

[7] Y. Shi and R. Eberhart. A modified particle swarm optimizer. In *Proc. IEEE World Congress on Computational Intelligence (WCCI 1998)*, pages 69–73, Anchorage, AK, 1998.

[8] Y. Shi and R. Eberhart. Parameter selection in particle swarm optimization. *Lect. Notes Comput. Sc.*, 1447:591–600, 1998.

[9] X. Yao and Y. Liu. Fast Evolution Strategies. *Control Cybern.*, 26(3):467–496, 1997.

[10] X. Yao, Y. Liu and G. Lin. Evolutionary Programming Made Faster. *IEEE T. Evol. Comput.*, 3(2):82–102, 1999.

# A DISTRIBUTED MULTILEVEL ANT COLONIES APPROACH FOR GRAPH PARTITIONING

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

**Abstract** The paper presents a distributed implementations of an ant colony optimization metaheuristic for the solution of a mesh partitioning problem. The usefulness and efficiency of the algorithm, in its sequential form, to solve that particular optimization problem has already been shown in previous work. In this paper a straightforward implementations on a distributed architecture is presented and the main algorithmic issues that had to be addressed are discussed. Algorithms are evaluated on a set of well known graph-partitioning problems from the Graph Collection Web page.

**Keywords:** Algorithm, Ant-colony optimization, Distributed computing, Mesh partitioning, Multilevel approach

## 1. Introduction

Real engineering problems have complex dynamic behavior and one of the widely accepted formalisms for their modeling are partial differential equations (PDEs). The fraction of PDEs that have solutions in a closed analytical form is quite small and in general their solution relies on numerical approximations. Finite-element method is a well known numerical method that efficiently solves complex PDEs problems. In order to find an approximation of an unknown solution function $f(x)$, this method discretizes the underlying domain into a set of geometrical elements consisting of nodes. This process is known as meshing. The value of the function $f(x)$ is then computed for each of these nodes, and the solutions for the other data points are interpolated from these values [4].

Generated mesh structures can have large number of elements, therefore a common approach would involve a mesh-partitioning task in order to solve the finite-element method using multiple parallel processors. Consequently, the mesh-partitioning task aims to achieve minimal interprocessor communication and at the same time to maintain a processor workload balance.

Mesh-partitioning problem is a combinatorial optimization problem. Namely, it is a special case of the well-known graph-partitioning problem, which is known to be a $NP$-hard and is defined as follows: If $G(V, E)$ denotes an undirected graph consisting of a non-empty set $V$ of vertices and a set $E \subseteq V \times V$ of edges, then $k$-partition $D$ of $G$ comprises $k$ mutually disjointed subsets $D_1, D_2, \ldots, D_k$ (domains) of $V$ whose union is $V$. The set of edges that connect the different domains of a partition $D$ is called an edge-cut. A partition $D$ is balanced if the sizes of the domains are roughly the same, i.e., if $b(D) = \max_{1 \leq i \leq k} |D_i| - \min_{1 \leq i \leq k} |D_i| \approx 0$. The graph-partitioning problem is to find a balanced partition with a minimum edge-cut, denoted by $\zeta(D)$.

Employing metaheuristic approach in optimization has introduced efficient and practical solution of many complex real-world problems. A variety of heuristic based methods are used for solving the mesh-partitioning problem as well [1, 10, 12]. In spite of being very powerful approach, metaheuristic can still easily reach the computational time limits for large and difficult problems. Moreover, heuristics do not guarantee an optimal solution, and in general their performance could depend on the particular problem setting. An important issue that arises here is not only how to design/calibrate the algorithm for a maximum performance, but also how to make it robust in terms of dealing with different types of problems and settings. Parallel processing is an straightforward approach that addresses both issues, computational time and robustness.

One relatively new and promising metaheuristic that is competitive with standard mesh-partitioning tools, such as Chaco [9], JOSTLE (that has recently been commercialised and is available under the name of NetWorks), and k-METIS [11], is known as *Multilevel Ant-Colony Algorithm* (MACA) [14]. This method is a nature inspired heuristic that uses population of agents (artificial ants) mediated by pheromone trails to find a desired goal, i.e., an ant-colony optimization algorithm [6] for solving mesh-partitioning problem. In experimental analysis so far, MACA has performed very well on different size test graph problems [14]. Since it is a population-based algorithm, MACA is inherently suitable for parallel processing on many levels. Motivated by the good performance of MACA in the previous work and the possibility to improve it's performance (computational cost and/or solution quality), in this paper we

discus the result of parallelizing MACA on largest scale, executing entire algorithm runs concurrently on a multiple instruction stream, multiple data stream (MIMD) machine architecture. Explicitly, we present and experimentally evaluate two distributed versions of MACA, the Semi-Independent Distributed MACA and the Interactive Distributed MACA approach on a set of well known graph-partitioning problems from the Graph Partitioning Archive [8]. Both distributed approaches show comparable or better (stable) quality performance. Semi-independent distributed approach can obtain same or better quality for less computational time, which is gain on both scales: quality and cost.

The rest of the paper is organized as follows: Section 2 describes the MACA algorithm for solving the mesh-partitioning problem. Section 3 outlines possible parallel strategies and in detail describes the two distributed implementations of MACA. The experimental results are presented and discussed in Section 4. Conclusions and possible directions for further work are given in Section 5.

## 2. The Multilevel Ant-Colony Algorithm

The MACA is an ant-colony algorithm [6] for $k$-way mesh (graph) partitioning enhanced with a multilevel technique [17] for global improvement of the partitioning method. The MACA is a recursive-like procedure that combines four basic methods: graph partitioning (*the basic ant-colony optimization metaheuristic*), graph contraction (*coarsening*), partitioned graph expansion (*refinement*) and *bucket sorting.*

### 2.1 The Basic Ant-Colony Algorithm

The main idea of the ant-colony algorithm for $k$-way partitioning [13] is very simple: We have $k$ colonies of ants that are competing for food, which in this case represents the vertices of the graph. Final outcome of ants activities is stored food in their nests, i.e., they partition the mesh into $k$ submeshes.

The outline of the core optimization procedure in the MACA pseudocode is given in Algorithm 1. The algorithm begins with a initialization procedure that performs a random mapping of the input graph onto a grid, which represents the place where ants can move, locates the nests position on the grid and places the ants initially in their nest locus. While gathering food, the artificial ants perform probabilistic movements on the grid in three possible directions (forward, left and right), based on the pheromone intensity. When an ant finds food, it picks it up if the quantity of the temporarily gathered food in its nest is below a specified limit (the capacity of storage is limited in order to

maintain the appropriate balance between domains); otherwise, the ant moves in a randomly selected direction. The weight of the food is calculated from the number of the cut edges created by assigning the selected vertex to the partition associated with the nest of the current ant. If the food is too heavy for one ant to pick it up then an ant sends a help signal (within a radius of a few cells) to its neighbor coworkers to help it carrying the food to the nest locus. On the way back to the nest locus an ant deposits pheromone on the trail that it is making, so the other ants can follow its trail and gather more food from that, or a nearby, cell. When an ant reaches the nest locus, it drops the food in the first possible place around the nest (in a clockwise direction)and starts a new round of foraging.

Along with foraging food, ants can gather food from other nests as well. In this case when the food is too heavy to be picked up, the ant moves on instead of sending a help signal. In this way the temporary solution is significantly improved. Furthermore, the algorithm tries to maintain a high exploration level by restoring cells pheromone intensity to the initial value whenever the pheromone intensity of a certain cell drops below a fixed value.

## 2.2 The Multilevel Framework

The multilevel framework [2] as presented in Algorithm 2 and Fig. 1 combines a level based coarsening strategy together with a level based refinement method (in reverse order) to promote faster convergence of the optimization metaheuristic and solution to a larger problems.

*Coarsening* is a graph contraction procedure that is iterated $L$ times (on $L$ levels). Adequately, a coarser graph $G_{\ell+1}(V_{\ell+1}, E_{\ell+1})$ is obtained from a graph $G_\ell(V_\ell, E_\ell)$ by finding the largest independent subset of graph edges and then collapsing them. Each selected edge is collapsed and the vertices $u_1, u_2 \in V_\ell$ that are at either end of it are merged into the new vertex $v \in V_{\ell+1}$ with weight $|v| = |u_1| + |u_2|$. The edges that have not been collapsed are inherited by the new graph $G_{\ell+1}$ and the edges that have become duplicated are merged and their weight summed. Because of the inheritance the total weight of the graph remains the same and the total edge weight is reduced by an amount equal to the weight of the collapsed edges, which have no impact on the graph balance or the edge-cut.

*Refinement* is a graph expansion procedure that applies on a partitioned graph $G_\ell$ (partitioned with the ant-colony algorithm), which interpolates it onto its parent graph $G_{\ell-1}$. Because of the simplicity of the coarsening procedure, the interpolation itself is a trivial task. Namely, if

**Algorithm 1** Ant_Colony_Algorithm

```
1: Initialize()
2: while ending condition not satisfied do
3:     for all ants of colony do
4:         for all colonies do
5:             if carrying food then
6:                 if in nest locus then
7:                     Drop_Food()
8:                 else
9:                     Move_to_Nest()
10:                end if
11:            else
12:                if food here then
13:                    Pick_Up_Food()
14:                else
15:                    if food ahead then
16:                        Move_Forward()
17:                    else
18:                        if in nest locus then
19:                            Move_To_Away_Pheromone()
20:                        else
21:                            if help signal then
22:                                Move_To_Help()
23:                            else
24:                                Follow_Strongest_Forward_Pheromone()
25:                            end if
26:                        end if
27:                    end if
28:                end if
29:            end if
30:        end for
31:    end for
32:    for all grid cells do
33:        Evaporate_Pheromone()
34:    end for
35: end while
```

a vertex $v \in V_\ell$ belongs to the domain $D_i$, then after the refinement the matched pair $u_1, u_2 \in V_{\ell-1}$ that represents the vertex $v$, will also be in the domain $D_i$. In this way we expand the graph to its original size, and on every level $\ell$ of our expansion we run our basic ant-colony algorithm.

**Algorithm 2** Multilevel_Framework

```
1: structure[0] = Initialization()
2: for ℓ = 0 to L − 1 do
3:     structure[ℓ + 1] = Coarsening(structure[ℓ])
4: end for
5: for ℓ = L downto 0 do
6:     Solver(structure[ℓ])
7:     if ℓ > 0 then
8:         structure[ℓ − 1] = Refinement(structure[ℓ])
9:     end if
10: end for
```

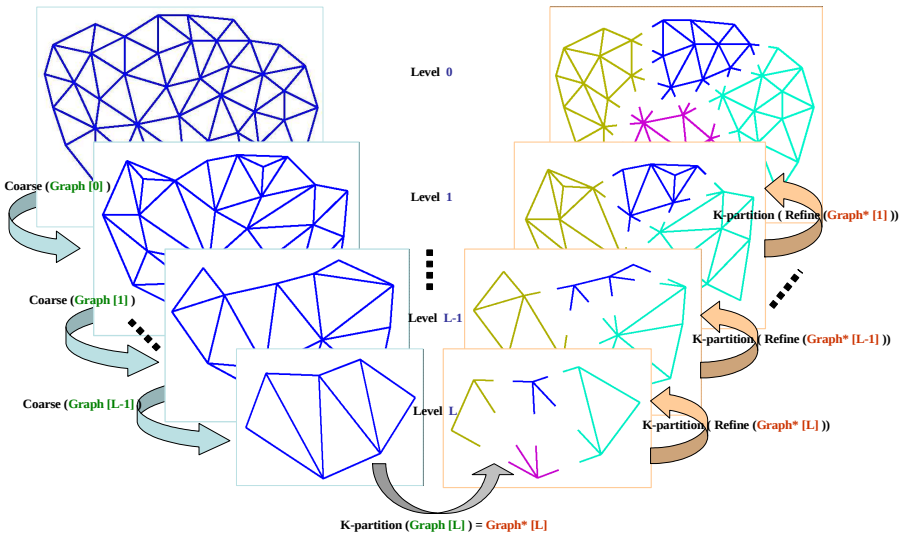



*Figure 1.* The three phases of multilevel $k$-way graph partitioning.

Large graph problems and the multilevel process by itself induce rapid increase of the number of vertices in a single cell as the number of levels goes up. To overcome this problem MACA employs a method, based on the basic *bucket sort* idea [7], that accelerates and improves the algorithm's convergence by choosing the most "promising" vertex from a given cell. Inside the cell, all vertices with a particular gain $g$ are put together in a "bucket" ranked $g$ and all nonempty buckets, implemented as double-linked list of vertices, are organized in a 2-3 tree. Additionally, MACA keeps separate 2–3 tree for each colony on every grid cell that has vertices in order to gain even faster searches.

## 3. Distributed Multilevel Ant-Colony Approaches

In general, ant-colony optimization algorithms can be parallelized on four different levels [5, 15, 16], as follows: (i) *parallelism on colony level*, (ii) *parallelism on ant level*, (iii) *data level parallelization*, and (iv) *functional parallelization*, where each one is differing in granularity and communication overhead between processors. We will in brief, in the first subsection, describe all four parallelization approaches, making a ground base for introduction of the proposed Semi-Independent Distributed MACA and Interactive Distributed MACA approaches in the second, and the third subsection, respectively.

### 3.1 Parallelization Strategies

(i) *Parallelism on colony level* is the most simple coarse-grained parallelization of the ant-colony optimization algorithms, where the problem is instantiated and solved simultaneously on all available processors. Furthermore, if no communication is required between processors (parallel independent algorithms searches, introduced by Stützle [16]), then this approach is refereed to as *parallel independent ant colonies* and it is suitable for algorithms that perform stochastic searches. Otherwise, if colonies, while searching for food, exchange information at a specified iteration (requires synchronized communication which implies master/slave implementation), then we refer to this approach as *parallel interactive ant colonies*. The communication cost of the second approach can become very expensive due to the required broadcasting of entire pheromone structures.

(ii) *Parallelism on ant level* is the first proposed parallel implementation [3] of an ant-colony optimization algorithm, where each ant (or a group of ants) is assigned a separate processor to build a solution. This means maintenance of a separate pheromone structures on every processor and therefore this approach requires a master processor that will synchronize the work of the rest (slave processors), including ant-processor scheduling, initializations, global pheromone updates and producing of the final solution.

(iii) *Data level parallelization* is a suitable approach for solving the multi-objective optimization problems, since it divides the main problem into a number of subproblems (objectives to optimize) and each one is solved by a colony on a separate processor.

(iv) *Functional parallelization* is a parallelization that introduces a concurrent execution of a specified operations (local search, solution construction, solution evaluation) performed by a single colony on a

master-slave architecture. When local heuristic searches are computationally expensive, a so-called *parallel local searches* are the preferred case. In particular, the assignment of a slave processor is to refine the solutions received from the master with local search heuristics, while the master is responsible for a solution construction, pheromone updates and collection of the refined solutions. The *parallel solution construction* is a second approach that organizes the available slave processors in two operational groups. Processors in the first one are responsible for a solution construction, while the second group processors are additionally grouped and scheduled to refine the corresponding solutions constructed by the first group processors. The last functional parallelization approach is called *parallel evaluation of solution elements*. This approach gives best performance in case of a computationally expensive solution evaluations. Compared to all aforementioned parallel strategies parallel evaluation of solution elements is the only approach that does not exploits parallelism within the metaheuristic algorithm.

An efficient parallelization of a given algorithm depends mainly on the available computing platform, the underlying problem and the algorithm itself. If there is a large communication overhead between the processors, then parallel performance can be degraded. When the algorithms uses global structures, such as the pheromone matrix or the grid matrix of 2–3 trees in MACA case, a shared memory system would gain on communication (less) over a distributed memory system. On the other hand, the most common and cheaper approach in the same time is a parallelization using distributed memory systems, i.e., MIMD architecture such as cluster of workstations. Our proposed MACA parallelization assumes distributed memory system as well and it is implemented on a cluster of workstations.

## 3.2 The Semi-Independent Distributed MACA

The Semi-Independent Distributed MACA (SIDMACA) is basically a distributed MACA approach that allows exchange of the best temporal solution at the end of every level of the multilevel optimization process. This exchange requires that the parallel executions of MACA instances on the available processors have to be synchronized once per level. Namely, the master processor is responsible for synchronizing the work of all slave processor that execute a copy of MACA, by managing the exchange information and communication process (sending commands and confirmation, such as Start, Stop, Initialize, Goto New Level, Best Partition, etc.), while the slave processors have to execute the instances of the MACA code, signal when finish the current level

optimization and send the best partition to the master. When all finish the current level, the master determines the best solution and broadcasts it to the slaves. In order to proceed with next level optimization, slave processors have to first update local memory structures (grid matrix) and afterwards perform partition expansion (refinement).

### 3.3 The Interactive Distributed MACA

The Interactive Distributed MACA (ItDMACA) is based on the parallel interactive colony approach which, by definition, implies master/slave implementation and synchronized communication. The information exchange between the colonies across the concurrent processors is initiated every time a piece of food has been taken or dropped on a new position. The information about the specific food, its new position and its owner is part of the message sent to and received from the master processor when picked up or dropped food. The master keeps and updates its own local grid matrix of temporal food positions (plays the role of shared memory) in order to maintain normal and consistent slaves activities.

The master processors is responsible for the synchronized work and communication of the slave processors, which includes listening, processing and broadcasting of the incoming clients messages during level optimization. When all slave processors finish level or run, it collects the best-level solution, determines and broadcast the global best-level solution to the slaves and guides them when to start the refinement procedure and all necessary updates in order to perform the next level optimization activities or a new run.

A slave processor executes a single instance of the MACA code. While optimization executing informs the master and waits for master's confirmation on every potential drop/pick, signals when finishes the current level optimization and send the best partition to the master. In the meantime, while waiting to go on the next level, it listens for an eventual changes send by the unfinished clients and performs the eventual updates on the grid. When the master signals that the current level is finished, by sending the new best temporal solution, the slave processor has to perform partition expansion (refinement) in order to start the next level optimization.

## 4. Experimental Evaluation

The proposed distributed versions of MACA were applied on a set of well-known graph problems and the results from their experimental evaluation are presented and discussed in this section. The section is structured in two subsection. The first subsection describes the im-

plementation of the distributed code, the experimental setting and the benchmark suite, whereas the second subsection presents and discusses the evaluation results.

## 4.1 Setup

Based on the MACA sequential code, both proposed distributed version, SIDMACA and ItDMACA, were implemented in Borland® Delphi™, using TCP/IP protocol for the server/client communication, based on the open source library Indy Sockets 10 (which supports clients and servers of TCP, UDP and RAW sockets as well as over 100 higher level protocols).

All experiments were performed on a 8-node cluster connected via a Giga-bit switch, where each node consists of two AMD Opteron™1.8-GHz processors, 2GB of RAM, and Microsoft®Windows®XP operating system.

The benchmark graphs used in the experimental analysis were taken from the Graph Collection Web page [8], and are described in Table 1.

*Table 1.* Benchmark graphs

| **Graph** $G(V,E)$ | $\|V\|$ | $\|E\|$ | **Graph** $G(V,E)$ | $\|V\|$ | $\|E\|$ |
|---|---|---|---|---|---|
| ***grid1*** | 252 | 476 | ***U1000.05*** | 1000 | 2394 |
| ***grid2*** | 3296 | 6432 | ***U1000.10*** | 1000 | 4696 |
| ***crack*** | 10240 | 30380 | ***U1000.20*** | 1000 | 9339 |

*Table 2.* Distribution of ants per colonies and number of iterations per level w.r.t the number of processors

| **Parameters** | **Number of processors** | | | | |
|---|---|---|---|---|---|
| | **1** | **2** | **4** | **6** | **8** |
| ants/colony | 120 | 60 | 30 | 20 | 15 |
| iteration/level | 600 | 600 | 600 | 600 | 600 |
| runs | 20 | 20 | 20 | 20 | 20 |

The total number of ants per colony was 120. As presented in Table 2, the number of ants per sub-colony is different and depends on the number $p$ of processors, i.e., $\frac{1}{p}$ of the total number of ants, while the number of total iterations per level per colony is constant.

All experiments were run 20 times on each graph with each algorithm and as final results were presented the mean value (also best and worst

values for edge-cut) of the considered evaluation criteria over all performed runs.

### 4.2 Results

The results presented in the following tables show the performance of the introduced DMACA approaches on the 2-partitioning and 4-partitioning graph problem. The *quality* of the partitioned graph is described with the edge-cut, $\zeta(D)$, and the balance, $b(D)$. Balance is defined as the difference (in the number of vertices) between the largest and the smallest domain.

Beside the quality, the second evaluation criteria is the effectiveness of the parallel algorithm which is in our case given by the *speed-up* measure, $S$, which is defined as:

$$S(p) = \frac{t_{\mathrm{S}}}{t_{\mathrm{T}}(p)}$$

and by the *relative speed-up* measure, $S_r$, which is defined as:

$$S_r(p) = \frac{t_{\mathrm{T}}(1)}{t_{\mathrm{T}}(p)},$$

where $t_{\mathrm{S}}$ is the time to solve a problem with the sequential code, $t_{\mathrm{T}}(1)$ is time to solve a problem with the parallel code with the one processor, and $t_{\mathrm{T}}(p)$ is time to solve the same problem with the parallel code on $p$ processors. Note that $S(p)$ and $S_r(p)$ were calculated based on the average time values of the 20 runs.

By theory, correct speed-up metric should be calculated according to the performance (elapsed computational time) of the best serial code for the underlying algorithm, as defined above and denoted with $S(p)$, whereas in practice this is usually translated into calculation of the relative speed-up metric $S_r(p)$, since the best serial code is not available and writing two codes is not acceptable. In our case the serial code is available, and the values of both speed-up metrics are included in the tables with results.

Additionally, for the reason of comparison, in the tables are given the measured *CPU time* for the computation of the obtained solutions, $t_{\mathrm{T}}$, as a triple of the time spent on pure computations, the time for communication with the master processor, $t_{\mathrm{C}}$, and the time for internal updates caused by the synchronization, $t_{\mathrm{U}}$. Note that $t_{\mathrm{C}}$ and $t_{\mathrm{U}}$ are part of the $t_{\mathrm{T}}$ spent for communication and updates, respectively.

Results in in Table 3 and Table 4 summarize the performance of SIDMACA for solving 2-partitioning and 4-partitioning graph problem, respectively, on the given graph set.

*Table 3.* Experimental results: 2-partitioning problem with SIDMACA

| | | **Quality** | | | | **Time** [s] | | **Speed-up** | |
|---|---|---|---|---|---|---|---|---|---|
| | | $\zeta(D)$ | | | $b(D)$ | $t_T$ | $t_C$ | $S(p)$ | $S_r(p)$ |
| **Graph** | $p$ | best | mean | worst | mean | mean | mean | mean | mean |
| | 1* | 18 | 18 | 18 | 0 | 9.80 | 0 | 1.00 | |
| ***grid1*** | 1 | 18 | 18 | 18 | 0 | 10.03 | 0.10 | | 1.00 |
| | 2 | 18 | 18 | 18 | 0 | 10.41 | 0.69 | 0.94 | 0.96 |
| | 4 | 18 | 18 | 19 | 0 | 10.00 | 2.67 | 0.98 | 1.00 |
| | 6 | 18 | 18 | 21 | 0 | 7.17 | 1.75 | 1.37 | 1.40 |
| | 8 | 18 | 19 | 21 | 0 | 5.60 | 1.44 | 1.75 | 1.79 |
| | 1* | 35 | 44 | 68 | 0 | 20.37 | 0 | 1.00 | |
| ***grid2*** | 1 | 34 | 41 | 68 | 0 | 23.81 | 0.10 | | 1.00 |
| | 2 | 35 | 40 | 69 | 0 | 23.28 | 1.58 | 0.88 | 1.02 |
| | 4 | 35 | 40 | 69 | 0 | 15.86 | 2.99 | 1.28 | 1.50 |
| | 6 | 35 | 41 | 70 | 0 | 11.73 | 2.51 | 1.74 | 2.03 |
| | 8 | 35 | 49 | 70 | 0 | 9.31 | 2.22 | 2.19 | 2.56 |
| | 1* | 1 | 1 | 2 | 0 | 87.53 | 0 | 1.00 | |
| ***U1000.05*** | 1 | 1 | 1 | 3 | 0 | 88.80 | 0.39 | | 1.00 |
| | 2 | 1 | 1 | 2 | 0 | 83.10 | 1.81 | 1.05 | 1.07 |
| | 4 | 1 | 1 | 1 | 0 | 60.86 | 4.54 | 1.44 | 1.46 |
| | 6 | 1 | 1 | 1 | 0 | 42.15 | 6.09 | 2.08 | 2.11 |
| | 8 | 1 | 1 | 1 | 0 | 32.19 | 6.16 | 2.72 | 2.76 |
| | 1* | 50 | 62 | 78 | 1 | 14.49 | 0 | 1.00 | |
| ***U1000.10*** | 1 | 39 | 62 | 73 | 1 | 15.03 | 0.17 | | 1.00 |
| | 2 | 40 | 59 | 76 | 1 | 14.97 | 1.16 | 0.97 | 1.00 |
| | 4 | 40 | 59 | 71 | 1 | 11.88 | 2.57 | 1.22 | 1.27 |
| | 6 | 50 | 61 | 72 | 1 | 8.72 | 1.95 | 1.66 | 1.72 |
| | 8 | 57 | 61 | 72 | 1 | 7.12 | 1.76 | 2.04 | 2.11 |
| | 1* | 221 | 277 | 370 | 8 | 12.14 | 0 | 1.00 | |
| ***U1000.20*** | 1 | 221 | 256 | 337 | 6 | 13.03 | 0.15 | | 1.00 |
| | 2 | 219 | 259 | 373 | 7 | 12.48 | 0.99 | 0.97 | 1.04 |
| | 4 | 219 | 266 | 369 | 7 | 10.67 | 2.51 | 1.14 | 1.22 |
| | 6 | 219 | 288 | 368 | 10 | 7.75 | 1.75 | 1.58 | 1.68 |
| | 8 | 219 | 278 | 370 | 9 | 6.05 | 1.34 | 2.01 | 2.15 |
| | 1* | 185 | 211 | 234 | 1 | 64.91 | 0 | 1.00 | |
| ***crack*** | 1 | 184 | 204 | 277 | 1 | 85.02 | 0.29 | | 1.00 |
| | 2 | 184 | 195 | 231 | 0 | 80.48 | 6.28 | 0.81 | 1.06 |
| | 4 | 185 | 203 | 246 | 0 | 52.25 | 10.40 | 1.24 | 1.63 |
| | 6 | 186 | 202 | 230 | 0 | 39.70 | 9.25 | 1.64 | 2.14 |
| | 8 | 185 | 203 | 225 | 0 | 32.04 | 8.45 | 2.03 | 2.65 |

* sequential code

*Table 4.* Experimental results: 4-partitioning problem with SIDMACA

| | | **Quality** | | | | **Time** [s] | | **Speed-up** | |
|---|---|---|---|---|---|---|---|---|---|
| | | $\zeta(D)$ | | | $b(D)$ | $t_T$ | $t_C$ | $S(p)$ | $S_r(p)$ |
| **Graph** | $p$ | best | mean | worst | mean | mean | mean | mean | mean |
| | 1* | 38 | 39 | 41 | 1 | 18.01 | 0 | 1.00 | |
| ***grid1*** | 1 | 38 | 39 | 42 | 1 | 19.67 | 0.08 | | 1.00 |
| | 2 | 38 | 39 | 41 | 0 | 19.38 | 0.67 | 0.93 | 1.01 |
| | 4 | 38 | 39 | 41 | 0 | 18.12 | 2.72 | 0.99 | 1.09 |
| | 6 | 38 | 39 | 41 | 0 | 13.50 | 1.69 | 1.33 | 1.46 |
| | 8 | 38 | 39 | 41 | 0 | 10.42 | 1.14 | 1.73 | 1.89 |
| | 1* | 96 | 104 | 118 | 4 | 47.38 | 0 | 1.00 | |
| ***grid2*** | 1 | 95 | 97 | 111 | 3 | 59.89 | 0.21 | | 1.00 |
| | 2 | 94 | 106 | 116 | 3 | 65.21 | 4.78 | 0.73 | 0.92 |
| | 4 | 92 | 105 | 123 | 2 | 47.96 | 10.14 | 0.99 | 1.25 |
| | 6 | 93 | 106 | 116 | 2 | 35.35 | 8.18 | 1.34 | 1.69 |
| | 8 | 93 | 103 | 115 | 2 | 28.16 | 6.89 | 1.68 | 2.13 |
| | 1* | 9 | 14 | 20 | 3 | 50.78 | 0 | 1.00 | |
| ***U1000.05*** | 1 | 7 | 14 | 22 | 2 | 57.88 | 0.20 | | 1.00 |
| | 2 | 9 | 14 | 23 | 2 | 60.96 | 8.28 | 0.83 | 0.95 |
| | 4 | 7 | 14 | 21 | 1 | 45.15 | 12.00 | 1.12 | 1.28 |
| | 6 | 8 | 13 | 18 | 0 | 36.26 | 9.99 | 1.40 | 1.60 |
| | 8 | 7 | 11 | 17 | 0 | 36.03 | 11.15 | 1.41 | 1.61 |
| | 1* | 95 | 114 | 166 | 3 | 35.17 | 0 | 1.00 | |
| ***U1000.10*** | 1 | 102 | 113 | 133 | 3 | 43.09 | 0.12 | | 1.00 |
| | 2 | 98 | 110 | 133 | 2 | 40.76 | 2.89 | 0.86 | 1.06 |
| | 4 | 92 | 112 | 163 | 2 | 39.03 | 10.12 | 0.90 | 1.10 |
| | 6 | 101 | 113 | 162 | 2 | 27.14 | 6.64 | 1.30 | 1.59 |
| | 8 | 91 | 115 | 161 | 3 | 19.96 | 4.64 | 1.76 | 2.16 |
| | 1* | 485 | 580 | 856 | 6 | 32.27 | 0 | 1.00 | |
| ***U1000.20*** | 1 | 479 | 592 | 838 | 6 | 36.77 | 0.13 | | 1.00 |
| | 2 | 485 | 586 | 817 | 6 | 36.19 | 1.85 | 0.89 | 1.02 |
| | 4 | 490 | 593 | 687 | 5 | 32.29 | 7.85 | 1.00 | 1.14 |
| | 6 | 490 | 632 | 730 | 6 | 22.65 | 4.70 | 1.42 | 1.62 |
| | 8 | 491 | 649 | 727 | 8 | 17.02 | 3.34 | 1.90 | 2.16 |
| | 1* | 373 | 415 | 522 | 15 | 191.07 | 0 | 1.00 | |
| ***crack*** | 1 | 374 | 425 | 496 | 14 | 259.03 | 0.27 | | 1.00 |
| | 2 | 377 | 426 | 495 | 11 | 217.40 | 14.39 | 0.88 | 1.19 |
| | 4 | 373 | 423 | 506 | 8 | 139.52 | 25.92 | 1.34 | 1.86 |
| | 6 | 384 | 431 | 493 | 6 | 109.29 | 23.66 | 1.75 | 2.37 |
| | 8 | 378 | 429 | 526 | 6 | 83.35 | 18.40 | 2.29 | 3.11 |

* sequential code

General observation is that parallel performance of the system w.r.t speed-up over the serial MACA is poor compared to the theoretical expected speed-up of $p$ when used $p$ processors, having maximal speed-up of 2.29 (graph ***crack***, $p = 8$) in case of 2-partitioning problem and maximal speed-up of 2.72 (graph ***U1000.05***, $p = 8$) in case of 4-partitioning problem overall considered graphs and parallel scenarios ($p = 2, 4, 6, 8$). For more then 2 processors employed $S > 1$ (except for the graph ***U1000.10***, $p = 4$, $k = 4$), while for 2-processor parallelization of the problems is evident speed-down up to 27% in case of 4-partitioning of graph ***grid2***. On the other side, results on SIDMACA show overall comparable/improved quality of the obtained solutions. The best solutions found in case of 2-partitioning are equal or better then the best serial code produced solutions (except for graph ***U1000.10***, $p = 4$ and ***crack***, $p = 6$). Moreover, the worst solutions found by SIDMACA are better than the ones from the MACA on the ***U1000*** graph set and ***crack*** graph. When solved the 4-partitioning problem, best found solution is better than the best ones from the serial code are observed for graphs: ***grid2***, ***U1000.05*** and ***U1000.10***. The remark on the better quality of the worst case found solutions is confirmed in case of graphs ***U1000.10***, ***U1000.10*** and partially for graphs ***grid2***, ***U1000.05*** and ***crack***.

Correspondingly, Table 5 and Table 6 illustrate the ItDMACA performance on the same graph set for the 2- and 4-partitioning graph problems when 2, 4 and 8 processor employed in parallel. Note that for $p = 8$ results are available only for the graphs ***grid1***, ***U1000.10*** and ***U1000.20***.

The results show no speed-up in case of 2-processor and 4-processor parallelization. Speed-up $S \geq 1$ is evident when 8 processor applied on the graphs for solving the 4-partitioning problem and for 2-partitioning of graphs ***U1000.10***, ***U1000.20***. Speed-down and low speed-ups are due to the big amount of time spent on communication and memory updates (synchronizations) during level optimization activities. The performance of ItDMACA w.r.t the quality of obtained solutions confirms the observation from the SIDMACA results. More specifically for the 2-partitioning problem, equal partition solutions in all runs are obtained for graphs ***grid1*** and ***U1000.05***, while significant improvement is evident for the ***U1000.10***, and slightly better solution for the rest of the graphs.

In general, comparable or improved solution quality is observed in the case of solving the 4-partitioning problem with ItDMACA as well. For $p = 8$, we gain speed-up and (i) better solution for graph ***U1000.20***, (ii) equal best found solution for graph ***grid1***, (iii) comparable solutions for graph ***U1000.10***.

*Table 5.* Experimental results: 2-partitioning problem with ItDMACA

| | | **Quality** | | | | **Time** [s] | | | **Speed-up** | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | $\zeta(D)$ | | | $b(D)$ | $t_T$ | $t_C$ | $t_U$ | $S(p)$ | $S_r(p)$ |
| **Graph** | $p$ | best | mean | worst | mean | mean | mean | mean | mean | mean |
| | 1* | 18 | 18 | 18 | 0 | 9.80 | 0 | 0 | 1.00 | |
| ***grid1*** | 1 | 18 | 18 | 18 | 0 | 44.79 | 34.45 | 0.11 | | 1.00 |
| | 2 | 18 | 18 | 18 | 0 | 37.61 | 17.87 | 1.54 | 0.26 | 1.19 |
| | 4 | 18 | 18 | 18 | 0 | 18.86 | 8.26 | 3.01 | 0.52 | 2.38 |
| | 8 | 18 | 18 | 18 | 0 | 11.83 | 5.01 | 3.00 | 0.83 | 3.79 |
| | 1* | 35 | 44 | 68 | 0 | 20.37 | 0 | 0 | 1.00 | |
| ***grid2*** | 1 | 35 | 45 | 69 | 0 | 143.34 | 34.45 | 0.11 | | 1.00 |
| | 2 | 34 | 42 | 68 | 0 | 92.74 | 56.55 | 9.08 | 0.22 | 1.55 |
| | 4 | 35 | 39 | 53 | 0 | 52.29 | 26.57 | 13.44 | 0.39 | 2.74 |
| | 1* | 1 | 1 | 2 | 0 | 87.53 | 0 | 0 | 1.00 | |
| ***U1000.05*** | 1 | 1 | 1 | 2 | 0 | 463.82 | 373.45 | 0.32 | | 1.00 |
| | 2 | 1 | 1 | 2 | 0 | 295.38 | 190.25 | 41.64 | 0.30 | 1.57 |
| | 4 | 1 | 1 | 2 | 0 | 182.04 | 90.89 | 61.65 | 0.48 | 2.55 |
| | 1* | 50 | 62 | 78 | 1 | 14.49 | 0 | 0 | 1.00 | |
| ***U1000.10*** | 1 | 39 | 60 | 76 | 1 | 44.47 | 27.11 | 0.20 | | 1.00 |
| | 2 | 39 | 63 | 77 | 1 | 30.27 | 11.54 | 4.58 | 0.48 | 1.47 |
| | 4 | 40 | 59 | 71 | 1 | 21.16 | 6.63 | 4.77 | 0.68 | 2.10 |
| | 8 | 40 | 59 | 70 | 1 | 14.73 | 4.45 | 4.32 | 0.98 | 3.02 |
| | 1* | 221 | 277 | 370 | 8 | 12.14 | 0 | 0 | 1.00 | |
| ***U1000.20*** | 1 | 219 | 268 | 373 | 7 | 24.41 | 11.23 | 0.15 | | 1.00 |
| | 2 | 219 | 272 | 371 | 8 | 21.83 | 5.43 | 2.58 | 0.56 | 1.12 |
| | 4 | 219 | 255 | 368 | 7 | 16.48 | 2.77 | 3.92 | 0.74 | 1.48 |
| | 8 | 235 | 262 | 308 | 5 | 10.65 | 1.73 | 3.14 | 1.14 | 2.29 |
| | 1* | 185 | 211 | 234 | 1 | 64.91 | 0 | 0 | 1.00 | |
| ***crack*** | 1 | 184 | 191 | 262 | 0 | 312.25 | 205.08 | 0.42 | | 1.00 |
| | 2 | 184 | 189 | 211 | 0 | 223.60 | 95.82 | 57.03 | 0.29 | 1.40 |
| | 4 | 184 | 187 | 207 | 0 | 150.94 | 48.21 | 63.33 | 0.43 | 2.07 |

* sequential code

As expected, the results on relative speed-up $S_r(p)$ are better than the speed-up $S(p)$ results. How big this difference is, is dependent on the size of the problem and algorithm implementation. Consequently, for SIDMACA the difference is not significant (except for graph ***grid2*** and ***crack***) compared to the ones in the ItDMACA, which in case of the ***grid2*** graph yields 7 times higher $S_r(p)$ than $S(p)$. This difference reveals that ItDMACA suffers from communication/update overhead, which for specific problems could be disadvantageous.

*Table 6.* Experimental results: 4-partitioning problem with ItDMACA

| | | **Quality** | | | | **Time** [s] | | | **Speed-up** | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | $\zeta(D)$ | | | $b(D)$ | $t_T$ | $t_C$ | $t_U$ | $S(p)$ | $S_r(p)$ |
| **Graph** | $p$ | best | mean | worst | mean | mean | mean | mean | mean | mean |
| | 1* | 38 | 39 | 41 | 1 | 18.01 | 0 | 0 | 1.00 | |
| ***grid1*** | 1 | 38 | 40 | 42 | 0 | 58.03 | 38.39 | 0.28 | | 1.00 |
| | 2 | 38 | 40 | 43 | 0 | 47.97 | 16.25 | 1.38 | 0.38 | 1.21 |
| | 4 | 38 | 39 | 41 | 1 | 26.45 | 11.07 | 3.35 | 0.68 | 2.19 |
| | 8 | 38 | 40 | 43 | 1 | 14.00 | 5.43 | 2.22 | 1.29 | 4.15 |
| | 1* | 96 | 104 | 118 | 4 | 47.38 | 0 | 0 | 1.00 | |
| ***grid2*** | 1 | 95 | 101 | 116 | 3 | 318.27 | 248.53 | 0.85 | | 1.00 |
| | 2 | 95 | 103 | 114 | 4 | 210.78 | 110.65 | 45.41 | 0.22 | 1.51 |
| | 4 | 95 | 105 | 118 | 4 | 132.13 | 66.09 | 30.91 | 0.36 | 2.41 |
| | 1* | 9 | 14 | 20 | 3 | 50.78 | 0 | 0 | 1.00 | |
| ***U1000.05*** | 1 | 9 | 14 | 21 | 3 | 342.12 | 278.76 | 0.62 | | 1.00 |
| | 2 | 7 | 16 | 33 | 3 | 225.56 | 134.97 | 37.33 | 0.23 | 1.52 |
| | 4 | 7 | 15 | 22 | 2 | 160.29 | 91.36 | 38.15 | 0.32 | 2.13 |
| | 1* | 95 | 114 | 166 | 3 | 35.17 | 0 | 0 | 1.00 | |
| ***U1000.10*** | 1 | 93 | 116 | 159 | 3 | 79.74 | 34.02 | 0.53 | | 1.00 |
| | 2 | 96 | 112 | 129 | 3 | 63.46 | 17.23 | 7.32 | 0.55 | 1.26 |
| | 4 | 98 | 117 | 197 | 5 | 46.29 | 8.99 | 9.70 | 0.76 | 1.73 |
| | 8 | 98 | 118 | 157 | 3 | 26.08 | 5.13 | 7.34 | 1.35 | 3.06 |
| | 1* | 485 | 580 | 856 | 6 | 32.27 | 0 | 0 | 1.00 | |
| ***U1000.20*** | 1 | 480 | 594 | 888 | 8 | 63.64 | 25.31 | 0.49 | | 1.00 |
| | 2 | 487 | 583 | 759 | 6 | 51.82 | 11.07 | 4.39 | 0.62 | 1.22 |
| | 4 | 486 | 594 | 762 | 5 | 36.72 | 6.65 | 7.51 | 0.88 | 1.73 |
| | 8 | 474 | 584 | 805 | 5 | 24.25 | 3.52 | 6.50 | 1.33 | 2.62 |
| | 1* | 373 | 415 | 522 | 15 | 191.07 | 0 | 0 | 1.00 | |
| ***crack*** | 1 | 372 | 415 | 507 | 15 | 720.23 | 401.47 | 1.11 | | 1.00 |
| | 2 | 377 | 433 | 496 | 11 | 565.13 | 194.86 | 150.72 | 0.34 | 1.27 |
| | 4 | 382 | 415 | 492 | 9 | 411.00 | 104.70 | 197.98 | 0.46 | 1.75 |

* sequential code

Additional experiments are needed in order to confirm the conclusions drawn from the initial experimental evaluations results, based on small number of processing nodes and a small set of graphs. There is a large space with possible directions for further work, such as:

- application on additional new graph problems, specially large and complex ones,

- try to solve the partitioning problem with more than 8 processors in parallel and find how the number of processors influences the solution quality and speed-up,
- shared memory implementation, since distributed implementations suffer from increased communication and local memory updates,
- how statistically significant is the difference in the performances of the proposed parallel implementations among them or/and vs. the sequential MACA algorithm.

## 5. Conclusions

An efficient parallelization of a given algorithm depends mainly on the available computing platform, the underlying problem and the algorithm itself. If there is a large communication overhead between the processors, then parallel performance can be degraded. When the algorithms uses global structures, such as the pheromone matrix or the grid matrix of 2–3 trees in MACA case, a shared memory system would gain on communication (less) over a distributed memory system. On the other hand, the most common and cheaper approach in the same time is a parallelization using distributed memory systems, i.e., MIMD architecture such as cluster of workstations.

In this paper, two distributed MACA versions were presented, Semi-Independent and Interactive, implemented on a cluster of workstations. The initial experimental evaluations confirms that parallelization efficiency is problem dependent. Overall, both approaches show comparable or better (stable) quality performance. While ItDMACA is more sensitive on the parallel performance efficiency, due to the synchronization overhead, SIDMACA can obtain same or better quality for less computational time, which is gain on both scales: quality and cost.

In order to see how significant is this improvement and how robust is this approach additional experimental analysis regarding different problem type (large and complex) and experiment setup should be performed.

## References

[1] A. Bahreininejad, B.H.V. Topping, and A.I. Khan. Finite Element Mesh Partitioning Using Neural Networks. *Adv. Eng. Softw.*, 27(1-2):103–115, 1996.

[2] S.T. Barnard and H.D. Simon. A Fast Multilevel Implementation of Recursive Spectral Bisection for Partitioning Unstructured Problems. *Concurr. Comp.-Pract. E.*, 6(2):101–117, 1994.

[3] M. Blondi and M. Bondanza. Parallelizzazione di un Algoritmo per la Risoluzione del Problema del Commesso Viaggiatore. Master's thesis, Politecnico di Milano, 1993.

[4] R.D. Cook, D.S. Malkus, M.E. Plesha, and R.J. Witt. *Concepts and Applications of Finite Element Analysis*. John Wiley & Sons, 2001.

[5] M. Dorigo, G. Di Caro. The Ant Colony Optimization Meta-Heuristic. In D. Corne, M. Dorigo, and F. Glover, editors, *New Ideas in Optimization*, McGraw-Hill, 1999.

[6] M. Dorigo. Optimization, Learning and Natural Algorithms. PhD Thesis, Dipartimento di Elettronica, Politecnico di Milano, 1992.

[7] C.M. Fiduccia and R.M. Mattheyses. A Linear Time Heuristic for Improving Network Partitions. In *Proc. 19th IEEE Design Automation Conf.*, Las Vegas, NV, 1982, pages 175–181.

[8] Graph Collection. `http://wwwcs.uni-paderborn.de/cs/ag-monien/RESEARCH/PART/graphs.html`.

[9] B. Hendrickson and R. Leland. A Multilevel Algorithm for Partitioning Graphs. In *Proc. ACM/IEEE Conf. Supercomputing*, San Diego, CA, 1995.

[10] P. Kadłuczka and K. Wala. Tabu Search and Genetic Algorithms for the Generalized Graph Partitioning Problem. *Control Cybern.*, 24(4)459–476, 1995.

[11] G. Karypis and V. Kumar. Multilevel k-way Partitioning Scheme for Irregular Graphs. *J. Parallel Distr. Com.*, 48(1):96–129, 1998.

[12] B.W. Kernighan and S. Lin. An Efficient Heuristic Procedure for Partitioning Graph. *Bell Sys. Tech. J.*, 49(2)291–307, 1970.

[13] A.E. Langham and P.W. Grant. Using Competing Ant Colonies to Solve k-way Partitioning Problems with Foraging and raiding strategies. *Lect. Notes Comp. Sc.*, 1674:621–625, 1999.

[14] P. Korošec, J. Šilc, and B. Robič. Solving the Mesh-partitioning Problem with an Ant-colony Algorithm. *Parallel Comput.*, 30(5-6):785–801, 2004

[15] M. Randall and A. Lewis. A Parallel Implementation of Ant Colony Optimization. *J. Parallel Distr. Com.*, 62(9):1421–1432, 2002.

[16] T. Stützle. Parallelization Strategies for Ant Colony Optimization. *Lect. Notes Comp. Sc.*, 1498:722–741, 1998.

[17] C. Walshaw and M. Cross. Mesh Partitioning: A Multilevel Balancing and Refinement Algorithm. *SIAM J. Sci. Comput.*, 22(1):63–80, 2001.

# SELF-ADAPTIVE DIFFERENTIAL EVOLUTION WITH SQP LOCAL SEARCH

Janez Brest, Aleš Zamuda, Borko Bošković, Sašo Greiner,
Mirjam Sepesy Maučec, Viljem Žumer
*Faculty of Electrical Engineering and Computer Science, University of Maribor*
*Maribor, Slovenia*
{janez.brest; ales.zamuda; borko.boskovic; saso.greiner; mirjam.sepesy; zumer}@uni-mb.si

**Abstract** In this paper we present experimental results of self-adaptive differential evolution algorithm hybridized with a local search method. The results of the proposed hybrid algorithm are evaluated on a set of benchmark functions provided by the IEEE Congress on Evolutionary Computation (CEC 2008) special session on Large Scale Global Optimization. Performance comparison of our algorithm with other algorithms is reported.

**Keywords:** Differential Evolution, Local search, Optimization, Self-adaptation

## 1. Introduction

In recent years numerous stochastic optimization algorithms have been proposed to solve real-parameter optimization problems, such as evolution strategies, real-parameter genetic algorithms, simulated annealing, differential evolution, particle swarm optimization, ant-colony optimization, evolutionary algorithms, etc. The optimization problem is to find $\vec{x}$, which optimizes the objective function $f(\vec{x})$ where $\vec{x} = [x_1, x_2, ..., x_D]^T$ is a set of real variables. $D$ is the dimensionality of the search space. Domains of the variables are defined by their lower and upper bounds: $x_{j,low}$, $x_{j,upp}$; $j \in \{1, ..., D\}$. A priori knowledge about the objective function is usually very limited and, in practice, the objective function is often nonlinear, multi-modal, etc.

In this paper we hybridize our self-adaptive differential evolution algorithm jDEdynNP [6] with a local search procedure. The performance of the new algorithm is evaluated on a set of benchmark functions provided by CEC 2008 special session on Large Scale Global Optimization

(LSGO) [21] at the IEEE World Congress on Computational Intelligence (WCCI 2008).

## 2. Background

In this section we give an overview of previous work. The references to source code of the original differential evolution (DE) algorithm are presented. Then the self-adaptive mechanism used in our DE algorithm is briefly outlined. In section 3 the SQP local search procedure is described.

To solve high-dimensional problems [21], cooperative coevolution [15, 19] can be used. Liu et al. [12] used FEP (Fast Evolutionary Programming) with cooperative coevolution (FEPCC) to speedup convergence rates on large-scale problems, Bergh and Engelbrecht [23] used a Cooperative Approach to Particle Swarm Optimisation (PSO), Yang, Tang and Yao used differential evolution with cooperative coevolution (DECC) [25], recently. Gao and Wang [9] used a memetic DE algorithm for high-dimensional problem optimization. There were 8 papers accepted to LSGO at CEC 2008: [26, 10, 6, 13, 22, 27, 28, 24] and many of them used DE.

### 2.1 The Differential Evolution Algorithm

DE is a population based evolutionary algorithm proposed by Storn and Price [20, 18]. The original DE has three control parameters: amplification factor of the difference vector – $F$, crossover control parameter – $CR$, and population size – $NP$. During one generation for each vector, DE employs the mutation, crossover and selection operations to produce a new vector for the next generation. DE [20, 16, 8] has been shown to be a simple yet powerful evolutionary algorithm for global optimization in many real problems [14].

In this paper we will skip a detailed description of the DE algorithm. The algorithm is widely used in many research areas and it is implemented in several programming languages (for source code of the algorithm see DE homepage: `http://www.icsi.berkeley.edu/~storn/code.html`).

### 2.2 The Self-adaptive DE Algorithm

In this subsection we revise our jDEdynNP-F algorithm [6], which was proposed at the CEC 2008 special session. The jDEdynNP-F algorithm applies self-adapted $F$ and $CR$ control parameters and a population size reduction method. Additionally, it implements a mechanism for sign

changing of $F$ control parameter with some probability based on the fitness values of randomly chosen vectors, which are multiplied by the $F$ control parameter (scaling factor) in the mutation operation of the DE algorithm.

The jDEdynNP-F algorithm uses the same self-adaptive control mechanism as it was first proposed in [4] and lately used in many other variants [7, 3, 5, 2]. This mechanism changes the control parameters $F$ and $CR$ during the run.

The jDEdynNP-F algorithm implements the method for gradually reducing population size [5] during the optimization process. The dynamic population size reduction mechanism is used in the jDEdynNP-F algorithm to start optimization with the greatest population at the beginning of the evolutionary process, and finishes optimization with the smallest population size. The population size is gradually reduced. In [5, 6] we proposed a reduction scheme where the new population size is equal to half of the previous population size.

The jDEdynNP-F algorithm applies a mechanism that changes the sign of the control parameter $F$ with some probability ($prob = 0.75$) when $f(\vec{x}_{r_2}) > f(\vec{x}_{r_3})$ during the mutation operation as presented in Fig. 1. *rand* generates random numbers uniformly distributed between 0 and 1. This mechanism uses *rand/1/bin/* DE strategy and was proposed in [6].

```
// individuals' objective function values are stored in array named cost
prob = 0.75;                          // probability for changing the sign
if (rand < prob && cost[r2] > cost[r3])
    F = -F;                           // sign change
```

*Figure 1.* The control parameter $F$ changes sign.

In this paper, the jDEdynNP-F algorithm is hybridized for the first time with a local search procedure, which is presented in the following section.

## 3. Sequential Quadratic Programming (SQP)

Sequential Quadratic Programming (SQP) [17, 1] is a non-linear optimization method based on gradient computation. It is a generalization of Newton's method to multiple dimensions, incorporating a quadratic approximation model for the objective function given an initial guess for the solution. Great strength of the SQP method is its ability to solve problems with nonlinear constraints. The approximation model is solved at each iteration to yield a step toward the solution of the original

*Table 1.* LSGO@CEC'08 benchmark functions

| | | | |
|---|---|---|---|
| $F_1$ | Shifted Sphere Function | uni-modal | separable |
| $F_2$ | Shifted Schwefel's Problem 2.21 | uni-modal | non-separable |
| $F_3$ | Shifted Rosenbrock's Function | multi-modal | non-separable |
| $F_4$ | Shifted Rastrigin's Function | multi-modal | separable |
| $F_5$ | Shifted Griewank's Function | multi-modal | non-separable |
| $F_6$ | Shifted Ackley's Function | multi-modal | separable |
| $F_7$ | FastFractal "DoubleDip" Function | multi-modal | separable |

problem. As with most optimization methods, SQP is not a single algorithm, but rather a conceptual method from which numerous specific algorithms have evolved [1].

The algorithm for SQP we have used is FSQP-AL and its implementation is given in CfSQP [11]. The norm of descant direction was set to $\epsilon$ =1.e–8. Constraints were enforced only in terms of bounds to search parameters, i.e. linear bound-constraints were used.

Hybridization of the chosen local search algorithm in our global optimization jDEdynNP-F algorithm was as follows. First, the global algorithm was run for 30% of maximum number of function evaluations (MAXFEs). Then after every 100 generations (note that we have dynamic population size during the evolutionary process), we employed the local search method on the fittest individual of the current population. The number of iterations of the SQP method that were used to refine the given solution was set to $\lfloor \frac{\sqrt{D}}{5} \rfloor$.

After the SQP local search procedure is called, we check whether the new obtained individual (result from SQP procedure) is better than the currently best individual. If SQP finds a better individual (in this case SQP returns a positive value), it will be stored, otherwise when SQP returns a negative value we do not use SQP in the rest of the evolutionary process. The suggested mechanism seems to work fine with fractal function $F_7$, when the SQP local search procedure usually could not make any improvement of the currently best individual.

## 4. Experimental Results

In this section we present results of experiments, which were made in order to present the performance of the proposed hybrid algorithm.

Table 1 shows characteristics of CEC 2008 benchmark functions.

Table 2 presents the obtained results of our hybrid algorithm on the benchmark functions. The error values ($f(\vec{x}) - f(\vec{x}^*)$) are presented

*Table 2.* Error values achieved for problems $F_1$–$F_6$, with $D = 1000$. Function value achieved for function $F_7$ with $D = 1000$

| | jDEdynNP-F [6] | | jDEdynNP-F with SQP | |
|---|---|---|---|---|
| | Mean | Std. dev. | Mean | Std. dev. |
| $F_1$ | 1.1369e–13 | 0.0000 | 1.1141e-13 | 1.1369e-14 |
| $F_2$ | 1.9529e+01 | 2.2525 | 2.6582e+01 | 1.6529 |
| $F_3$ | 1.3136e+03 | 1.3635e+02 | 3.2719e+02 | 1.7768e+02 |
| $F_4$ | 2.1668e–04 | 4.0563e–04 | 8.3060e-12 | 3.6271e-12 |
| $F_5$ | 3.9790e–14 | 1.4211e–14 | 5.0022e-14 | 1.2389e-14 |
| $F_6$ | 1.4687e–11 | 2.4310e–11 | 3.7630e-13 | 3.8851e-13 |
| $F_7$ | –1.3491e+04 | 4.6038e+01 | -1.3766e+04 | 8.2836e+01 |

in the table. The optimal solution results are known for benchmark functions $F_1$–$F_6$, while for function $F_7$ the optimal solution value is not given.

Figure 2 shows the convergence graphs for functions $F_1$–$F_7$ on $D = 100$ with and without SQP local search procedure for the best, median and worst individuals obtained at the end of the evolutionary process. Figures 3 and 4 show convergence graphs for functions when $D = 500$ and $D = 1000$, respectively.

The convergence graphs show that the algorithm with the SQP performs better than the algorithm without SQP in most cases, exception is function $F_2$ (*Schwefel's Problem 2.21*) when $D = 1000$. The algorithm with the SQP obviously gives better results on functions $F_3$ (*Rosenbrock's function*), and $F_4$ (*Rastrigin's function*) when $D = 1000$.

In this experiment we did not make fine tuning of the SQP's parameters, i.e. when starting SQP, how many iterations may be used by SQP, etc.

The summary result of the LSGO 2008 competition are available at `http://nical.ustc.edu.cn/papers/CEC2008_SUMMARY.pdf`, where results comparison on $D = 1000$ functions are presented. The jDEdynNP-F algorithm took third place, after [22] and [24].

The mean value obtained by our proposed algorithm with the SQP is lower than 1.e–10 (roughly speaking, a function *is solved*, when mean value drops under 1.e–10) for functions $F_1$ (6), $F_4$ (2), $F_5$ (6), and $F_6$ (4). In the parentheses after function we give a number of LSGO algorithms that also reached bound 1.e–10 (note, the competition included eight algorithms).

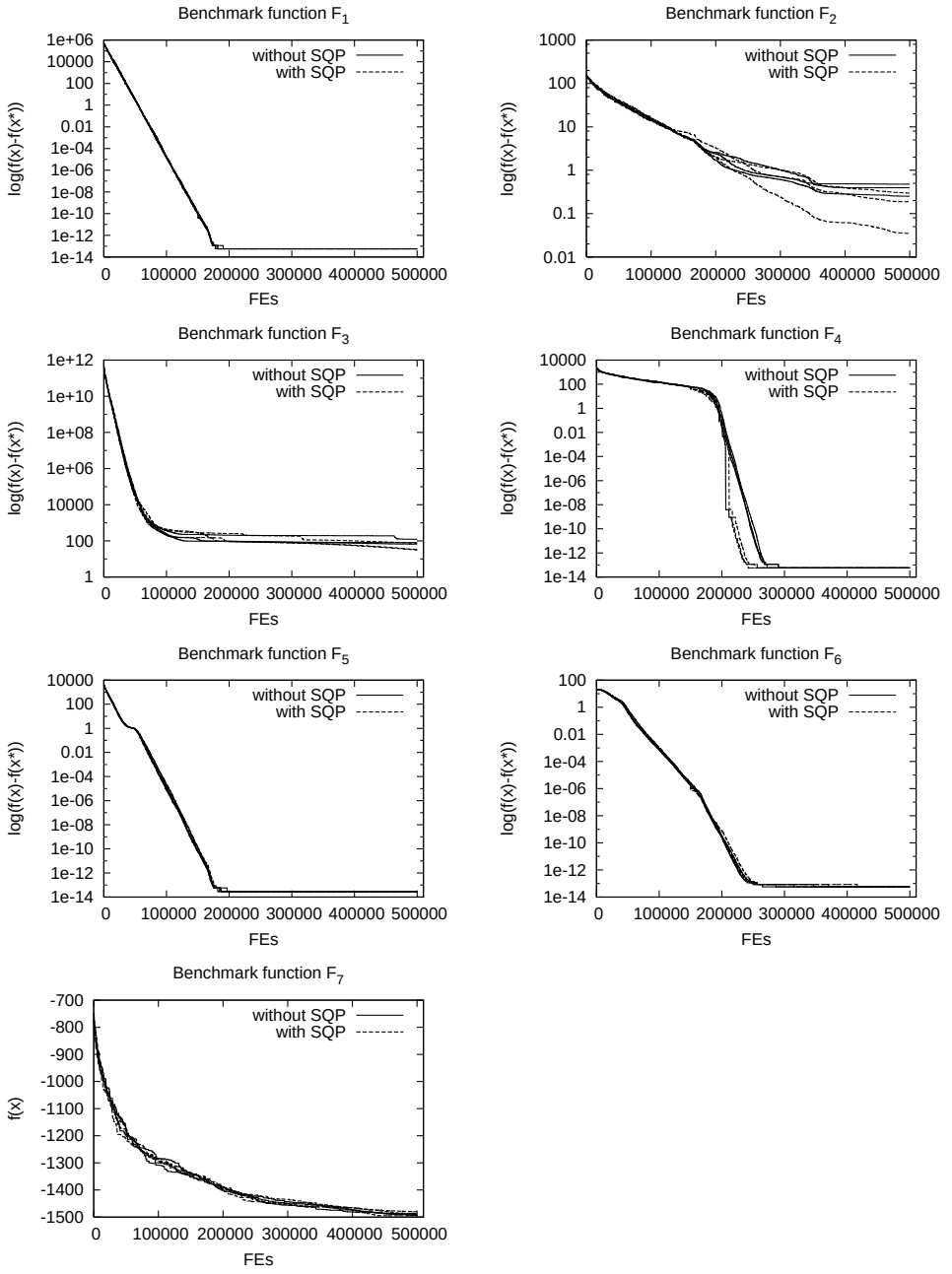



*Figure 2.* Convergence graphs for functions $F_1$–$F_6$ with $D = 100$.

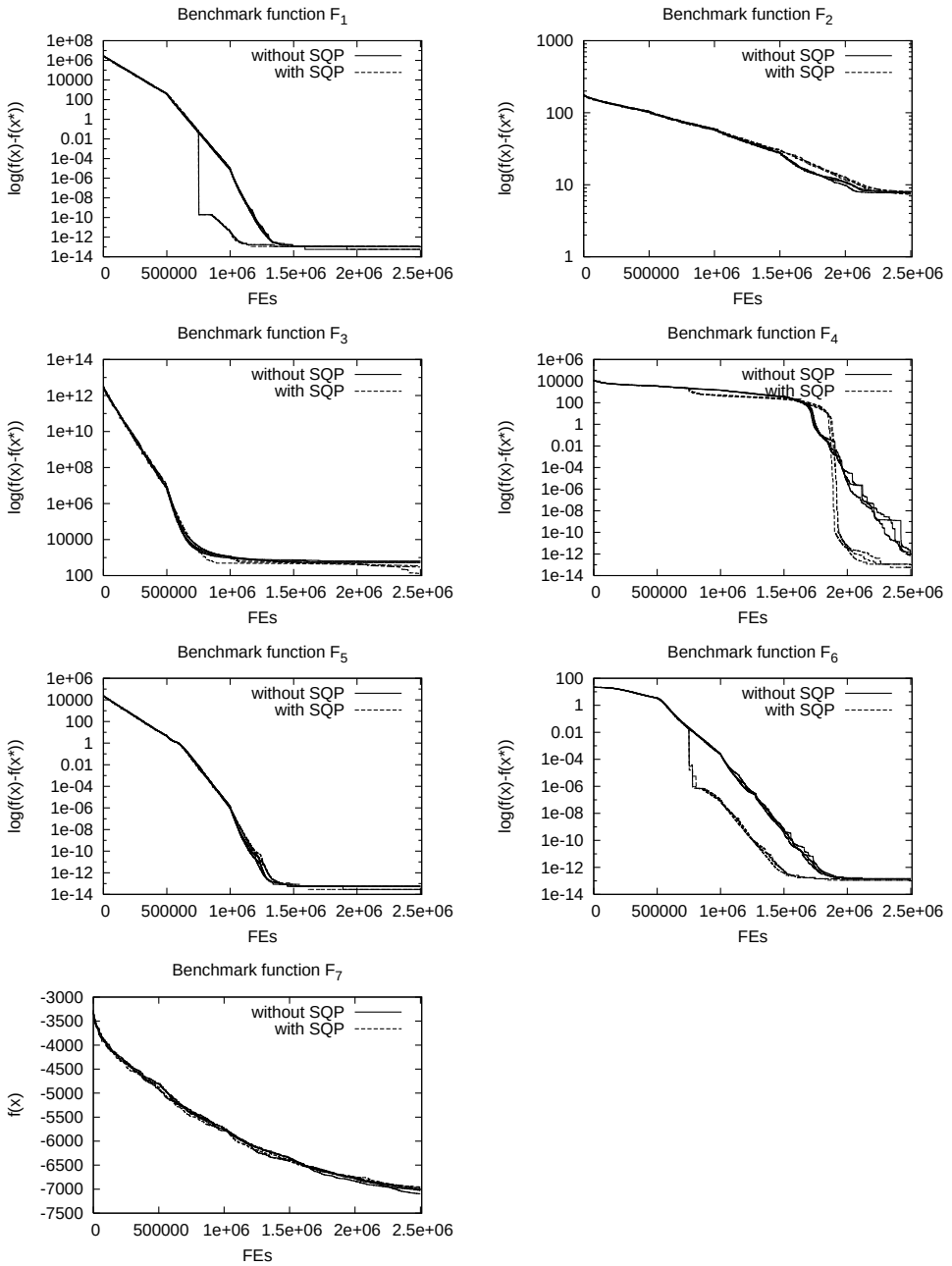



*Figure 3.* Convergence graphs for functions $F_1$–$F_6$ with $D = 500$.

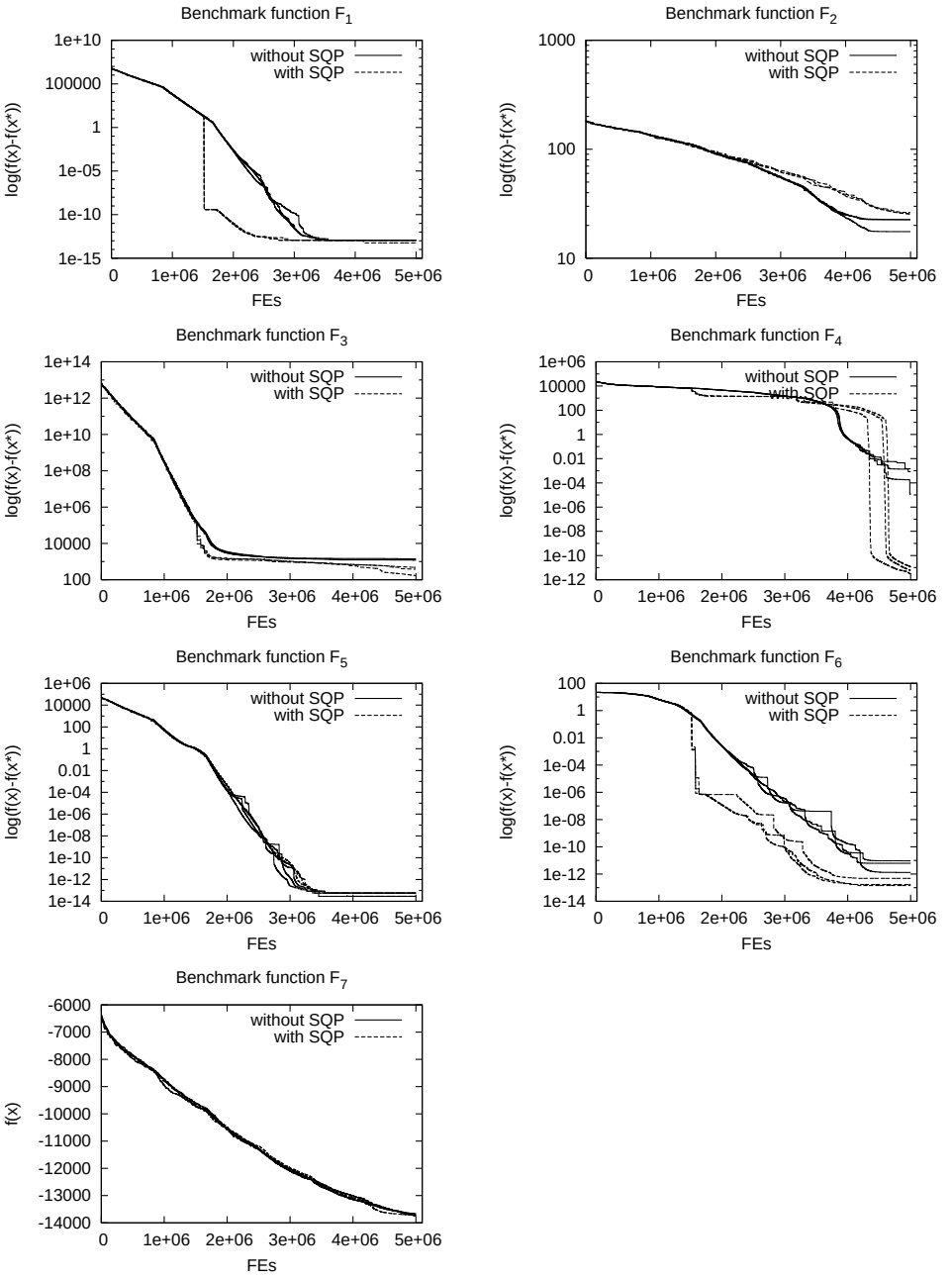



*Figure 4.* Convergence graphs for functions $F_1$–$F_6$ with $D = 1000$.

Our algorithm has rank 4, 2, and 3 for functions $F_2$, $F_3$, $F_7$, respectively. Based on this comparison of the jDEdynNP-F with SQP with the LSGO 2008 results, we can conclude that our algorithm is highly competitive on all LSGO 2008 functions when $D = 1000$.

## 5. Conclusions

This paper presents our attempt to hybridize self-adaptive differential jDEdynNP-F algorithm with SQP local search procedure. The experimental results confirm that the proposed hybrid algorithm might perform better than the algorithm without SQP. The better parameter setting for the SQP and deep insight of it are challenges for future work.

## Acknowledgment

We thank the authors of CfSQP for providing us with the academic license of their software.

## References

[1] P.T. Boggs and J.W. Tolle. Sequential quadratic programming for large-scale nonlinear optimization. *J. Comput. Appl. Math.*, 124(1-2):123–137, 2000.

[2] J. Brest. Differential Evolution with Self-Adaptation. In Juan Ramon Rabunal Dopico, Julian Dorado de la Calle, and Alejandro Pazos Sierra, editors, *Encyclopedia of Artificial Intelligence*, Information Science Reference, Hershey, 2008, pp. 488–493.

[3] J. Brest, B. Bošković, S. Greiner, V. Žumer, and M. Sepesy Maučec. Performance comparison of self-adaptive and adaptive differential evolution algorithms. *Soft Comput.*, 11(7):617–629, 2007.

[4] J. Brest, S. Greiner, B. Bošković, M. Mernik, and V. Žumer. Self-Adapting Control Parameters in Differential Evolution: A Comparative Study on Numerical Benchmark Problems. *IEEE T. Evol. Comput.*, 10(6):646–657, 2006.

[5] J. Brest, and M. Sepesy Maučec. Population Size Reduction for the Differential Evolution Algorithm. *Appl. Intell.*, DOI: 10.1007/s10489-007-0091-x.

[6] J. Brest, A. Zamuda, B. Bošković, M. Sepesy Maučec, and V. Žumer. High-Dimensional Real-Parameter Optimization using Self-Adaptive Differential Evolution Algorithm with Population Size Reduction. In *Proc. IEEE Congress on Evolutionary Computation (CEC 2008)*, pages 2032-2039, Hong Kong, 2008.

[7] J. Brest, V. Žumer, and M. Sepesy Maučec. Control Parameters in Self-Adaptive Differential Evolution. In B. Filipič and J. Šilc, editors, *Bioinspired Optimization Methods and Their Applications*, pages 35–44, Ljubljana, Slovenia, October 2006.

[8] V. Feoktistov. *Differential Evolution: In Search of Solutions (Springer Optimization and Its Applications)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[9] Y. Gao and Y.-J. Wang. A Memetic Differential Evolutionary Algorithm for High Dimensional Functions' Optimization. In *Proc. Third International Conference on Natural Computation (ICNC 2007)*, pages 188–192, 2007.

[10] S.-T. Hsieh, T.-Y. Sun, C.-C. Liu, and S.-J. Tsai. Solving Large Scale Global Optimization Using Improved Particle Swarm Optimizer. In *Proc. IEEE World Congress on Computational Intelligence*, pages 1777–1784, Hong Kong, 2008.

[11] C. T. Lawrence, J.L. Zhou, and A.L. Tits. User's Guide for CFSQP Version 2.5: A C Code for Solving (Large Scale) Constrained Nonlinear (Minimax) Optimization Problems, Generating Iterates Satisfying All Inequality Constraints. Technical report TR-94-16r1, Institute for Systems Research, University of Maryland, College Park, MD, 1997.

[12] Y. Liu, X. Yao, Q. Zhao, and T. Higuchi. Scaling Up Fast Evolutionary Programming with Cooperative Coevolution. In *Proc. IEEE Congress on Evolutionary Computation (CEC 2001)*, pages 1101–1108, 2001.

[13] C. MacNish and X. Yao. Direction Matters in High-Dimensional Optimisation. In *Proc. IEEE World Congress on Computational Intelligence (WCCI 2008)*, pages 2377–2384, Hong Kong, 2008.

[14] Z. Michalewicz and D.B. Fogel. *How to Solve It: Modern Heuristics*. Springer, Berlin, 2000.

[15] M.A. Potter and K. De Jong. A Cooperative Coevolutionary Approach to Function Optimization. In Y. Davidor, H.-P Schwefel, and R. Männer, editors, *Parallel Problem Solving from Nature – PPSN III*, pages 249–257, 1994.

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

[17] G.V. Reklaitis, A. Ravindran, and K.M. Ragsdell. *Engineering Optimization: Methods and Applications*. Wiley-Interscience, 1983.

[18] J. Rönkkönen, S. Kukkonen, and K.V. Price. Real-Parameter Optimization with Differential Evolution. In *Proc. IEEE Congress on Evolutionary Computation (CEC 2005)*, pages 506–513, Edinburgh, UK, 2005.

[19] D. Sofge, K. De Jong, and A. Schultz. A Blended Population Approach to Cooperative Coevolution for Decomposition of Complex Problems. In *Proc. IEEE Congress on Evolutionary Computation (CEC 2002)*, pages 413–418, 2002.

[20] 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.

[21] K. Tang, X. Yao, P. N. Suganthan, C. MacNish, Y. P. Chen, C. M. Chen, and Z. Yang. Benchmark Functions for the CEC'2008 Special Session and Competition on High-Dimenasional Real-Parameter Optimization. Technical report, Nature Inspired Computation and Applications Laboratory, USTC, China, 2007. `http://nical.ustc.edu.cn/cec08ss.php`.

[22] L.-Y. Tseng and C. Chen. Multiple Trajectory Search for Large Scale Global Optimization. In *Proc. IEEE World Congress on Computational Intelligence (WCCI 2008)*, pages 3057–3064, Hong Kong, 2008.

[23] F. van den Bergh and A.P. Engelbrecht. A Cooperative Approach to Particle Swarm Optimisation. *IEEE T. Evol. Comput.*, 8(3):225–239, 2004.

[24] Y. Wang and B. Li. A Restart Univariate Estimation of Distribution Algorithm: Sampling under Mixed Gaussian and Lévy probability Distribution. In *Proc. IEEE World Congress on Computational Intelligence (WCCI 2008)*, pages 3918–3925, Hong Kong, 2008.

[25] Z. Yang, K. Tang, and X. Yao. Differential Evolution for High-Dimensional Function Optimization. In *Proc. IEEE Congress on Evolutionary Computation (CEC 2007)*, pages 3523–3530, Singapore, 2007.

[26] Z. Yang, K. Tang, and X. Yao. Multilevel Cooperative Coevolution for Large Scale Optimization. In *Proc. IEEE World Congress on Computational Intelligence (WCCI 2008)*, pages 1663-1670, Hong Kong, 2008.

[27] A. Zamuda, J. Brest, B. Bošković, and V. Žumer. Large Scale Global Optimization Using Differential Evolution With Self-adaptation and Cooperative Co-evolution. In *Proc. IEEE World Congress on Computational Intelligence (WCCI 2008)*, pages 3719–3726, Hong Kong, 2008.

[28] S.Z. Zhao, J.J. Liang, P.N. Suganthan, and M.F. Tasgetiren. Dynamic Multi-Swarm Particle Swarm Optimizer with Local Search for Large Scale Global Optimization. In *Proc. IEEE World Congress on Computational Intelligence (WCCI 2008)*, pages 3846–3853, Hong Kong, 2008.

# STATISTICAL MULTI-COMPARISON OF EVOLUTIONARY ALGORITHMS

Mathieu Barrette, Tony Wong, Bruno de Kelper, Pascal Côté
*École de technologie supérieure, University of Québec*
*Québec, Canada*
{mathieu.barette; tony.wong; bruno.dekelper}@etsmtl.ca

**Abstract** Performance benchmarking is an essential task in computational intelligence research. Evolutionary optimization algorithms are stochastic processes and to obtain significant results proper statistical tools must be used. This paper presents a step-by-step nonparametric comparison procedure able to assess the relative performance of several single-objective evolutionary algorithms. This comparison procedure is based on a simple ranking scheme and is statistically relevant under a controlled risk of error. A useful feature of the procedure is the ability to visualize the dynamical behavior of the algorithms along with the confidence intervals of their performance. It also has the advantage of eliminating the use of arbitrary weighting coefficients when several comparisons criteria are involved.

**Keywords:** Benchmarking, Evolutionary algorithms, Multiple comparisons

## 1. Introduction

Comparing the performance of different evolutionary optimization algorithms (EAs) is an essential task especially when developing new approaches using the computational intelligence paradigm. Usually a benchmarking process is performed in order to demonstrate the effectiveness of the newly devised algorithm against some other well-known EAs. There exists several performance criteria commonly used in the literature. For example, the mean speed of convergence excluding failed trial runs [1], the number of trial runs reaching the global optimum [12], the mean solution fitness [13] and the best solution fitness [5, 6, 11]. All these performance criteria are contextually valid but they should be used within some proper comparison procedure that is statistically significant. Generally, evolutionary algorithm (EA) benchmarking involves several trial runs. Because of the limited sample size, sample statistics will often

not be able to reveal their underlying distribution. Worst, misinterpretation of the benchmarking results may lead to erroneous conclusions. Thus, to perform comparison analysis, it is also necessary to determine if the observed performance differences are truly significant.

This paper is organized as follows: Section 2 describes the basic evolutionary optimization approach. Section 3 presents the multiple comparisons procedure followed by a discussion of the complete experimental protocol. The last section shows the comparison results for five well-known EAs using synthetic optimization problems.

## 2. Evolutionary Optimization

An evolutionary algorithm is an iterative stochastic process often used to optimize difficult numerical problems such as $F(\mathbf{x}) : \mathbb{R}^N \rightarrow \mathbb{R}$ where $\mathbb{R}$ is the set of real numbers, $N$ is the problem dimension and $\mathbf{x} = \{x_1, x_2, \ldots, x_N\}$ is the set of decision variables bounded by $L_i \leq x_i \leq H_i, i = 1, 2, \ldots, N$. Discussions in this paper are oriented towards minimization problems but they are equally applicable to maximization problem since $\min\{F(\mathbf{x})\} = -\max\{-F(\mathbf{x})\}$. In this paper $\mathbf{x}_n$ denotes the best solution found by an EA instance after $n$ iterations and $f(n) \equiv F(\mathbf{x}_n)$ the corresponding objective function value or more succinctly the cost value. We shall consider that an iteration is synonymous to one evaluation of $F(\cdot)$, thus $f(n)$ is the cost value obtained after $n$ evaluation of the objective function. In most context, the number of iterations $n^*$ needed to reach the global optimum is not bounded and should be considered as a random variable with unknown distribution. Since $n^*$ is not known beforehand, it is often desirable to install some stopping criteria which limit the execution time of an EA.

### 2.1 Stopping Criterion

It is possible to stop an evolutionary process once it reaches a zone enclosing the global optimum. We define this zone $Z$ by adding a small threshold value $\varepsilon$ (usually in the order of $10^{-6}$) to the optimal solution $f^*$ of a given optimization problem. In other words, the evolutionary process should be stopped as soon as $f(n) \leq Z$ where $Z \equiv f^* + \varepsilon$.

However, evolutionary optimizers are not always immune to local attractors. They may get imprisoned in a local optimum and not be able to reach $Z$. A maximum allowable number of iterations $n_{\max}$ is recommended to confine the optimization process within a reasonable processing time. So, there are usually two stopping criteria in play while evaluating EA performances: i) reaching the optimum area $Z$; ii) exhausting the maximum number of iterations $n_{\max}$.

## 3. Performance Comparison

Since EAs are stochastic processes, it is necessary to execute several trial runs in order to obtain a fair comparison. In practice, not all EAs can reach the optimum zone for all trials. This fact is shown in Fig. 1 where five different evolutionary optimizers are used to optimize a numerical function. In this experiment, only $EA_1$ and $EA_5$ are able to reach $Z$ for all trials. For $EA_2$ and $EA_4$ they failed to reach $Z$ for some trials but succeeded for some others. Finally, $EA_3$ always misses the optimum area.

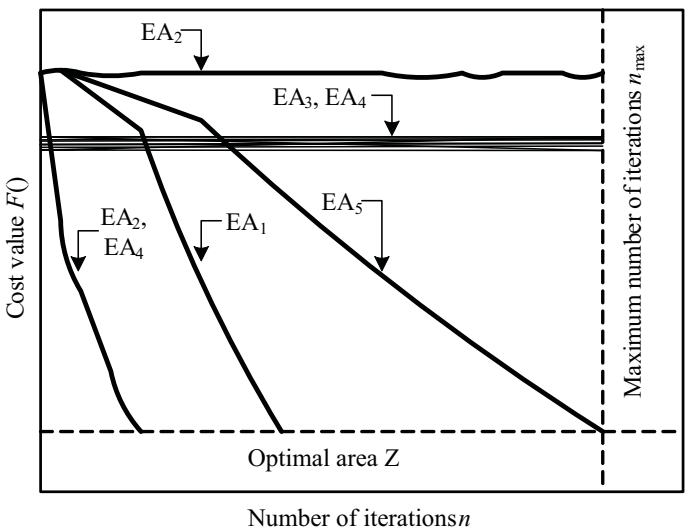



*Figure 1.* Typical performance results for five different EAs.

Thus, the finite confinement provided by $Z$ and $n_{\max}$ give rise to two different types of optimization results for each trial run: i) an EA instance reaches the optimal zone; ii) an EA instance failed to reach the optimal zone. Therefore, a fair performance comparison procedure must also take into account the number of successful (failed) trial runs and the number of iterations needed to reach the optimal zone.

### 3.1 Comparison Ranking Scheme

As mentioned in the introduction, EA performance evaluation can be performed using different criteria. In this paper, we propose the integration of several comparison criteria by ranking the trial run results according to their relative performance. Consider $M$ trial runs, for a given EA, within which $I \leq M$ trials reached the optimal zone $Z$ while

the remaining $M - I$ trials failed to reach $Z$. We assign ranks to the trial runs according to the following scheme: assign ranking numbers, in increasing order, to all $I$ successful trial runs with the best trials receiving the smallest rank number. For example, the rank number may be based on the number of iterations $n$ needed to reach $Z$. For the $M-I$ failed trial runs, $n = n_{\max}$ and the ranking should be based on another criterion. To illustrate the proposed ranking scheme, we will use the cost value $f(n)$ to rank the failed trials. This ranking scheme can be carried out graphically by using the results shown in Fig. 1 as follows: starting from the leftmost of the optimal zone which is parallel to the $x$-axis, move along this axis and assign a number to the trials intersecting the optimal zone. When we reach the $n_{\max}$ value continue ranking the trials with the cost value $f(n_{\max})$ given by the $y$-axis. This ranking scheme is given by Eqs. (1) and (2) where $q$ denotes the "quality" of a trial and $R$ its rank:

$$q_j = n_j + f_j, \quad j = 1, 2, \ldots, M; \tag{1}$$

$$R_j = 0.5 + \sum_{k=1}^{M} s(q_j - q_k), \quad s(u) = \begin{cases} 0 & \text{if } u < 0, \\ 0.5 & \text{if } u = 0, \\ 1 & \text{if } u > 0. \end{cases} \tag{2}$$

Extending the ranking scheme to classify multiple EA performance is straightforward. Consider the case of $k$ different EAs using $m_i$ trials each for a total of $M = k \times m_i$ trials. The resulting ranks can be arranged into an $k \times m_k$ matrix as shown in Eq. (3).

$$\begin{matrix} R_{1,1} & R_{2,1} & \cdots & R_{k,1} \\ R_{1,2} & R_{2,2} & \cdots & R_{k,2} \\ \vdots & \vdots & \vdots & \vdots \\ R_{1,m_1} & R_{2,m_2} & \cdots & R_{k,m_k} \end{matrix} \tag{3}$$

### 3.2 Hypothesis Testing

Performance comparison can be initiated after the ranking assignment. Statistical hypothesis testing (SHT) is a fair approach to determine the significance of the observed differences. In SHT the probability of obtaining at least as extreme as that observed, based on the truthfulness of the null hypothesis, is called the $p$-value. If the $p$-value is smaller than a threshold (for example $\alpha$) then the difference is statistically significant. In this paper, the null hypothesis $H_0$ and the alternate hypothesis $H_1$ can be defined as

$$\begin{matrix} H_0 & : & \text{The } k \text{ results are equivalent} \\ H_1 & : & \text{At least two results differ} \end{matrix} \tag{4}$$

Here the data are ranks, and one nonparametric test between $k$ different and independent samples based on their ranks is the well-known Kruskall-Wallis test defined by the following statistic [4, 7]:

$$H = \frac{12}{M(M+1)} \sum_{i=1}^{k} \frac{\overline{R}_i^2}{m_i} - 3(M+1), \tag{5}$$

where $\overline{R}_i$ is the mean rank of the $i$-th EA and $M$ is the total number of trial runs. This statistic approximates the $\chi^2$ distribution with $k-1$ degrees of freedom if the null hypothesis of equal populations is true. This approximation is accurate except when $k = 3$ and $m_i \leq 5, i = 1, 2, \ldots, k$. Also, in the case of tied ranks, as it changes the variance, an adjustment should be made by dividing $H$ with

$$1 - \frac{\sum_{i=1}^{g}(T_i^3 - t)}{M^3 - M} \tag{6}$$

where $g$ is the number of groupings of different tied ranks and $t_i$ is the number of tied values within group $i$.

If the SHT rejects the null hypothesis then at least one of the k EAs performed differently from at least one other EA. However, SHT will not indicate which one of the $k$ EAs is involved. To get this information multiple comparisons should be used.

### 3.3 Multiple Comparison Procedure

Standard SHT works well in evaluating a single null hypothesis but it cannot be iterated to account for more hypotheses. It is because multiple SHT would increase the threshold value $\alpha$ and thus lower the significance level of the tests. For more than one test, it is also necessary to control the so-called familywise error rate which is the probability of making one or more type I error (accepting the null hypothesis when it is in fact false) over all pairwise tests. In this paper, multiple comparison procedure (MCP) is used as a post-hoc comparison procedure. It is only executed when the null hypothesis is rejected by the Kruskall-Wallis test. For a ranking analysis, Tukey's honestly significant difference criterion (HSD) MCP defined by Eqs. (7) and (8) is recommended [7, 8].

$$|\overline{R}_i - \overline{R}_j| \leq \frac{Q_{k,\infty}^{(\alpha)}}{\sqrt{2}} \sqrt{d_{ij}} \quad i, j < k, \tag{7}$$

$$d_{ij} = \left( \frac{M(M+1)}{12} \frac{\sum(t^3 - t)}{12(M-1)} \right) \left( \frac{1}{m_i} + \frac{1}{m_i} \right), \tag{8}$$

where $Q_{k,v}^{(\alpha)}$ is the upper $\alpha$ point of the studentized range distribution with parameter $k$ and $v$ degrees of freedom. Only if Eq. (7) is true then EA $i$ and $j$ can be assumed to have significant differences.

## 4. Experimental Protocol and Result Analysis

According to the above discussions, an experimental protocol incorporating result ranking, statistical hypothesis testing and multiple comparisons can be described in eight simple steps.

1 Define a testbench comprising well-known optimization problems.

2 Define $n_{\max}$ the maximum number of evaluations.

3 Define $\varepsilon$ the threshold value of the optimum zone $Z$.

4 Select the set of EAs for comparison.

5 Run a predefined number of trials for each EA.

6 Use Eqs. (1) and (2) to compute trial rankings.

7 Execute Kruskall-Wallis test with $\alpha = 0.05$ (for example).

8 If the null hypothesis is rejected, do the HSD MCP.

Using the above protocol, five well-known EAs, in their canonical form, are chosen for comparison. They are: differential evolution (DE) [1], evolution strategy (ES) [2], particle swarm optimizer (PSO) [3, 10, 13], harmony search (HS) [5, 6, 11] and the stochastic hill climbing with learning by vector of normal distribution (SHCLVND or HCL) [12]. The meaning, role and settings of the EA controlling parameters are explained in the cited references. The testbench also includes seven synthetic optimization problems. Six of them have known optimum and they are detailed in Table 1.

For each optimization problem, 50 trials are to be conducted using all five EAs while the optimum zone $Z$ is defined by $\varepsilon = 10^{-6}$. Six optimization problems have been shifted since they have zero-centered optimum. Finally, two versions of the synthetic problems (two-variable and 30-variable) are used in order to verify the effects of problem scaling. In all, 14 problems are to be solved using this experimental protocol.

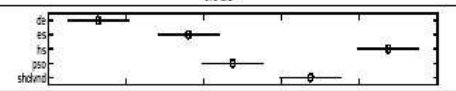
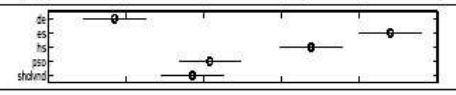
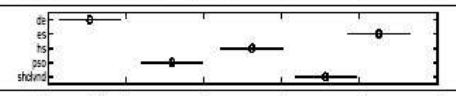
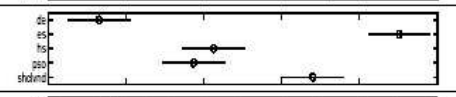
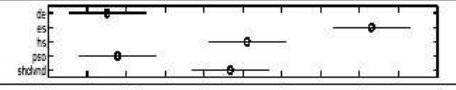
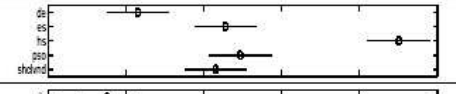
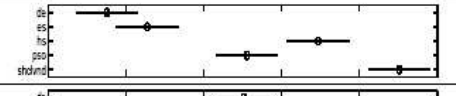
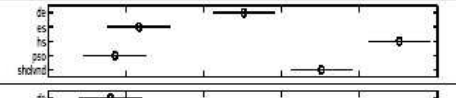
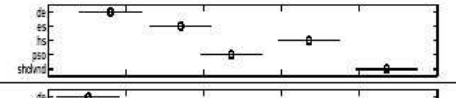
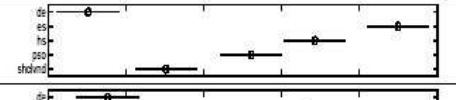
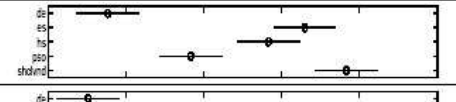
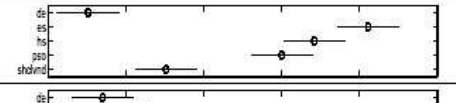
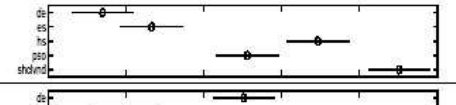
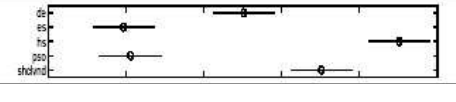
Table 2 details the benchmarking results. In this table column D gives the problem dimension, column $n_{\max}$ denotes the maximum allowable number of iterations, column F is the mean cost value found by the EAs, column CR (convergence rate) is the percentage of trials that reached the optimum zone in the sample, column CS (convergence speed) is the

*Table 1.* Testbench functions

| Function | | $\lvert H_i - L_i \rvert$ | Optimum |
|---|---|---|---|
| Ackley | $F_1(\vec{x}) = -20 \cdot \exp(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}) - \exp(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi \cdot x_i)) + 20 + \exp(1)$ | 60 | $F_1(\vec{0}) = 0$ |
| Corana parabola | $F_2(\vec{x}) = \sum_{i=1}^{n} \begin{cases} c_r\,(sgn\,(z_i)\,t_i + z_i)^2\,d_i & \text{if } (\lvert x_i - z_j\rvert < \lvert t_i\rvert) \wedge (\lvert x_i\rvert > \lvert t_i\rvert) \\ d_i \cdot x_i^2 & \text{otherwise} \end{cases}$ $c_r = 0.15 \quad s_i = 0.2 \quad t_i = 0.05 \quad z_i = \left\lfloor \frac{x_i}{s_i} + 0.49999 \right\rfloor \cdot s_i \cdot sgn(x_i)$ $n = 2 : d = [1, 1000] \quad n = 30 : d = [1, 1000, 10, 10, 1, 10, 100, 1000, 1, 10, ...]$ | 2000 | $F_2(\vec{0}) = 0$ |
| Griewangk | $F_3(\vec{x}) = \sum_{i=1}^{n}(\frac{x_i^2}{4000}) - \prod_{i=1}^{n}(\cos(\frac{x_i}{\sqrt{i}})) + 1$ | 1200 | $F_3(\vec{0}) = 0$ |
| Hyper-Ellipsoid | $F_4(\vec{x}) = \sum_{i=1}^{n} i \cdot x_i^2$ | 10.24 | $F_4(\vec{0}) = 0$ |
| Michalewicz | $F_5(\vec{x}) = \sum_{i=1}^{n} \sin(x_i) \cdot (\sin(\frac{i \cdot x_i^2}{\pi}))^{20}$ | $\pi$ | - |
| Rastrigin | $F_6(\vec{x}) = 10 \cdot n + \sum_{i=1}^{n}(x_i^2 - 10 \cdot \cos(2\pi \cdot x_i))$ | 1200 | $F_6(\vec{0}) = 0$ |
| De Jong's #1 sphere | $F_7(\vec{x}) = \sum_{i=1}^{n} x_i^2$ | 10.24 | $F_7(\vec{0}) = 0$ |

mean number of iterations needed by the trials reaching the optimum zone – failed trials are excluded from this statistic. The last column is the most important one. It is the graphical results of the multiple comparisons procedure. The result analysis of a MCP can be quite difficult to investigate numerically. That is the reason why they are generally supported by a graphical representation that exposes simultaneously all relations between the $k$ EAs. The horizontal lines are the confidence intervals of the mean ranks. When two EAs exhibit overlapping confidence intervals, this means that, under the $\alpha$ error probability, there is no significant difference between the EAs. Since the ranking is given in increasing order (good results have small ranking numbers – see Section 3.1), the best performing EAs are the leftmost ones in the last column of Table 2. We observed that the solution quality and the trial ranks are dependent on $n_{\max}$ and it is also true for the convergence rate, the convergence speed and the fitness value. Since $n_{\max}$ is decided beforehand by trial-and-error it is often desirable to assess the dynamical behavior of the algorithms during their execution. By computing the rank at different temporal states and executing a MCP each time a rank change occurs, it is now possible to visualize the algorithm's performance for different numbers of iterations. In Fig. 2, the $y$-axis is the mean rank of each EA and each horizontal band represents the mean rank of an EA for different temporal range. The band's height is the confidence interval obtained with the MCP. As shown in Fig. 2, rank changes may occur at any point during the optimization process and if $n_{\max}$ is small the outcome could be quite different.

*Table 2.* Benchmarking results

| F | $D$ | $N_{max}$ | EA | F | CR | CS | MCP |
|---|---|---|---|---|---|---|---|
| F1 | 2 | 9.96E5 | DE | 1E-6 | 100 | 1630 |  |
| | | | ES | 2.8 | 86 | 1763 | |
| | | | HS | 2.6E-5 | 0 | - | |
| | | | PSO | 1E-6 | 100 | 5016 | |
| | | | HCL | 1E-6 | 100 | 943 770 | |
| F1 | 30 | 1.311E6 | DE | 1E-6 | 100 | 531 750 |  |
| | | | ES | 18.8 | 2 | 27 649 | |
| | | | HS | 1.9 | 0 | - | |
| | | | PSO | 1.1 | 32 | 30 593 | |
| | | | HCL | 1E-6 | 98 | 1 305 200 | |
| F2 | 2 | 8.5E5 | DE | 1E-6 | 100 | 1456 |  |
| | | | ES | 59 841 | 8 | 1644 | |
| | | | HS | 1E-6 | 100 | 179 050 | |
| | | | PSO | 1E-6 | 100 | 4182 | |
| | | | HCL | 1E-6 | 100 | 747 830 | |
| F2 | 30 | 1.5E6 | DE | 1E-6 | 100 | 640 710 |  |
| | | | ES | 1.69E8 | 0 | - | |
| | | | HS | 4372 | 0 | - | |
| | | | PSO | 112 726 | 16 | 373 730 | |
| | | | HCL | 198 088 | 0 | - | |
| F3 | 2 | 6.33E5 | DE | 2E-3 | 72 | 3557 |  |
| | | | ES | 3.2E-2 | 6 | 1160 | |
| | | | HS | 3.7E-3 | 50 | 62 256 | |
| | | | PSO | 8.9E-4 | 88 | 17 330 | |
| | | | HCL | 1E-6 | 100 | 553 590 | |
| F3 | 30 | 9.41E5 | DE | 1E-6 | 100 | 409 130 |  |
| | | | ES | 1E-2 | 32 | 20 520 | |
| | | | HS | 10E-2 | 0 | - | |
| | | | PSO | 1.7E-2 | 32 | 20 932 | |
| | | | HCL | 1E-6 | 100 | 934 580 | |
| F4 | 2 | 3.128E5 | DE | 1E-6 | 100 | 676 |  |
| | | | ES | 1E-6 | 100 | 756 | |
| | | | HS | 1E-6 | 100 | 10 328 | |
| | | | PSO | 1E-6 | 100 | 1602 | |
| | | | HCL | 1E-6 | 100 | 251 280 | |
| F4 | 30 | 8.1E5 | DE | 1E-6 | 100 | 315 690 |  |
| | | | ES | 1E-6 | 100 | 18 251 | |
| | | | HS | 20.4 | 0 | - | |
| | | | PSO | 1E-6 | 100 | 17 888 | |
| | | | HCL | 1E-6 | 100 | 795 660 | |
| F5 | 2 | 3.5E5 | DE | 1E-6 | 100 | 707 |  |
| | | | ES | 0.11 | 84 | 760 | |
| | | | HS | 1E-6 | 100 | 45 703 | |
| | | | PSO | 1E-6 | 100 | 1715 | |
| | | | HCL | 1E-6 | 100 | 268 050 | |
| F5 | 30 | 3E6 | DE | 1E-6 | 100 | 1 249 900 |  |
| | | | ES | 11.4 | 0 | - | |
| | | | HS | 6.9 | 0 | - | |
| | | | PSO | 5.1 | 0 | - | |
| | | | HCL | 0.41 | 0 | - | |
| F6 | 2 | 8.48E5 | DE | 1E-6 | 100 | 1636 |  |
| | | | ES | 0.975 | 32 | 1520 | |
| | | | HS | 1E-6 | 100 | 119 790 | |
| | | | PSO | 1E-6 | 100 | 4731 | |
| | | | HCL | 1E-6 | 100 | 776 820 | |
| F6 | 30 | 1.3E6 | DE | 1E-6 | 100 | 821 740 |  |
| | | | ES | 155.7 | 0 | - | |
| | | | HS | 118.4 | 0 | - | |
| | | | PSO | 107.2 | 0 | - | |
| | | | HCL | 1.4 | 24 | 1 235 200 | |
| F7 | 2 | 2.972E5 | DE | 1E-6 | 100 | 649 |  |
| | | | ES | 1E-6 | 100 | 732 | |
| | | | HS | 1E-6 | 100 | 6907 | |
| | | | PSO | 1E-6 | 100 | 1637 | |
| | | | HCL | 1E-6 | 100 | 234 310 | |
| F7 | 30 | 7.4E5 | DE | 1E-6 | 100 | 276 690 |  |
| | | | ES | 1E-6 | 100 | 15 442 | |
| | | | HS | 1.73 | 0 | - | |
| | | | PSO | 1E-6 | 100 | 15 692 | |
| | | | HCL | 1E-6 | 100 | 701 230 | |

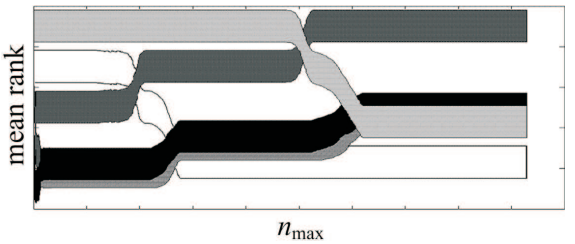



*Figure 2.* MCP progress plot of EA performance.

Interestingly, the benchmarking performed in this work revealed that the differential evolution algorithm is efficient for all seven test problems. DE's performance for the Ackley, Corona's parabola, Griewangk, Michalewicz and Rastrigin functions is significantly different compared to all other EAs. However, this benchmarking also exposed influences by the problem dimension on the EA performances. For instance, DE and ES performed equally well for the 2D hyper-ellipsoid problem but DE is outperformed by PSO and ES when the problem is scaled to 30 dimensions. The same phenomenon is observed for the De Jong sphere problem. In this benchmarking, HS and SHCLVND are outperformed by DE on all test functions.

## 5. Conclusions

This comparison procedure has the advantage to be statistically consistent and adequate to evaluate performance analysis of the EAs. With the ranking scheme, it also allows a unification of the three major criteria under a unique ranking number without the use of arbitrary artefacts such as weighting coefficients or function objective transformations. For these reasons, a statistical multiple comparisons procedure is more rigorous and more robust than the usual single-value objective numerical comparisons.

## References

[1] R. Storn and K. Price. Differential Evolution A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces. *J. Global Optim.*, 11(4):341–359, 1997.

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

[3] M. Clerc and J. Kennedy. The Particle Swarm Explosion, Stability, and Convergence in a Multidimensional Complex Space. *IEEE T. Evol. Comput.*, 6(1):58–73, 2002.

[4] W.J. Conover. *Practical Nonparametric Statistics, 3rd ed.*, John Wiley & Sons, Inc., 1998.

[5] Z.W. Geem and C.-L. Tseng. New Methodology, Harmony Search and Its Robustness. Late-Breaking Papers of GECCO-2002, pages 174–178, New York City, USA, July 2002.

[6] Z.W. Geem and C.-L. Tseng. Engineering Applications of Harmony Search. Late-Breaking Papers of GECCO-2002, pages 169–173, New York City, USA, July 2002.

[7] J.D. Gibbons. *Nonparametric Methods for Quantitative Analysis, 2nd ed.*, American Sciences Press Inc., Ohio, 1985.

[8] J.D. Gibbons and S. Chakraborti. *Nonparametric statistical inference, 3rd ed.*, Marcel Dekker, New York, 1992.

[9] Y. Hochberg and A.C. Tamhane. *Multiple Comparison Procedures*. Wiley, USA, 1987.

[10] J. Kennedy and R. Eberhart. Particle Swarm Optimization. In *Proc. IEEE International Conference on Neural Networks*, pages 1942–1948, 1995.

[11] K.S. Lee and Z.W. Geem. A new structural optimization method based on the harmony search algorithm. *J. Comp. Struc.*, 82:781–798, 2004.

[12] S. Rudlof and M. Koeppen. Stochastic Hill Climbing with Learning by Vectors of Normal Distributions. Technical Report, Fraunhofer-Institut for Production Systems and Design Technology , 1996.

[13] Y. Shi and R.C. Eberhart. Empirical Study of Particle Swarm Optimization. In *Proc. IEEE Congress on Evolutionary Computation (CEC 1999)*, pages 1945–1950, Washington, D.C., 1999.

# TESTING APPROACHES FOR GLOBAL OPTIMIZATION OF SPACE TRAJECTORIES

Massimiliano Vasile, Edmondo Minisci
*Department of Aerospace Engineering, University of Glasgow*
*Glasgow, United Kingdom*
{mvasile; eminisci}@eng.gla.ac.uk

Marco Locatelli
*Dipartimento di Informatica, Università di Torino*
*Torino, Italy*
locatell@di.unito.it

**Abstract** In this paper the procedures to test global search algorithms applied to space trajectory design problems are discussed. Furthermore, a number of performance indexes that can be used to evaluate the effectiveness of the tested algorithms are presented. The performance indexes are then compared and the actual significance of each one of them is highlighted. Three global optimization algorithms are tested on three typical space trajectory design problems.

**Keywords:** Global optimization, Space trajectory design, Stochastic optimization

## 1. Introduction

In the last decade many authors have used global optimization techniques to find optimal solutions to space trajectory design problems. Many different methods have been proposed and tested on a variety of cases. From pure Genetic Algorithms (GAs) [7, 5, 8, 1] to the newer Differential Evolution (DE)[12], and hybrid methods [14], the general intent is to improve over the pure grid or enumerative search. Sometimes, the actual advantage of using a global method is difficult to appreciate, in particular when stochastic based techniques are used. In fact, if, on one hand, a stochastic search provides a non-zero probability to find an optimal solution even with a small number of function evaluations, on the

other hand, the repeatability of the result and therefore the reliability of the method can be questionable. The first actual assessment of the suitability of global optimization methods to the solution of space trajectory design problems can be found in two studies by the University of Reading [6] and by the University of Glasgow [11]. One of the interesting outcomes of both studies was that DE performed particularly well on most of the problems, compared to other methods. In both studies, the indexes of performance for stochastic methods were: the average value of the best solution found for each run over a number of independent runs, the corresponding variance and the best value from all the runs. For deterministic methods, the index of performance was the best value for a given number of function evaluations. In this paper, we propose a testing methodology for global optimization methods addressing specifically black-box problems in space trajectory design. In particular, we focus our attention on stochastic based approaches. The paper discusses the actual significance of a number of performance indexes and proposes an approach to test a global optimization algorithm.

## 2. Testing Procedure

In this section we describe a testing procedure that can be used to investigate the complexity of the problem and to derive the performance indexes described in the next section. If we call $A$ a generic solution algorithm and $p$ a generic problem with objective function $f$, we can define a general procedure as in Algorithm 1.

**Algorithm 1** Convergence Test

1: set the max number of function evaluations for $A$ equal to $N$
2: apply $A$ to $p$ for $n$ times
3: **for all** $i \in [1, ..., n]$ **do** $\phi(N, i) = \min f(A(N), p, i)$
4: **end for**
5: compute: $\phi_{min}(N) = \min_{i \in [1,...,n]} \phi(N, i)$, $\phi_{max}(N) = \max_{i \in [1,...,n]} \phi(N, i)$

Now if the algorithm $A$ is convergent, when the number of function evaluations $N$ goes to infinity the two functions $\phi_{min}$ and $\phi_{max}$ converge to the same value, the global minimum value denoted as $f_{global}$. If we fix a tolerance value $tol_f$, we could consider the following random variable as a possible quality measure of an algorithm

$$N^* = \min\{\phi_{max}(N) - f_{global} \leq tol_f \ : \ \forall \ N \geq N^*\}.$$

The larger (the expected value of) $N^*$ is, the slower is the convergence of $A$. However, such measure can be unpractical since, though finite,

$N^*$ could be very large. In practice, what we would like is not to choose $N$ large enough so that a success is always guaranteed, but rather, for a fixed $N$ value, we would like to maximize the probability of hitting a global minimizer. Now, let us define the following quantity:

$$\delta_f(\mathbf{x}) = f(\mathbf{x}) - f_{global} \tag{1}$$

(in case the global minimum value $f_{global}$ is not known, we can substitute it with the best known value $f_{best}$). We can now define a new procedure, summarized in Algorithm 2.

**Algorithm 2** Convergence to the global optimum

1: set the max number of function evaluations for $A$ equal to $N$
2: apply $A$ to $p$ for $n$ times
3: set $j = 0$
4: **for all** $i \in [1, ..., n]$ **do**
5: $\phi(N, i) = \min f(A(N), p, i)$
6: $\mathbf{x} = arg\ \phi(N, i)$
7: compute $\delta_f(\mathbf{x})$
8: **if** $(\delta_f(\mathbf{x}) < tol_f)$ **then** $j = j + 1$
9: **end if**
10: **end for**

The output of such procedure is the fraction $j/n$ of the $n$ runs of $A$ which end up with a success, i.e. an estimate of the probability of success for $A$. A key point is properly setting the value of $n$, because a value of $n$ too small would correspond to an insufficient number of samples to have a proper statistics. This choice will be discussed in Section 2.1. The value of the tolerance parameter $tol_f$, which defines the concept of success, is problem dependent. Note that, in the case of multiple minima with equal $f$ also the distance $\|x - x_{global(best)}\|$ would be relevant, however in the following we are only interested in the value of the merit function.

## 2.1 Performance Indexes

Now that the testing procedure is defined we can define the performance indexes. For a stochastic based algorithm different performance indexes can be defined. In the following we will discuss about the significance of some of them keeping in mind the practical use of a global optimization, or global search, algorithm in space trajectory design.

The current practice is mainly focused on the evaluation of best value, mean and variance values of the best solutions found on $n$ runs [8, 7, 12].

An algorithm is considered as better performing as the obtained mean value is closer to the global optimum and a small variance is considered as a suggestion of robustness. This approach, however, does not consider three main issues: a) generally the distribution of the best values cannot be approximated with a gaussian distribution; b) from a practical standpoint, we are not interested in the mean values, which could be faraway from the global optimum; c) we want to know the level of confidence in the repetibility and global optimality of the results.

An alternative index that can be used to assess the effectiveness of a stochastic algorithm is the success rate, which is related to the $j$ value in Algorithm 2, being $Sp = j/n$. Considering the success as the referring index for a comparative assessment implies two main advantages. First, it gives an immediate and unique indication of the algorithm effectiveness, addressing all the issues highlighted above, and, second, the success rate can be represented with a binomial probability density function (PDF), independently of the number of function evaluations, the problem and the type of optimization algorithm. This latter means, moreover, that we can design the experiment and fix the number of runs, $n$, on the basis of the error we can accept on the success value. A usual starting point to sample size determination for a binomial distribution is to assume both the normal approximation for the sample proportion $p$ of successes, i.e. $p \sim N\{\theta, \theta(1-\theta)/n\}$, and the requirement that $Pr[|p-\theta| \leq d|\theta]$ should be at least $1-\alpha$ [2]. This leads to expression in (2) and to the conservative rule in (3), obtained if $\theta = 0.5$

$$n \geq \theta(1-\theta)\chi^2_{(1),\alpha}/d^2 \tag{2}$$

$$n \geq 0.25\chi^2_{(1),\alpha}/d^2 \tag{3}$$

For our tests we considered $n = 200$, which should "guarantee" an error $\leq 0.05$ ($d = 0.05$) with a 95% confidence ($\alpha = 0.05$).

## 3. Problem Description

Three different test-cases, with different difficulty levels, are considered. In all of these cases the objective will be to minimize the variation of the velocity of the spacecraft due to a propelled maneuver, $\Delta v$. Minimizing the $\Delta v$ means minimizing the propellant mass required to perform the maneuver, since propellant mass increases exponentially with $\Delta v$.

A simple, but already significant, application is to find the best launch date and time of flight to transfer a spacecraft from Earth to the asteroid Apophis. The transfer is computed as the solution of a Lambert's problem [3], therefore the design variables are the departure date from

the first celestial body, $t_0$ and the flight time $T_1$ from the first to the second body. The launch date from the Earth has been taken in the interval [3653, 10958] (number of elapsed days since January 1st 2000, MJD2000), while the time of flight has been taken in the interval [50, 900] days. The best known solution is $f_{best}$=4.3745658 km/s.

The second test-case consists of a transfer from Earth to Mars with the use of the relative movement and gravity of Venus to alter the path and speed of the spacecraft in order to save fuel. The mission is implemented as two Lambert arcs, Earth-Venus and Venus-Mars, plus a gravity assist maneuver at Venus. The problem has dimension 6, $t_0$ [d, MJD2000] [3650, 3650+365.25*15], $T_1$ [d] [50, 400], $\gamma_1$ [rad] [$-\pi$, $\pi$], $r_{p,1}$ [1, 5], $\alpha_2$ [0.01, 0.9], $T_2$ [d] [50, 700], where $\gamma_1$, $r_{p,1}$ and $\alpha_2$ are related to the gravity assist maneuver and are the angle of hyperbola plane, the radius of the pericentre of the hyperbola adimensionalised with the radius of the planet, and the fraction of time of flight before the deep space manoeuvre, respectively. The best known solution is $f_{best}$=2.9811 km/s.

The third test is a multi gravity assist trajectory from the Earth to Saturn following the sequence Earth-Venus-Venus-Earth-Jupiter-Saturn (EVVEJS). Gravity assist maneuvers have been modeled through a linked-conic approximation with powered maneuvers, i.e., the mismatch in the outgoing velocity is compensated through a $\Delta v$ maneuver at the assisting planet. No deep-space maneuvers are possible and each planet-to-planet transfer is computed as the solution of a Lambert's problem. The objective function is given in [9] and also in this case the dimensionality of the problem is 6, $t_0$ [d, MJD2000] [-1000, 0], $T_1$ [d] [30, 400], $T_2$ [d] [100, 470], $T_3$ [d] [30, 400], $T_4$ [d] [400, 2000], $T_5$ [d] [1000, 6000]. The best known solution is $f_{best}$=4.9307 km/s. Due to format requirements, it is not possible to exhaustively describe the problems, but they are freely available on request as black-box executables.

## 4. Used Algorithms

We tested three global search algorithms belonging to the class of stochastic algorithms. More precisely, two belong to the class of Evolutionary Algorithms (EAs), and one to the class of agent-based algorithms.

We considered 6 different settings for the DE, resulting from combining 3 sets of populations, $[5\,d, 10\,d, 20\,d]$, where d is the dimensionality of the problem, 2 strategies, 6 (DE, best, 1, bin) and 7 (DE, rand, 1, bin) [13], and single values of step-size and crossover probability, $F = 0.75$ and $CR = 0.8$ respectively, on the basis of common use. DEs with strat-

egy 6 are indicated, in Table 1, as Algorithms 1, 2 and 3, while those with strategy 7 are 4, 5 and 6, depending on the population size (from the smallest to the biggest).

For the Particle Swarm Optimization (PSO) algorithm ([4]), 9 different settings were considered, resulting from the combination of 3 sets of population, again $[5\,d, 10\,d, 20\,d]$, 3 values for the maximum velocity bound, $V_{max} \in [0.5, 0.7, 0.9]$ (corresponding in Table 1 to Algorithms 7-9, 10-12 and 13-15, respectively), and single values for weights, $C_1 = 1$ and $C_2 = 2$. Regarding the GA ([10]) application, only the influence of the population size was considered ([100, 200, 400] for the bi-impulse test case and [200, 400, 600] for the other two cases, corresponding to Algorithms 16-18 in Table 1), with single values for crossover and mutation probability, $Cr = 1$ and $Mp = 1/d$. All algorithms operated on normalized ([0,1]) search spaces, with random initial populations.

*Table 1.* Numbering of tested algorithms

| Alg. | Id. | Alg. | Id. | Alg. | Id. |
|---|---|---|---|---|---|
| DE(5 $d$, 6) | 1 | PSO(5 $d$, 0.5) | 7 | PSO(5 $d$, 0.9) | 13 |
| DE(10 $d$, 6) | 2 | PSO(10 $d$, 0.5) | 8 | PSO(10 $d$, 0.9) | 14 |
| DE(20 $d$, 6) | 3 | PSO(20 $d$, 0.5) | 9 | PSO(20 $d$, 0.9) | 15 |
| DE(5 $d$, 7) | 4 | PSO(5 $d$, 0.7) | 10 | GA(100) | 16 |
| DE(10 $d$, 7) | 5 | PSO(10 $d$, 0.7) | 11 | GA(200) | 17 |
| DE(20 $d$, 7) | 6 | PSO(20 $d$, 0.7) | 12 | GA(400) | 18 |

## 5. Comparison Among Performance Indexes

The results of the tests are summarized in Table 2 and 3, where success probability (Table 2) and best value, mean and variance of best results (Table 3) are given for each of 18 (set) solvers. For both EA and EVM cases, success probability allows a fair classification and gives a clear indication of the best performing algorithms. Algorithms 5 and 6 perform undoubtedly much better than the others and GAs (Algorithms 16-18) appear to be the worst performing ones. Algorithm 4 wins a bronze medal, but if we can be confident on its third position for the EVM problem, we cannot have the same level of confidence regarding the third position for EA, because of the proximity of other algorithms. Actually, due to the binomial nature of the success and the adopted sample size, it is not possible to fairly discriminate between algorithms for which the success distance is smaller than the expected error (0.05 in our computations). Therefore, Algorithm 4 has to be considered at the

same level of Algorithms 11, 13 and other PSO settings. For the same reasons, we can say that, among the PSO settings, Algorithms 11 and 13 perform better than Algorithm 8 but the remaining PSO algorithms work at the same level.

*Table 2.* Success for the 18 algorithms on the three test-cases. To compute the success, following $tol_f$ values were used: 0.001 for EA, $3 - f_{best}$ for EVM and $5 - f_{best}$ for EVVEJS

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|
| EA | 0.140 | 0.300 | 0.355 | 0.450 | 0.770 | 0.855 | 0.355 | 0.345 | 0.410 |
| EVM | 0.050 | 0.050 | 0.050 | 0.150 | 0.250 | 0.370 | 0.040 | 0.035 | 0.080 |
| EVVEJS | 0.020 | 0.005 | 0.015 | 0.000 | 0.000 | 0.000 | 0.000 | 0.005 | 0.000 |
| | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 |
| EA | 0.395 | 0.425 | 0.410 | 0.435 | 0.385 | 0.420 | 0.160 | 0.240 | 0.105 |
| EVM | 0.045 | 0.060 | 0.055 | 0.035 | 0.070 | 0.075 | 0.005 | 0.010 | 0.035 |
| EVVEJS | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 | 0.005 |

An analogous vagueness is associated to the results of a number of algorithms when applied to the EVM case and to almost all of them when applied to the EVVEJS case. For the EVVEJS transfer, in particular, the probability of success, though not always 0, is at best, including the error margin, not greater than 0.07.

For cases when the success probability cannot give practically useful information to classify the algorithms, the user could be tempted to use mean and variance values, but this practice is strongly heedless. Since, as anticipated in Section 2.1 and confirmed by tests (see Fig. 1), the PDF of the best values is not a gaussian and, moreover, changes during the process, mean and variance values are not enough to understand the algorithm behaviour and we cannot say anything about their exactness.

Even if we suppose mean and variance are correct (in some way), looking at these two values can bring to incorrect conclusions. For instance, if we consider the values for the Algorithms 14 and 17 applied to EVM, we could conclude that Algorithm 17 performs better than Algorithm 14, because of a smaller mean value and a smaller variance (regarded as an index of robustness). But if we are interested in global optimal solutions, Algorithm 14 is noticeably better: it is able to find the global solution, even if it is less robust and gets stuck many times in a far basin (see Fig. 2).

In order to solve an uncertainty condition, for instance when the success probability appears uniformly null, relaxing the $tol_f$ value could be

*Table 3.* Indexes: Best value, Mean Best, Variance Best.

| | EA (N=5000) | | | EVM (N=100000) | | | EVVEJS (N=400000) | | |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 4.3746 | 4.6962 | 0.0736 | 2.9811 | 3.6032 | 0.3240 | 4.9307 | 12.5129 | 15.0723 |
| 2 | 4.3746 | 4.5734 | 0.0312 | 2.9811 | 3.5118 | 0.0880 | 4.9307 | 11.3672 | 15.7534 |
| 3 | 4.3746 | 4.5198 | 0.0166 | 2.9811 | 3.4335 | 0.0823 | 4.9307 | 9.9694 | 15.9349 |
| 4 | 4.3746 | 4.5126 | 0.0236 | 2.9811 | 3.2936 | 0.0307 | 5.3034 | 8.1468 | 9.7486 |
| 5 | 4.3746 | 4.4197 | 0.0074 | 2.9811 | 3.2336 | 0.0277 | 5.3034 | 6.3851 | 5.0131 |
| 6 | 4.3746 | 4.3919 | 0.0026 | 2.9813 | 3.1699 | 0.0285 | 5.3034 | 5.5596 | 1.3999 |
| 7 | 4.3746 | 4.5120 | 0.0124 | 2.9811 | 3.8194 | 0.6794 | 5.0275 | 12.6827 | 16.4910 |
| 8 | 4.3746 | 4.5103 | 0.0119 | 2.9811 | 3.7812 | 0.6252 | 4.9558 | 11.9736 | 18.7031 |
| 9 | 4.3746 | 4.4990 | 0.0125 | 2.9811 | 3.6537 | 0.5233 | 5.3034 | 11.1931 | 17.9212 |
| 10 | 4.3746 | 4.5098 | 0.0142 | 2.9811 | 4.0427 | 1.0135 | 5.0125 | 11.7365 | 17.7327 |
| 11 | 4.3746 | 4.4919 | 0.0120 | 2.9811 | 3.9285 | 0.8346 | 5.0553 | 10.7274 | 17.2695 |
| 12 | 4.3746 | 4.5037 | 0.0136 | 2.9811 | 3.7329 | 0.5971 | 5.0177 | 10.4668 | 18.4276 |
| 13 | 4.3746 | 4.4959 | 0.0133 | 2.9811 | 4.2185 | 1.0569 | 5.2450 | 11.8344 | 21.7106 |
| 14 | 4.3746 | 4.5503 | 0.3720 | 2.9811 | 3.9747 | 0.8676 | 5.0223 | 10.5636 | 18.4262 |
| 15 | 4.3746 | 4.4983 | 0.0131 | 2.9811 | 3.8127 | 0.7466 | 5.0310 | 10.5256 | 15.1327 |
| 16 | 4.3746 | 4.5743 | 0.0260 | 2.9885 | 3.7821 | 0.2413 | 5.1595 | 10.6525 | 15.1862 |
| 17 | 4.3746 | 4.4959 | 0.0146 | 2.9926 | 3.5435 | 0.1400 | 5.0242 | 8.3140 | 9.9140 |
| 18 | 4.3746 | 4.4507 | 0.0084 | 2.9827 | 3.4452 | 0.0983 | 4.9821 | 6.9770 | 6.6833 |

useful. Focusing on the EVVEJS case, there is no way to correctly discriminate among the algorithms on the basis of data in Table 2, but if the success threshold is raised from 5 to 5.3, then a superior performance of GAs is revealed. Most likely, this behaviour is due to a combination of large population size, mutation operator and non-deterministic selection, which reduce the local convergence, and allows for a better exploration of the search space.

As previously stated, for all the tests, 200 runs were performed in order to maintain the error on the success probability within a predefined margin. The extreme importance of the sample size appears evident when we look at Fig. 3, where the variation of the success probability is shown as function of $n$. For $n \leq 50$, the success is extremely oscillating and the confidence on the obtained value should be considered poor.

## 6. Conclusions

The work focuses on the testing procedures for the application of global optimization algorithms to space trajectory design and tries to set the basis for a standard and consistent procedure. The current testing practice and the currently used performance indexes are criticized and a preliminary testing/analysis procedure is proposed and the probability

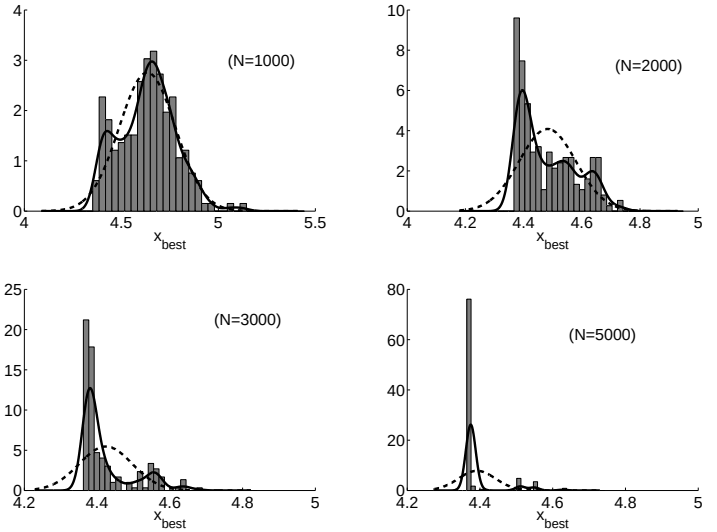
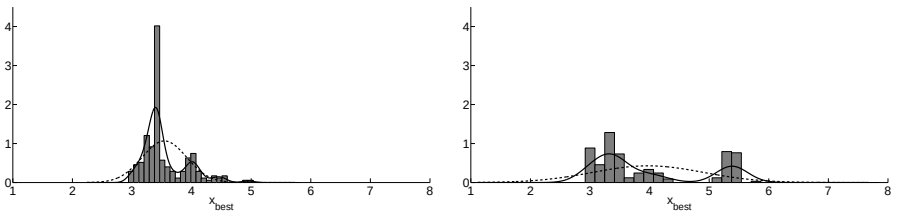


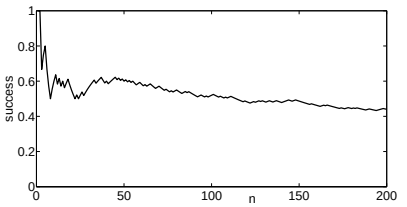

*Figure 1.* Variation of the PDF for best solutions with $N$for the Algorithm 6 applied to EA test-case; discrete, incorrect gaussian approximation (dashed) and kernel based approximation (continuous) are shown.



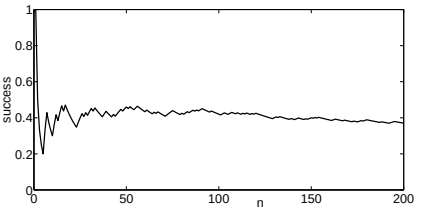

(a) GA (b) PSO

*Figure 2.* Examples of two PDFs, which could bring to incorrect conclusions; for both cases, discrete, incorrect gaussian approximation (dashed) and kernel based approximation (continuous) are shown.

of success is indicated as the most useful index, when performance of different algorithms are to be compared. Moreover, the binomial nature of this index allows to link the number of performed runs to the expected error on the success itself, while it is not possible to have the same statistical consistency when mean and variance values are utilized, because of the unknown nature of the PDF for the best values. In general, it should




(a) Alg. 13 on EA case




(b) Alg. 6 on EVM case

*Figure 3.* The influence of sample size. The success probability is shown as function of the sample size for two different algorithm/test-case combinations.

be stressed that if the comparative tests have to be reliable, the number of runs cannot be lower than a threshold depending on the nature of the considered indexes.

In the future, the testing procedure will be improved by considering also the heuristics costs and the link between the performance of some heuristics and the main structures of the test-cases.

## References

[1] O. Abdelkhalik and D. Mortari. N-Impulse Orbit Transfer Using Genetic Algorithms. *J. Spacecraft Rockets*, 44(2):456–459, 2007.

[2] C.J. Adcock. Sample size determination: a review. *The Statistician*, 46(2):261–283, 1997.

[3] H. Battin. *An Introduction to the Mathematics and Methods of Astrodynamics*. AIAA, 1999.

[4] M. Clerc. *Particle Swarm Optimization*. ISTE, 2006.

[5] V. Coverstone-Carroll. Near-optimal low-thrust orbit transfers generated by a genetic algorithm. *J. Spacecraft Rockets*, 33(6):859–862, 1996.

[6] P. Di Lizia and G. Radice. Advanced Global Optimization Tools for Mission Analysis and Design. ESA Ariadna Report ITT AO4532/18139/04/NL/MV, Call 03/4101, 2004.

[7] P.J. Gage, R.D. Braun ,and I.M. Kroo. Interplanetary trajectory optimization using a genetic algorithm. *J. Astronaut. Sci.*, 43(1):59–75, 1995.

[8] Y.H. Kim and D.B. Spencer. Optimal Spacecraft Rendezvous Using Genetic Algorithms. *J. Spacecraft Rockets*, 39(6):859–865, 2002.

[9] A.V. Labunsky, O.V. Papkov, and K.G. Sukhanov. *Multiple Gravity Assist Interplanetary Trajectories*. ESI Book Series, 1998.

[10] M. Mitchell. *An introduction to genetic algorithms*. MIT Press, 1998.

[11] D.R. Myatt, V.M. Becerra, S.J. Nasuto, and J.M. Bishop. Advanced Global Optimization Tools for Mission Analysis and Design. ESA Ariadna Report ITT AO4532/18138/04/NL/MV, Call03/4101, 2004.

[12] A.D. Olds, C.A. Kluever, and M.L. Cupples. Interplanetary Mission Design Using Differential Evolution. *J. Spacecraft Rockets*, 44(5):1060–1070, 2007.

[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] M. Vasile, L. Summerer, and P. De Pascale. Design of Earth-Mars Transfer Trajectories using Evolutionary Branching Techniques. *Acta Astronaut.*, 56(8):705–720, 2005.

# A DISCRETE PARTICLE SWARM OPTIMIZER FOR CLUSTERING SHORT-TEXT CORPORA

Leticia C. Cagnina, Marcelo L. Errecalde, Diego A. Ingaramo
*LIDIC, Universidad Nacional de San Luis*
*San Luis, Argentina*
{lcagnina; merreca; daingara}@unls.edu.ar

Paolo Rosso
*Natural Language Engineering Lab., DSCI, Universidad Politécnica de Valencia*
*Valencia, España*
prosso@dsic.upv.es

**Abstract** Work on "short-text clustering" is relevant, particularly if we consider the current/future mode for people to use "small-language", e.g. blogs, text-messaging, snippets, etc. Potential applications in different areas of natural language processing may include re-ranking of snippets in information retrieval, and automatic clustering of scientific texts available on the Web. Despite its relevance, this kind of problems has not received too much attention by the computational linguistic community due to the high challenge that this problem implies. In this work, we propose the CLUDIPSO algorithm, a novel approach for clustering short-text collections based on a discrete Particle Swarm Optimizer. Our approach explicitly considers clustering as an optimization problem where a given arbitrary objective function must be optimized. We used two unsupervised measures of cluster validity with this purpose: the *Expected Density Measure* and the *Global Silhouette* coefficient. These measures have shown interesting results in recent works on short-text clustering. The results indicate that our approach is a highly competitive alternative to solve this kind of problems.

**Keywords:** Clustering as optimization, Particle swarm optimization, Short-text clustering

## 1. Introduction

In clustering tasks the main goal is that the objects within a group be similar (or related) to one another and different from (or unrelated to) the objects in other groups [22]. When clustering tasks involve documents, different aspects can negatively affect the similarity estimation between documents and, in consequence, document clustering is usually harder than other problems addressed in cluster analysis research.

In those cases where clustering techniques are applied to collections containing *very short* documents, additional difficulties are introduced due to the low frequencies of the document terms. Research work on "short-text clustering" (that is, clustering of short-length documents) is relevant, particularly if we consider the current/future mode for people to use 'small-language', e.g. blogs, text-messaging, snippets, etc. Potential applications in different areas of natural language processing may include re-ranking of snippets in information retrieval, and automatic clustering of scientific texts available on the Web.

Clustering of short-text collections is one of the most difficult tasks in natural language processing and, in this work, we propose a new discrete Particle Swarm Optimizer algorithm for this kind of problems. Our approach explicitly considers clustering as an optimization problem where a given arbitrary objective function must be optimized. We used two unsupervised measures of cluster validity with this purpose, which have shown interesting results in recent works on short-text clustering.

The remainder of the paper is organized as follows. Section 2 presents some considerations about the particularities that arise when considering clustering as an optimization problem; here, we also describe the cluster validity measures that were used as objective function to be optimized. Section 3 describes in detail our proposed approach. In Section 4 some general features of the corpora used in the experiments are presented. The experimental setup and the analysis of the results obtained from our empirical study is provided in Section 5. Finally, some general conclusions are drawn and possible future work is discussed.

## 2. Clustering as Optimization

Document clustering consists in the assignment of documents to unknown categories. This task is more difficult than supervised text categorization because the information about categories and correctly categorized documents is not provided in advance. An important consequence of this lack of information is that in realistic document clustering problems, results cannot usually be evaluated with typical *external* measures like $F$-Measure or the Entropy, because the correct catego-

rizations specified by a human editor are not available. Therefore, the quality of the resulting groups is evaluated with respect to *structural* properties expressed in different *Internal Clustering Validity Measures* (ICVMs). Classical ICVMs used as cluster validity measures include the *Dunn* and *Davies-Bouldin* indexes, the *Global Silhouette* (GS) coefficient and new graph-based measures like the *Expected Density Measure* (EDM) (denoted $\bar{\rho}$) and the $\lambda$-Measure [20] (see [13] and [20] for more detailed descriptions of these ICVMs).

These unsupervised measures of cluster validity -or any arbitrary criterion function that gives a reasonable estimation of the quality of the obtained groups- can be used as an objective function whose optimization drives the entire clustering process. In this approach, adopted by diverse algorithms (e.g. K-means [14], Cobweb [10], Autoclass [3] and CLUTO [25]) the criterion function is explicit and can be easily stated. As observed in [25], this class of algorithms can be thought of as consisting of two key components: 1) the criterion function that the clustering solution optimizes, and 2) the actual algorithm that achieves this optimization.

For the first issue we selected the GS coefficient and the EDM $\bar{\rho}$ [20, 22], two ICVMs that have shown an adequate *correlation* degree with the categorization criteria of a human editor in recent works on clustering of short-text corpora [8, 13].

The GS measure is obtained computing the average cluster silhouette of all found clusters. The cluster silhouette of a cluster $C$ is the average silhouette coefficient of all objects belonging to $C$. The silhouette coefficient for the object $i$ is obtained as follows: $s(i) = \frac{b(i)-a(i)}{max(a(i),b(i))}$ with $-1 \leq s(i) \leq 1$. The $a(i)$ value denotes the average dissimilarity of the object $i$ to the remaining objects in its own cluster, and $b(i)$ is the average dissimilarity of object $i$ to all objects in the nearest cluster.

The EDM $\bar{\rho}$ of a clustering $\mathcal{C}$ is: $\overline{\rho}(\mathcal{C}) = \sum_{i=1}^{k} \frac{|V_i|}{|V|} \cdot \frac{w(G_i)}{|V_i|^{\theta}}$. $\mathcal{C} = \{C_1, .., C_k\}$ is the clustering of a weighted graph $G = \langle V, E, w \rangle$ and $G_i = \langle V_i, E_i, w_i \rangle$ is the induced subgraph of $G$ with respect to cluster $C_i$. The density $\theta$ of the graph from the equation $|E| = |V|^{\theta}$ where $w(G) = |V| + \sum_{e \in E} w(e)$, is computed as: $w(G) = |V|^{\theta} \Leftrightarrow \theta = \frac{\ln(w(G))}{\ln(|V|)}$.

An important issue to be considered is that those ICVMs can be used for driving or for evaluating the clustering algorithms but the real effectiveness of these algorithms only can be evaluated with external measures that incorporate the categorization criteria of the users. A very popular external measure used at this end is the $F$-measure.

In the context of clustering, $F$-Measure is an external validity measure that combines both, *precision* and *recall*. It may be formally defined as

follows. Let $D$ represents the set of documents, $\mathcal{C} = \{C_1, \ldots, C_k\}$ be a clustering of $D$ and $\mathcal{C}^* = \{C_1^*, \ldots, C_l^*\}$ designates the human reference classification of $D$. The *recall* of a cluster $j$ with respect to a class $i$, $rec(i,j)$ is defined as $|C_j \cap C_i^*|/|C_i^*|$. The *precision* of a cluster $j$ with respect to a class $i$, $prec(i,j)$ is defined as $|C_j \cap C_i^*|/|C_j|$. Thus, the $F$-measure of the cluster $j$ with respect to a class $i$ is $F_{i,j} = \frac{2 \cdot prec(i,j) \cdot rec(i,j)}{prec(i,j)+rec(i,j)}$ and the overall $F$-measure is defined as: $F = \sum_{i=1}^{l} \frac{|C_i^*|}{|D|} \cdot \max_{j=1,..,k}\{F_{i,j}\}$ A clustering result with an $F$-measure value equals to 1 corresponds to a "ideal" clustering, i.e., a grouping that exactly matches the clustering specified by a human expert.

With respect to the algorithm used for optimizing the EDM $\bar{\rho}$ and GS measures, we will describe our approach in the next section.

## 3. Our Proposed Approach: CLUDIPSO

Different *Particle Swarm Optimization* (PSO) approaches have been previously proposed in the literature to solve the clustering problem in general. However, few adaptations have been presented for document clustering. A PSO-based clustering algorithm that outperforms the $K$-means algorithm in image classification tasks is proposed in [16]. Van der Merwe and Engelbrecht presented an hybridization of the PSO and $K$-means algorithms for clustering general datasets. Basically, the result obtained by a $K$-means algorithm is used as a single particle in the initial swarm of the PSO algorithm [23]. In [24], Xiao presents an hybrid adaptation, based on the synergism of a PSO algorithm and a Self Organizing Map for clustering gene expression data. Cui proposes in [5] an hybrid method based on the combination of a PSO and a $K$-means algorithm in document clustering tasks. Firstly, a global search process is carried out by the PSO algorithm. Then, the best result obtained by the PSO algorithm is used by the K-means algorithm for determining the initial centroids.

Discrete PSO implementations were suggested in the research community for different combinatorial optimization problems [11, 4]. However, as far as we know, no approaches have been used for clustering short-text corpora.

Our proposal for this problem, named CLUDIPSO (CLUstering with a DIscrete PSO), is based on a PSO algorithm that operates on a population of particles. In CLUDIPSO, each valid clustering is represented as a particle. The particles are $n$-dimensional integer vectors, where $n =$ number of documents in the collection. The best position found so far for the swarm (*gbest*) and the best position reached by each particle (*pbest*) are recorded. The particles evolve at each iteration using two updat-

ing formulas, one for velocity (Equation (2)) and another for position. Since the task was modeled with a discrete approach, a new formula was developed for updating the positions (shown in Equation (1)). This modification was introduced to accelerate the convergence velocity of the algorithm (principal incoming of discrete PSO models).

$$par_{id} = pb_{id} \tag{1}$$

$$v_{id} = w(v_{id} + \gamma_1(pb_{id} - par_{id}) + \gamma_2(pg_d - par_{id})) \tag{2}$$

where $par_{id}$ is the value of the particle $i$ at the dimension $d$, $v_{id}$ is the velocity of particle $i$ at the dimension $d$, $w$ is the inertia factor [6] whose goal is to balance global exploration and local exploitation, $\gamma_1$ is the personal learning factor, and $\gamma_2$ the social learning factor, both multiplied by 2 different random numbers within the range [0..1]. $pb_{id}$ is the best position reached by the particle $i$ and $pg_d$ is the best position reached by any particle in the swarm.

It is important to note that in our approach the process of updating particles is not as direct as in the continuous case. In CLUDIPSO, the updating process is not carried out on all dimensions at each iteration. In order to determine which dimensions of a particle will be updated we do the following steps: 1) all dimensions of the velocity vector are normalized in the [0..1] range, according to the process proposed by Hu et al. [12] for a discrete PSO version; 2) a random number $r \in [0..1]$ is calculated; 3) all the dimensions (in the velocity vector) higher than $r$ are selected in the position vector, and updated using the Equation (1).

To help avoiding convergence to a local optimum, we used a dynamic mutation operator [2] which is applied to each individual with a *pm*-probability. This value is calculated considering the total number of iterations in the algorithm (*cycles*) and the current cycle number as the Equation (3) indicates:

$$pm = max\_pm - \frac{max\_pm - min\_pm}{max\_cycle} * current\_cycle \tag{3}$$

where $max\_pm$ and $min\_pm$ are the maximum and minimum values that $pm$ can take, $max\_cycle$ is the total number of cycles that the algorithm will iterate, and $current\_cycle$ is the current cycle in the iterative process. The mutation operation is applied if the particle is the same that its own *pbest*, as was suggest by [12]. The mutation operator swaps two random dimensions of the particle.

## 4. Data Sets

The complexity of clustering problems with short-text corpora demands a meticulous analysis of the features of each collection used in the experiments. For this reason, we will focus on specific characteristics of the collections such as document lengths and its closeness with respect to the topics considered in these documents. We attempt with this decision to avoid introducing other factors that can make incomparable the results.

We select for the experimental work the CICling-2002 collection, perhaps the only short-text collection that has been considered in a significative number of research works on short-text clustering [15, 1, 17, 13, 8]. CICling-2002 corpus is considered a high complexity collection since its documents are narrow domain scientific abstracts (short-length documents with an high vocabulary overlapping). Our choice of this collection is not casual. In the majority of works that have used CICling-2002, this corpus has shown a higher difficulty degree than the other collections considered. Therefore, if a good performance on this collection is achieved, we can be confident that good results will be also obtained with other easier corpus.

In order to verify this last assertion we also used the Micro4News corpus, a collection recently proposed in [8]. Micro4News is a collection significatively easier than CICling-2002 with respect to the length of documents and vocabulary overlapping. However, other features such as the number of groups and number of documents per group were maintained the same for both collections in order to obtain comparable results.

Space limitations prohibit a more detailed explanation of these corpora, but the interested reader can obtain more information in [7].

## 5. Parameter Settings and Analysis of Results

The documents of CICling-2002 and Micro4News used in the experiments were represented using the popular *Vector Space Model* and the "SMART codifications" [19] associated. In this case, we used the cosine similarity and the codification *ntc* that refers to the scheme where the weight for the $i$-th component of the vector for the document $d$ is computed as $tf_{d,i} \times \log(\frac{N}{df_i})$ and then cosine normalization is applied. Here, $N$ denotes the number of documents in the collection, $tf_{d,i}$ is the term frequency of the $i$-th term in the document $d$ and $df_i$ refers to the document frequency of $i$-th term over the collection.

We performed 50 independent runs per problem, with 10,000 iterations (*cycles*) per run. CLUDIPSO used the following parameters: swarm size = 50 particles, dimensions at each particle = number of doc-

uments ($N$), $pm\_min = 0.4$, $pm\_max = 0.9$, inertia factor $w = 0.9$, personal and social learning factors for $\gamma_1$ and $\gamma_2$ were set to 1.0. The parameter settings were empirically derived after numerous experiments.

Our results were compared with the results obtained with other three clustering algorithms: $K$-means, MajorClust [21] and DBSCAN [9]. $K$-means is one of the most popular clustering algorithms and, MajorClust and DBSCAN are representative of the density-based approach to the clustering problem. Basically, these two last algorithms attempt to separate the set of objects (documents) into subsets of similar densities. Our motivation for choosing these two density-based algorithms was to compare the performance of our algorithm with other approaches that also attempt to maximize the density of the resulting groups. Furthermore, MajorClust has shown in recent works to be one of the most successful algorithms for document clustering in general and short-text clustering problems in particular. A significative difference between the algorithms considered is whether the algorithm requires information about the number correct of groups ($k$) or not. This information has to be provided to $K$-means and CLUDIPSO but MajorClust and DBSCAN determine the cluster number $k$ automatically.

## 5.1 CICLing2002

We focus our analysis of the results obtained by the different algorithms considering the EDM $\bar{\rho}$ (that we will refer as $\bar{\rho}$ from now on) and $F$-measure values (Table 1) and the GS and $F$-measure values (Table 2). The $F$-measure values are listed in order to show the correlation between these and the metric values. In Table 1 we can observe that CLUDIPSO and MajorClust obtain the highest values of $\bar{\rho}_{avg}$ and $\bar{\rho}_{min}$ for this collection. However, CLUDIPSO is outperformed by both density-based algorithms (DBSCAN and MajorClust) if we consider the results of $\bar{\rho}_{max}$. In order to understand this last result, it is important to consider that both density-based algorithms can generate clusterings with different number of groups. Furthermore, in previous works we have observed that higher values of $\bar{\rho}$ can usually be obtained when the result has a smaller number of groups. CLUDIPSO and $k$-means only can generate clusterings with a fixed number of groups and, therefore, it is impossible for these algorithms to reach these $\bar{\rho}$ values. In that sense, the GS measure is not affected by the number of clusters obtained and thus, it can be more informative to consider the GS values shown in Table 2. In this table, we can observe that CLUDIPSO clearly outperforms the GS values of all the remaining algorithms.

*Table 1.* CICLing2002: $\bar{\rho}$ and $F$-measure values for the different algorithms

| **Algorithm** | $\bar{\rho}_{avg}$ | $\bar{\rho}_{min}$ | $\bar{\rho}_{max}$ | $F_{avg}$ | $F_{min}$ | $F_{max}$ |
|---|---|---|---|---|---|---|
| K-Means | 0.87 | 0.84 | 0.91 | 0.46 | 0.35 | 0.57 |
| MajorClust | 0.92 | 0.91 | 0.94 | 0.43 | 0.37 | 0.58 |
| DBSCAN | 0.91 | 0.88 | 0.95 | 0.47 | 0.42 | 0.56 |
| CLUDIPSO | 0.92 | 0.91 | 0.93 | 0.63 | 0.42 | 0.74 |

*Table 2.* CICLing2002: GS and $F$-measure values for the different algorithms

| **Algorithm** | $GS_{avg}$ | $GS_{min}$ | $GS_{max}$ | $F_{avg}$ | $F_{min}$ | $F_{max}$ |
|---|---|---|---|---|---|---|
| K-Means | 0.07 | -0.06 | 0.22 | 0.46 | 0.35 | 0.57 |
| MajorClust | 0.14 | -0,24 | 0.36 | 0.43 | 0.37 | 0.58 |
| DBSCAN | 0.08 | -0.11 | 0.21 | 0.47 | 0.42 | 0.56 |
| **CLUDIPSO** | **0.39** | **0.36** | **0.41** | **0.6** | **0.5** | **0.72** |

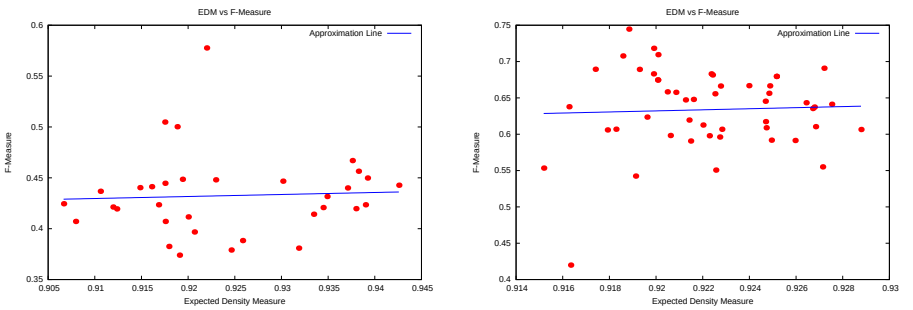



*Figure 1.* CICLing2002: $\bar{\rho}$ vs $F$-measure for MajorClust (left) and CLUDIPSO (right).

On other hand, if we now consider the $F$-measure values obtained by CLUDIPSO when it used $\bar{\rho}$ (Table 1) and GS (Table 2) as objective functions, in both cases this algorithm obtained excellent results with respect to the $F$-measure, outperforming the remaining algorithms considered. We can also appreciate this good performance of CLUDIPSO with respect to MajorClust in the Fig. 1 where the $F$-measure values obtained by CLUDIPSO and MajorClust are compared. Here, we can observe that the majority of results produced by CLUDIPSO have $F$-measure values greater than 0.55. These results differ significatively of those obtained by MajorClust, which obtains a majority of $F$-measure values lower than 0.5.

### 5.2 Micro4News

An obvious question that arises from the previous experiments is if the good performance of CLUDIPSO with respect to the other algorithms considered can also be expected with other more simple collections. Therefore, we also analyze the results obtained with Micro4News, a collection with documents significatively larger than CiCling-2002 that refer to well differentiated topics. In this case, the results shown in the Tables 3 and 4 are similar to those obtained in the previous collection respect to the $\bar{\rho}$ and the GS measures. With respect to the $F$-measure, CLUDIPSO once more achieves very high values, and it reaches (when GS is used as objective function) the highest possible $F$-measure value ($F_{max} = 1$) that corresponds to the optimum clustering (respect to the criteria of a human expert). Based on these results we can conclude that CLUDIPSO, used as an optimization algorithm of different ICVMs like $\bar{\rho}$ or GS, gives very good $F$-measure values in collections with diverse complexity levels. This suggests that the mechanisms used in this algorithm for clustering of documents usually agree with the grouping criteria of a human expert and it deserves additional research work.

*Table 3.* Micro4News: $\bar{\rho}$ and $F$-measure values for the different algorithms

| **Algorithm** | $\bar{\rho}_{avg}$ | $\bar{\rho}_{min}$ | $\bar{\rho}_{max}$ | $F_{avg}$ | $F_{min}$ | $F_{max}$ |
|---|---|---|---|---|---|---|
| K-Means | 0.99 | 0.89 | 1.07 | 0.69 | 0.46 | 0.96 |
| MajorClust | 1.08 | 1.05 | 1.1 | 0.9 | 0.76 | 0.96 |
| DBSCAN | 1.05 | 1.01 | 1.1 | 0.82 | 0.71 | 0.88 |
| CLUDIPSO | 1.07 | 1.06 | 1.07 | 0.93 | 0.87 | 0,96 |

*Table 4.* Micro4News: GS and $F$-measure values for the different algorithms

| **Algorithm** | $GS_{avg}$ | $GS_{min}$ | $GS_{max}$ | $F_{avg}$ | $F_{min}$ | $F_{max}$ |
|---|---|---|---|---|---|---|
| K-Means | 0.39 | 0.05 | 0.74 | 0.69 | 0.46 | 0.96 |
| MajorClust | 0.69 | 0.64 | 0.74 | 0.9 | 0.76 | 0.96 |
| DBSCAN | 0.54 | 0.36 | 0.67 | 0.82 | 0.71 | 0.88 |
| **CLUDIPSO** | **0.72** | **0.69** | **0.74** | **0.93** | **0.85** | **1** |

## 6. Conclusions and Future Works

In this work we present two new ideas for clustering short-text corpora: 1) a novel discrete PSO-based algorithm adapted for this kind of

problems (CLUDIPSO) and 2) the use of two interesting ICVMs ($\bar{\rho}$ and GS) as an explicit objective function to be optimized. The results obtained by CLUDIPSO indicate that our approach is a highly competitive alternative to solve problems of clustering short-text corpora.

At the present time, we are testing our approach with other short-text collections and we are also defining a new continuous PSO version that uses the $\bar{\rho}$ and GS ICVMs.

## Acknowledgment

The authors acknowledge partial support by the MCyT TIN2006-15265-C06-04 project, the ANPCyT and the Universidad Nacional de San Luis.

## References

[1] M. Alexandrov, A. Gelbukh, and P. Rosso. An Approach to Clustering Abstracts. *Lect. Notes Comp. Sc.*, 3513:8–13, 2005.

[2] L. Cagnina, S. Esquivel, and R. Gallard. Particle Swarm Optimization for Sequencing Problems: a Case Study. In *Proc. IEEE Congress on Evolutionary Computation (CEC 2004)*, pages 536–541, Portland, OR, 2004.

[3] P. Cheeseman and J. Stutz. Bayessian Classification (AutoClass): Theory and Results. In U. Fayyad, G. Piatetsky-Shapiro, P. Smith, and R. Uthurusamy, (editors), *Advances in Knowledge Discovery and Data Mining*, pages 153–180. 1996.

[4] M. Clerc. Discrete Particle Swarm Optimization Illustrated by the Traveling Salesman Problem. In *New optimization techniques in engineering*, pages 219–239. 2004.

[5] X. Cui, T. Potok, and P. Palathingal. Document Clustering using Particle Swarm Optimization. In *Proc. of IEEE Swarm Intelligence Symposium (SIS-2005)*, 2005.

[6] R. Eberhart and Y. Shi. A Modified Particle Swarm Optimizer. In *Proc. International Conference on Evolutionary Computation*, Anchorage, AK, 1998.

[7] M. Errecalde and D. Ingaramo. *Short-text Corpora for Clustering Evaluation*, LIDIC, 2008. `http://www.dirinfo.unsl.edu.ar/~ia/resources/shortexts.pdf`

[8] M. Errecalde, D. Ingaramo, and P. Rosso. Proximity Estimation and Hardness of Short-text Corpora. In *Proc. 5th International Workshop on Text-based Information Retrieval (TIR-2008)*, 2008.

[9] M. Ester, H. Kriegel, J. Sander, and X. Xu. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In *Proc. 2nd International Conference on Knowledge Discovery and Data Mining (KDD-96)*, pages 226–231, 1996.

[10] D. Fisher. Knowledge Acquisition via Incremental Conceptual Clustering. *Mach. Learn.*, 2(2):139–172, 1987.

[11] A. Guner and M. Sevkli. A Discrete Particle Swarm Optimization Algorithm for Uncapacitated Facility Location Problem. In *J. Artif. Evol. Appl.*, Article ID 861512, 2008.

[12] X. Hu, R. Eberhart, and Y. Shi. Swarm Intelligence for Permutation Optimization: a Case Study on n-queens Problem. In *Proc. IEEE Swarm Intelligence Symposium*, pages 243–246, 2003.

[13] D. Ingaramo, D. Pinto, P. Rosso, and M. Errecalde. Evaluation of Internal Validity Measures in Short-Text Corpora. *Lect. Notes Comp. Sc.*, 4919:555–567, 2008.

[14] J. MacQueen. Some Methods for Classification and Analysis of Multivariate Observations. In *Proc. 5th Symp. Math. Statis, Prob.*, pages 281–297, 1967.

[15] P. Makagonov, M. Alexandrov, and A. Gelbukh. Clustering Abstracts Instead of Full Texts. *Lect. Notes Comp. Sc.*, 3206:129–135, 2004.

[16] M. Omran, A. Engelbrecht, and A. Salman. Particle Swarm Optimization Method for Image Clustering. In *Int. J. Pattern Recogn.*, 19(3):297–321, 2005.

[17] D. Pinto, J. Benedí, and P. Rosso. Clustering Narrow-domain Short Texts by Using the Kullback-Leibler Distance. *Lect. Notes Comp. Sc.*, 4394:611–622, 2007.

[18] D. Pinto and P. Rosso. On the Relative Hardness of Clustering Corpora. *Lect. Notes Comp. Sc.*, 4629:155–161, 2007.

[19] G. Salton. *The Smart Retrieval System: Experiments in Automatic Document Processing*. Prentice Hall, 1971.

[20] B. Stein, S. Meyer zu Eissen, and F. Wisbrock. On Cluster Validity and the Information Need of Users. In *Proc. 3rd IASTED Artificial Intelligenece and Applications Conference*, pages 216–221, 2003.

[21] B. Stein and O. Niggemann. On the Nature of Structure and its Identification. *Lect. Notes Comp. Sc.*, 1665:122–134, 1999.

[22] P.-N. Tan, M. Steinbach, and V. Kumar. *Introduction to Data Mining*. Addison-Wesley, 2006.

[23] D. Van and A. Engelbrecht. Data Clustering using Particle Swarm Optimization. In *Proc. IEEE Congress on Evolutionary Computation (CEC 2003)*, pages 215–220, Canbera, Australia, 2003.

[24] X. Xiao, E. Dow, R. Eberhart, Z. Ben Miled, and R. Oppelt. Gene Clustering Using Self-Organizing Maps and Particle Swarm Optimization. In *Proc. of the 17th International Symposium on Parallel and Distributed Processing (IPDPS '03)*, Nice, France, 2003.

[25] Y. Zhao and G. Karypis. Empirical and Theoretical Comparison of Selected Criterion Functions for Document Clustering. *Mach. Learn.*, 55(3):311–331, 2004.

# III

# APPLICATIONS

# SOLVING ENGINEERING OPTIMIZATION PROBLEMS WITH THE SIMPLE CONSTRAINED PARTICLE SWARM OPTIMIZER

Leticia C. Cagnina, Susana C. Esquivel
*LIDIC, Universidad Nacional de San Luis*
*San Luis, Argentina*
{lcagnina; esquivel}@unsl.edu.ar

Carlos A. Coello Coello
*CINVESTAV-IPN (Evolutionary Computational Group)*
*Mexico D. F., Mexico*
ccoello@cs.cinvestav.mx

**Abstract** This paper introduces a particle swarm optimization algorithm to solve constrained engineering optimization problems. The proposed approach uses a relatively simple method to handle constraints and a different mechanism to update the velocity and position of each particle. The algorithm is validated using four standard engineering design problems reported in the specialized literature and it is compared with respect to algorithms representative of the state-of-the-art in the area. Our results indicate that the proposed scheme is a promising alternative to solve this sort of problems because it obtains good results with a low number of objective functions evaluations.

**Keywords:** Constrained optimization, Engineering problems, Particle swarm optimization

## 1. Introduction

Engineering design optimization problems are normally adopted in the specialized literature to show the effectiveness of new constrained optimization algorithms. These nonlinear engineering problems have been investigated by many researchers that used different methods to solve them: Branch and Bound using SQP [24], Recursive Quadratic Pro-

gramming [9], Sequential Linearization Algorithm [20], Integer-discrete-continuous Nonlinear Programming [11], Nonlinear Mixed-discrete Programming [19], Simulated Annealing [27], Genetic Algorithms [26], Evolutionary Programming [8] and, Evolution Strategies [25] among many others. These types of problems normally have mixed (e.g., continuous and discrete) design variables, nonlinear objective functions and nonlinear constraints, some of which may be active at the global optimum. Constraints are very important in engineering design problems, since they are normally imposed on the statement of the problems and sometimes are very hard to satisfy, which makes the search difficult and inefficient.

Particle Swarm Optimization (PSO) is a relatively recent bio-inspired metaheuristic, which has been found to be highly competitive in a wide variety of optimization problems. However, its use in engineering optimization problems and in constrained optimization problems, in general, has not been as common as in other areas (e.g., for adjusting weights in a neural network). The approach described in this paper contains a constraint-handling technique as well as a mechanism to update the velocity and position of the particles, which is different from the one adopted by the original PSO.

This paper is organized as follows. Section 2 briefly discusses the previous related work. Section 3 describes in detail our proposed approach. Section 4 presents the experimental setup adopted and provides an analysis of the results obtained from our empirical study. Our conclusions and some possible paths for future research are provided in Section 5.

## 2. Literature Review

Guo et al. presented a hybrid swarm intelligent algorithm with an improvement in global search reliability. They tested the algorithm with two of the problems adopted here (E02 and E04). Despite they claim that their algorithm is superior for finding the best solutions (in terms of quality and robustness), the solution that they found for E02 is greater than its best known value and for E04 the results obtained are not comparable to ours, because they used more constraints in the definition of that problem [13].

Shamim et al. proposed a method based on a socio-behavioral simulation model. The idea behind this approach is that the leaders of all societies interact among themselves for the improvement of the society. They tested their algorithm using three of the problems adopted here (E01, E02 and E03). The best values reported for these three problems

are close from the optimal known values. The number of fitness function evaluations was 19,259 for E01, 19,154 for E02 and 12,630 for E03 [1].

Mahdavi et al. developed an improved harmony search algorithm with a novel method for generating new solutions that enhances the accuracy and the convergence rate of the harmony search. They used three of the problems adopted here (E01, E03 and E04) to validate their approach, performing 300,000, 200,000 and 50,000 evaluations, respectively. For E01 and E02, the best values reported are not the best known values because the ranges of some variables in E01 are different from those of the original description of the problem ($x_4$ is out of range), which makes such solution infeasible under the description adopted here. The value reported by them for E04 is very close to the best value known [21].

Bernardino et al. hybridized a genetic algorithm embedding an artificial immune system into its search engine, in order to help moving the population into the feasible region. The algorithm was used to solve four of the test problems adopted here (E01, E02, E03 and E04), using 320,000, 80,000, 36,000 and 36,000 evaluations of the objective functions, respectively. The best values found for E01, E02 and E04 are close to the best known. For E03 the value reported is better than the best known, because one of the decision variables is out of range ($x_5$). The values in general, are good, although the number of evaluations required to obtain them is higher than those required by other algorithms [4].

Hernandez Aguirre et al. proposed a PSO algorithm with two new perturbation operators aimed to prevent premature convergence, as well as a new neighborhood structure. They used an external file to store some particles and, in that way, extend their life after the adjustment of the tolerance of the constraints. The authors reference three algorithms which obtained good results for the problems adopted in their study: two PSO-based algorithms and a Differential Evolution (DE) algorithm. One of the PSO-based approaches compared [16] used three of the problems adopted here (E01, E02 and E04), performing 200,000 objective function evaluations. The other PSO-based approach compared [14] was tested with the same set of problems and the best known values were reached for E02 and E04 after 30,000 objective function evaluations. The DE algorithm [22] reported good results with 30,000 evaluations for the four problems. This same number of evaluations was performed by the algorithm proposed by Hernandez et al. and their results are the best reported until now for the aforementioned problems [15].

For that reason, we used these last two algorithms to compare the performance of our proposed approach. The DE algorithm [22] will be referenced as "Mezura" and, the PSO by [15] as "COPSO".

## 3. Our Proposed Approach: SiC-PSO

The particles in our proposed approach (called ***Si****mple* ***C****onstrained* ***P****article* ***S****warm* ***O****ptimizer*, or SiC-PSO), are $n$-dimensional values (continuous, discrete or a combination of both) vectors, where $n$ refers to the number of decision variables of the problem to be solved. Our approach adopts one of the most simple constraint-handling methods currently available. Particles are compared by pairs: 1) if the two particles are feasible, we choose the one with a better fitness function value; 2) if the two particles are infeasible, we choose the particle with the lower infeasibility degree; 3) if one particle is feasible and the other is infeasible, we choose the feasible one. This strategy is used when the *pbest*, *gbest* and *lbest* particles are chosen. When an individual is found infeasible, the amount of violation (this value is normalized with respect to the largest violation stored so far) is added. So, each particle saves its infeasibility degree reached until that moment.

As in the basic PSO [10], our proposed algorithm records the best position found so far for each particle (*pbest* value) and, the best position reached by any particle into the swarm (*gbest* value). In other words, we adopt the *gbest* model. But in previous works, we found that the *gbest* model tends to converge to a local optimum very often [7]. Motivated by this, we proposed a formula to update the velocity, using a combination of both the *gbest* and the *lbest* models [5]. Such a formula (equation 1) is adopted here as well. The *lbest* model is implemented using a ring topology [17] to calculate the neighborhoods of each particle. For a size of neighborhood of three particles and a swarm of six particles (1,2,3,4,5,6), the neighborhoods considered are the following: (1,2,3), (2,3,4), (3,4,5), (4,5,6), (5,6,1) and (6,1,2). The formula for updating particles is the same that in the basic PSO and it is shown in equation 2.

$$v_{id} = w(v_{id} + c_1 r_1 (pb_{id} - p_{id}) + c_2 r_2 (pl_{id} - p_{id}) + c_3 r_3 (pg_d - p_{id})) \quad (1)$$

$$p_{id} = p_{id} + v_{id} \quad (2)$$

where $v_{id}$ is the velocity of the particle $i$ at the dimension $d$, $w$ is the inertia factor [10] whose goal is to balance the global exploration and the local exploitation, $c_1$ is the personal learning factor, and $c_2$, $c_3$ are the social learning factors, $r_1$, $r_2$ and $r_3$ are three random numbers within the range [0..1], $pb_{id}$ is the best position reached by the particle $i$, $pl_{id}$ is the best position reached by any particle in the neighborhood of particle $i$ and, $pg_d$ is the best position reached by any particle in the swarm. Finally, $p_{id}$ is the value of the particle $i$ at the dimension $d$.

We empirically found that for some difficult functions, a previous version of our algorithm could not find good values. The reason was its diversification of solutions which kept the approach from converging. In SiC-PSO we changed the common updating formula (equation 2) of the particles for the update equation presented by Kennedy [18]. In Kennedy's algorithm, the new position of each particle is randomly chosen from a Gaussian distribution with the mean selected as the average between the best position recorded for the particle and the best in its neighborhood. The standard deviation is the difference between these two values. We adapted that formula adding the global best (*gbest*) to the best position of the particle and the best in its neighborhood. We also changed the way in which the standard deviation is determined. We use the *pbest* and, the *gbest* instead of the *lbest* as was proposed by Kennedy. We determined those changes after several empirical tests with different Gaussian random generator parameters. Thus, the position is updated using the following equation:

$$p_i = N\left(\frac{p_i + pl_i + pg}{3}, |p_i - pg|\right) \tag{3}$$

where $p_i$, $pl_i$ and $pg$ are defined as before and, $N$ is the value returned by the Gaussian random generator. SiC-PSO used equation 3 and equation 2 for the updating of positions of the particles. We considered a high probability for selecting equation 3 (0.925) over equation 2. We chose that probability after conducting numerous experiments.

## 4. Parameter Settings and Analysis of Results

A set of 4 engineering design optimization problems was chosen to evaluate the performance of our proposed algorithm. A detailed description of the test problems may be consulted in the appendix at the end of this paper. We performed 30 independent runs per problem, with a total of 24,000 objective function evaluations per run. We also tested the algorithm with 27,000 and 30,000 evaluations of the objective function, but no performance improvements were noticed in such cases. Our algorithm used the following parameters: swarm size = 8 particles, neighborhood size = 3, inertia factor $w = 0.8$, personal learning factor and social learning factors for $c_1$, $c_2$ and $c_3$ were set to 1.8. These parameter settings were empirically derived after numerous previous experiments.

Our results were compared with respect to the best results reported in the specialized literature. Those values were obtained by Hernandez Aguirre et al. [15] and Mezura et al. [22]. We reference those results into the tables shown next as "COPSO" and "Mezura", respectively. It

is important remark that COPSO and Mezura algorithms reached the best values after 30,000 fitness function evaluations, which is a larger value than that required by our algorithm. The best values are shown in Table 1 and, the mean and standard deviations over the 30 runs are shown in Table 2.

*Table 1.* Best results obtained by SiC-PSO, COPSO and Mezura

| Prob. | Optimal | SiC−PSO | COPSO | Mezura |
|---|---|---|---|---|
| E01 | 1.724852 | 1.724852 | 1.724852 | 1.724852 |
| E02 | 6,059.714335 | 6,059.714335 | 6,059.714335 | 6,059.7143 |
| E03 | NA | 2,996.348165 | 2,996.372448 | 2,996.348094* |
| E04 | 0.012665 | 0.012665 | 0.012665 | 0.012689 |

*Infeasible solution. NA Not avaliable.

*Table 2.* Means and Standard Deviations for the results obtained

| | Mean | | | St. Dev. | | |
|---|---|---|---|---|---|---|
| Prob. | SiC−PSO | COPSO | Mezura | SiC−PSO | COPSO | Mezura |
| E01 | 2.0574 | 1.7248 | 1.7776 | 0.2154 | 1.2E-05 | 8.8E-02 |
| E02 | 6,092.0498 | 6,071.0133 | 6,379.9380 | 12.1725 | 15.1011 | 210.0000 |
| E03 | 2,996.3482 | 2,996.4085 | 2,996.3480* | 0.0000 | 0.0286 | 0.0000* |
| E04 | 0.0131 | 0.0126 | 0.0131 | 4.1E-04 | 1.2E-06 | 3.9E-04 |

*Infeasible solution.

The three algorithms reached the best known values for E01. For E02, SiC-PSO and COPSO reached the best known, but Mezura reported a value with a precision of only 4 digits after the decimal point, and the exact value reached by them is not reported. For E03, SiC-PSO reached the best value, COPSO reached a value slightly worse than ours, and Mezura reached an infeasible value. SiC-PSO and COPSO reached the best value for E04, although Mezura reported a value that is worse than the best known. In general, COPSO obtained the best mean values, except for E03 for which best mean was found by our algorithm. The lower standard deviation values for E01 and E04 was obtained by COPSO; for E02 and E03, our SiC-PSO found the minimum values.

Tables 3a, 4b, 5c and 6d show the solution vectors of the best solution reached by SiC-PSO as well as the values of the constraints, for each of the problems tested.

*Table 3a.* SiC-PSO Solution vector for E01 (welded beam)

| | Best Solution |
|---|---|
| $x_1$ | 0.205729 |
| $x_2$ | 3.470488 |
| $x_3$ | 9.036624 |
| $x_4$ | 0.205729 |
| $g_1(\vec{x})$ | -1.819E-12 |
| $g_2(\vec{x})$ | -0.003721 |
| $g_3(\vec{x})$ | 0.000000 |
| $g_4(\vec{x})$ | -3.432983 |
| $g_5(\vec{x})$ | -0.080729 |
| $g_6(\vec{x})$ | -0.235540 |
| $g_7(\vec{x})$ | 0.000000 |
| $f(\vec{x})$ | 1.724852 |

*Table 4b.* SiC-PSO Solution vector for E02 (pressure vessel)

| | Best Solution |
|---|---|
| $x_1$ | 0.812500 |
| $x_2$ | 0.437500 |
| $x_3$ | 42.098445 |
| $x_4$ | 176.636595 |
| $g_1(\vec{x})$ | -4.500E-15 |
| $g_2(\vec{x})$ | -0.035880 |
| $g_3(\vec{x})$ | -1.164E-10 |
| $g_4(\vec{x})$ | -63.363404 |
| $f(\vec{x})$ | 6,059.714335 |

*Table 5c.* SiC-PSO Solution vector for E03 (speed reducer)

| | Best Solution |
|---|---|
| $x_1$ | 3.500000 |
| $x_2$ | 0.700000 |
| $x_3$ | 17 |
| $x4$ | 7.300000 |
| $x_5$ | 7.800000 |
| $x_6$ | 3.350214 |
| $x_7$ | 5.286683 |
| $g_1(\vec{x})$ | -0.073915 |
| $g_2(\vec{x})$ | -0.197998 |
| $g_3(\vec{x})$ | -0.499172 |
| $g_4(\vec{x})$ | -0.901471 |
| $g_5(\vec{x})$ | 0.000000 |
| $g_6(\vec{x})$ | -5.000E-16 |
| $g_7(\vec{x})$ | -0.702500 |
| $g_8(\vec{x})$ | -1.000E-16 |
| $g_9(\vec{x})$ | -0.583333 |
| $g_{10}(\vec{x})$ | -0.051325 |
| $g_{11}(\vec{x})$ | -0.010852 |
| $f(\vec{x})$ | 2,996.348165 |

*Table 6d.* SiC-PSO Solution vector for E04 (tension/compression spring)

| | Best Solution |
|---|---|
| $x_1$ | 0.051583 |
| $x_2$ | 0.354190 |
| $x_3$ | 11.438675 |
| $g_1(\vec{x})$ | -2.000E-16 |
| $g_2(\vec{x})$ | -1.000E-16 |
| $g_3(\vec{x})$ | -4.048765 |
| $g_4(\vec{x})$ | -0.729483 |
| $f(\vec{x})$ | 0.012665 |

## 5. Conclusions and Future Work

We have presented a simple PSO algorithm (SiC-PSO) for constrained optimization problems. The proposed approach uses a simple constraint-

handling mechanism, a ring topology for implementing the *lbest* model and a novel formula to update the position of particles. SiC-PSO had a very good performance when is applied to several engineering design optimization problems. We compared our results with respect to those obtained by two algorithms that had been previously found to perform well in the same problems. These two algorithms are more sophisticated than our SiC-PSO. Our algorithm obtained the optimal values for each of the test problems studied, while performing a lower number of objective function evaluations. Also, the performance of our approach with respect to the mean and standard deviation is comparable with that shown by the other algorithms. Thus, we consider our approach to be a viable choice for solving constrained engineering optimization problems, due to its simplicity, speed and reliability. As part of our future work, we are interested in exploring other PSO models and in performing a more detailed statistical analysis of the performance of our proposed approach.

## Acknowledgment

The first and second author acknowledge support from the ANPCyT (National Agency to Promote Science and Technology, PICT 2005 and Universidad Nacional de San Luis. The third author acknowledges support from CONACyT project no. 45683-Y.

## References

[1] S. Akhtar, K. Tai, and T. Ray. A Socio-behavioural Simulation Model for Engineering Design Optimization. *Eng. Optimiz.*, 34(4):341–354, 2002.

[2] J. Arora. *Introduction to Optimum Design*. McGraw-Hill, New York, 1989.

[3] A. Belegundu. *A Study of Mathematical Programming Methods for Structural Optimization*. PhD thesis, Department of Civil Environmental Engineering, University of Iowa, Iowa, 1982.

[4] H. Bernardino, H. Barbosa, and A. Lemonge. A Hybrid Genetic Algorithm for Constrained Optimization Problems in Mechanical Engineering. In *Proc. IEEE Congress on Evolutionary Computation (CEC 2007),* pages 646–653, Singapore, 2007.

[5] L. Cagnina, S. Esquivel, and C. Coello Coello. A Particle Swarm Optimizer for Constrained Numerical Optimization. In *Proc. 9th International Conference on Parallel Problem Solving from Nature (PPSN IX.)*, pages 910–919, Reykjavik, Island, 2006.

[6] L. Cagnina, S. Esquivel, and C. Coello Coello. A Bi-population PSO with a Shake-Mechanism for Solving Constrained Numerical Optimization. In *Proc. IEEE Congress on Evolutionary Computation (CEC 2007)*, pages 670–676, Singapore, 2007.

[7] L. Cagnina, S. Esquivel, and R. Gallard. Particle Swarm Optimization for Sequencing Problems: a Case Study. In *Proc. IEEE Congress on Evolutionary Computation (CEC 2004)*, pages 536–541, Portland, Oregon, USA, 2004.

[8] Y. Cao and Q. Wu. Mechanical Design Optimization by Mixed-variable Evolutionary Programming. In *Proc. IEEE International Conference on Evolutionary Computation*, pages 443–446, 1997.

[9] J. Cha and R. Mayne. Optimization with Discrete Variables via Recursive Quadratic Programming: part II. *Transaction of ASME*, 111:130–136, 1989.

[10] R. Eberhart and Y. Shi. A Modified Particle Swarm Optimizer. In *Proc. International Conference on Evolutionary Computation*, Anchorage, AK, 1998.

[11] J. Fu, R. Fenton, and W. Cleghorn. A Mixed Integer-discrete-continuous Programming Method and its Applications to Engineering Design Optimization. *Eng. Optimiz.*, 17(4):263–280, 1991.

[12] J. Golinski. An Adaptive Optimization System Applied to Machine Synthesis. *Mech. Mach. Theory*, 8(4):419–436, 1973.

[13] C. Guo, J. Hu, B. Ye, and Y. Cao. Swarm Intelligence for Mixed-variable Design Optimization. *J. Zheijiang Univ. Sc.*, 5(7):851–860, 1994.

[14] S. He, E. Prempain, and Q. Wu. An Improved Particle Swarm Optimizer for Mechanical Design Optimization Problems. *Eng. Optimiz.*, 36(5):585–605, 2004.

[15] A. Hernandez Aguirre, A. Muñoz Zavala, E. Villa Diharce, and S. Botello Rionda. COPSO: Constrained Optimization via PSO Algorithm. Technical report No. I-07-04/22-02-2007, Center for Research in Mathematics (CIMAT), 2007.

[16] X. Hu, R. Eberhart, and Y. Shi. Engineering Optimization with Particle Swarm. 2003.

[17] J. Kennedy. Small World and Mega-Minds: Effects of Neighborhood Topologies on Particle Swarm Performance. In *Proc. Congress on Evolutionary Computation*, pages 1931–1938, 1999.

[18] J. Kennedy and R. Eberhart. Bores Bones Particle Swarm. In *Proc. IEEE Swarm Intelligence Symposium*, pages 80–89, 2003.

[19] H. Li and T. Chou. A Global Approach of Nonlinear Mixed Discrete Programming in Design Optimization. *Eng. Optimiz.*, 22:109–122, 1994.

[20] H. Loh and P. Papalambros. A Sequential Linearization Approach for Solving Mixed-discrete Nonlinear Design Optimization Problems. *J. Mech. Design*, 113(3):325–334, 1991.

[21] M. Mahdavi, M. Fesanghary, and E. Damangir. An Improved Harmony Search Algorithm for Solving Optimization Problems. *Applied Mathematics and Computation*, 188(2007):1567–1579, 2007.

[22] E. Mezura and C. Coello. Useful Infeasible Solutions in Engineering Optimization with Evolutionary Algorithms. *Lect. Notes Comp. Sc.*, 3789:652–662, 2005.

[23] K. Ragsdell and D. Phillips. Optimal Design of a Class of Welded Structures using Geometric Programming. *ASME Journal of Engineering for Industries*, 98(3):1021–1025, 1976.

[24] E. Sandgren. Nonlinear Integer and Discrete Programming in Mechanical Design Optimization. *ASME Journal of Mechanical Design*, 112:223–229, 1990.

[25] G. Thierauf and J. Cai. Evolution Strategies-parallelization and Applications in Engineering Optimization. In B.H.V. Topping, editors, *Parallel and Distributed Precessing for Computational Mechanics*, 1997.

[26] S. Wu and T. Chou. Genetic Algorithms for Nonlinear Mixed Discrete-integer Optimization Problems via Meta-genetic Parameter Optimization. *Eng. Optimiz.*, 24:137–159, 1995.

[27] C. Zhang and H. Wang. Mixed-discrete Nonlinear Optimization with Simulated Annealing. *Eng. Optimiz.*, 21:277–291, 1993.

# Appendix: Engineering problems

Formulating of the engineering design problems used to test the algorithm proposed.

## E01: Welded beam design optimization problem

The problem is to design a welded beam for minimum cost, subject to some constraints [23]. Figure A.1 shows the welded beam structure which consists of a beam A and the weld required to hold it to member B. The objective is to find the minimum fabrication cost, considerating four design variables: $x_1$, $x_2$, $x_3$, $x_4$ and constraints of shear stress $\tau$, bending stress in the beam $\sigma$, buckling load on the bar $P_c$, and end deflection on the beam $\delta$. The optimization model is summarized in the next equation:
*Minimize:* $f(\vec{x}) = 1.10471{x_1}^2 x_2 + 0.04811 x_3 x_4 (14.0 + x_2)$
subject to:

$$g_1(\vec{x}) = \tau(\vec{x}) - 13,600 \leq 0$$
$$g_2(\vec{x}) = \sigma(\vec{x}) - 30,000 \leq 0$$
$$g_3(\vec{x}) = x_1 - x_4 \leq 0$$
$$g_4(\vec{x}) = 0.10471({x_1}^2) + 0.04811 x_3 x_4 (14 + x_2) - 5.0 \leq 0$$
$$g_5(\vec{x}) = 0.125 - x_1 \leq 0$$
$$g_6(\vec{x}) = \delta(\vec{x}) - 0.25 \leq 0$$
$$g_7(\vec{x}) = 6,000 - Pc(\vec{x}) \leq 0$$

with:

$$\tau(\vec{x}) = \sqrt{(\tau')^2 + (2\tau'\tau'')\frac{x_2}{2R} + (\tau'')^2}$$
$$\tau' = \frac{6,000}{\sqrt{2} x_1 x_2}$$
$$\tau'' = \frac{MR}{J}$$
$$M = 6,000\left(14 + \frac{x_2}{2}\right)$$
$$R = \sqrt{\frac{{x_2}^2}{4} + \left(\frac{x_1 + x_3}{2}\right)^2}$$
$$J = 2\left\{x_1 x_2 \sqrt{2}\left[\frac{{x_2}^2}{12} + \left(\frac{x_1 + x_3}{2}\right)^2\right]\right\}$$
$$\sigma(\vec{x}) = \frac{504,000}{x_4 {x_3}^2}$$
$$\delta(\vec{x}) = \frac{65,856,000}{(30 \times 10^6) x_4 {x_3}^3}$$
$$Pc(\vec{x}) = \frac{4.013(30 \times 10^6)\sqrt{\frac{{x_3}^2 {x_4}^6}{36}}}{196}\left(1 - \frac{x_3\sqrt{\frac{30\times10^6}{4(12\times10^6)}}}{28}\right)$$

with $0.1 \leq x_1, x_4 \leq 2.0$, and $0.1 \leq x_2, x_3 \leq 10.0$.
Best solution: $x^* = (0.205730, 3.470489, 9.036624, 0.205729)$ where $f(x^*) = 1.724852$.

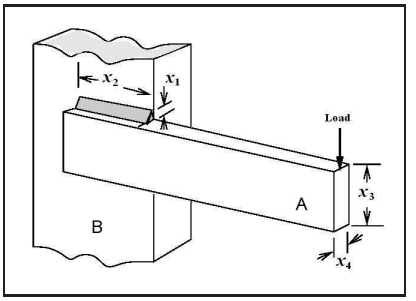



*Figure A.1.* Weldem Beam.

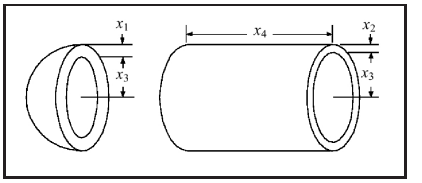



*Figure A.2.* Pressure Vessel.

## E02: Pressure Vessel design optimization problem

A compressed air storage tank with a working pressure of 3,000 psi and a minimum volume of 750 ft$^3$. A cylindrical vessel is capped at both ends by hemispherical heads (see Fig. A.2). Using rolled steel plate, the shell is made in two halves that are joined by teo longitudinal welds to form a cylinder. The objective is minimize the total cost, including the cost of the materials forming the welding [24]. The design variables are: thickness $x_1$, thickness of the head $x_2$, the inner radius $x_3$, and the length of the cylindrical section of the vessel $x_4$. The variables $x_1$ and $x_2$ are discrete values which are integer multiples of 0.0625 inch. Then, the formal statement is:
*Minimize:* $f(\vec{x}) = 0.6224x_1x_3x_4 + 1.7781x_2{x_3}^2 + 3.1661{x_1}^2x_4 + 19.84{x_1}^2x_3$
subject to:

$$g_1(\vec{x}) = -x_1 + 0.0193x_3 \leq 0$$

$$g_2(\vec{x}) = -x_2 + 0.00954x_3 \leq 0$$

$$g_3(\vec{x}) = -\pi{x_3}^2{x_4}^2 - \frac{4}{3}\pi{x_3}^3 + 1,296,000 \leq 0$$

$$g_4(\vec{x}) = x_4 - 240 \leq 0$$

with $1 \times 0.0625 \leq x_1$, $x_2 \leq 99 \times 0.0625$, $10.0 \leq x_3$, and $x_4 \leq 200.0$.
Best solution: $x^* = (0.8125, 0.4375, 42.098446, 176.636596)$ where $f(x^*) = 6,059.714335$.

## E03: Speed Reducer design optimization problem

The design of the speed reducer [12] shown in Fig. A.3, is considered with the face width $x_1$, module of teeth $x_2$, number of teeth on pinion $x_3$, length of the first shaft between bearings $x_4$, length of the second shaft between bearings $x_5$, diameter of the first shaft $x_6$, and diameter of the first shaft $x_7$ (all variables continuous except $x_3$ that is integer). The weight of the speed reducer is to be minimized subject to constraints on bending stress of the gear teeth, surface stress, transverse deflections of the shafts and stresses in the shaft. The problem is:
*Minimize:* $f(\vec{x}) = 0.7854x_1{x_2}^2(3.3333x_3^2 + 14.9334x_3 - 43.0934) - 1.508x_1(x_6^2 + x_7^2) + 7.4777(x_6^3 + x_7^3) + 0.7854(x_4x_6^2 + x_5x_7^2)$
subject to:

$$g_1(\vec{x}) = \frac{27}{x_1x_2^2x_3} - 1 \leq 0$$

$$g_2(\vec{x}) = \frac{397.5}{x_1x_2^2x_3^2} - 1 \leq 0$$

$$g_3(\vec{x}) = \frac{1.93x_4^3}{x_2x_3x_6^4} - 1 \leq 0$$

$$g_4(\vec{x}) = \frac{1.93x_5^3}{x_2x_3x_7^4} - 1 \leq 0$$

$$g_5(\vec{x}) = \frac{1.0}{110x_6^3}\sqrt{\left(\frac{745.0x_4}{x_2x_3}\right)^2 + 16.9 \times 10^6} - 1 \leq 0$$

$$g_6(\vec{x}) = \frac{1.0}{85x_7^3}\sqrt{\left(\frac{745.0x_5}{x_2x_3}\right)^2 + 157.5 \times 10^6} - 1 \leq 0$$

$$g_7(\vec{x}) = \frac{x_2x_3}{40} - 1 \leq 0$$

$$g_8(\vec{x}) = \frac{5x_2}{x_1} - 1 \leq 0$$

$$g_9(\vec{x}) = \frac{x_1}{12x_2} - 1 \leq 0$$

$$g_{10}(\vec{x}) = \frac{1.5x_6 + 1.9}{x_4} - 1 \leq 0$$

$$g_{11}(\vec{x}) = \frac{1.1x_7 + 1.9}{x_5} - 1 \leq 0$$

with $2.6 \leq x_1 \leq 3.6$, $0.7 \leq x_2 \leq 0.8$, $17 \leq x_3 \leq 28$, $7.3 \leq x_4 \leq 8.3$, $7.8 \leq x_5 \leq 8.3$, $2.9 \leq x_6 \leq 3.9$, and $5.0 \leq x_7 \leq 5.5$.
Best solution: $x^* = (3.500000, 0.7, 17, 7.300000, 7.800000, 3.350214, 5.286683)$ where $f(x^*) = 2,996.348165$.

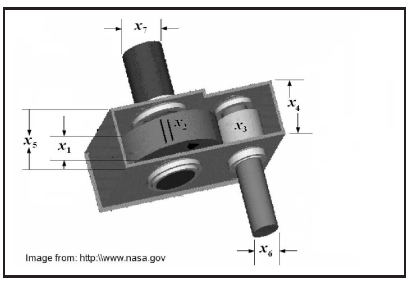



*Figure A.3.* Speed Reducer.

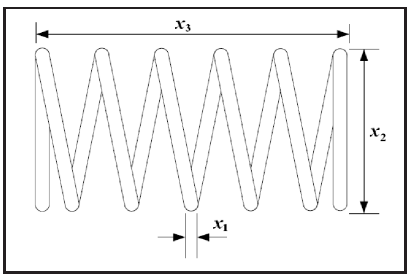



*Figure A.4.* Tension/Compression Spring.

## E04: Tension/compression spring design optimization problem

This problem [2] [3] minimizes the weight of a tension/compression spring (Fig. A.4), subject to constraints of minimum deflection, shear stress, surge frequency, and limits on outside diameter and on design variables. There are three design variables: the wire diameter $x_1$, the mean coil diameter $x_2$, and the number of active coils $x_3$. The mathematical formulation of this problem is:

*Minimize:* $f(\vec{x}) = (x_3 + 2)x_2x_1^2$
subject to:

$$g_1(\vec{x}) = 1 - \frac{x_2^3 x_3}{7,178x_1^4} \leq 0$$

$$g_2(\vec{x}) = \frac{4x_2^2 - x_1x_2}{12,566(x_2x_1^3) - x_1^4} + \frac{1}{5,108x_1^2} - 1 \leq 0$$

$$g_3(\vec{x}) = 1 - \frac{140.45x_1}{x_2^2x_3} \leq 0$$

$$g_4(\vec{x}) = \frac{x_2 + x_1}{1.5} - 1 \leq 0$$

with $0.05 \leq x_1 \leq 2.0$, $0.25 \leq x_2 \leq 1.3$, and $2.0 \leq x_3 \leq 15.0$.
Best solution: $x^* = (0.051690, 0.356750, 11.287126)$ where $f(x^*) = 0.012665$.

# ANALYSIS OF AN IMMUNE ALGORITHM FOR PROTEIN STRUCTURE PREDICTION

Andrew J. Bennett, Roy L. Johnston, Eleanor Turpin
*University of Birmingham*
*Birmingham, United Kingdom*
ajb@tc.bham.ac.uk; r.l.johnston@bham.ac.uk; msc76ext@cs.bham.ac.uk

Jun Q. He
*Aberystwyth University,*
*Aberystwyth, United Kingdom*
jqh@aber.ac.uk

**Abstract** The aim of a protein folding simulation is to determine the native state of a protein from its amino acid sequence. In this paper we describe the development and application of an Immune Algorithm (IA) to find the lowest energy conformations for the 2D (square) HP lattice bead protein model. Here we introduce a modified chain growth constructor to produce the initial population, where intermediate infeasible structures are recorded, thereby reducing the risk of attempting to perform wasteful point mutations during the mutation phase. We also investigate various approaches for population diversity tracking, ultimately allowing a greater understanding of the progress of the optimization.

**Keywords:** HP lattice bead model, Immune algorithm, Population diversity tracking, Protein modelling

## 1. Introduction

Predicting the 3-dimensional secondary and tertiary structure of a protein molecule from its (primary structure) amino acid sequence alone is an important problem in chemical biology [1]. Under certain physiological conditions, the amino acid chain will reliably fold into a specific native state (biologically active conformation). The protein folding problem is the search for this native state for a given sequence of amino acid residues. The reliability of protein folding is said to be dominated by the presence of a "folding funnel" on the folding energy landscape since

systematic or random searching is clearly infeasible for large numbers of amino acids [2]. Therefore, discovering the nature of the folding energy landscape is necessary to develop a better understanding of the folding dynamics [3].

Many protein models have been developed, ranging from simple, minimalist models such as the HP lattice bead model [4], to more complicated and computationally expensive models such as off-lattice interpretations. The most common lattice structures are 2D square and 3D cubic. More computationally intense models include the dynamical lattice and all-atom models, both introducing more complicated fitness functions.

In this work, the standard HP lattice bead model has been incorporated into an immune algorithm. Despite the minimalistic approach employed by this model, it has been shown to belong to the "NP-Hard" set of problems [5]. Monte Carlo [6], chain growth algorithms [4], simulated annealing [7], genetic algorithms [5, 8, 9], ant colony optimization [10] and more recently immune algorithms [11] have been developed by many researchers as heuristic and approximate solutions for this and other computationally hard problems.

## 2. Methodology

### 2.1 The HP Lattice Bead Model

In this work, the standard HP lattice bead model is embedded in a 2-dimensional square lattice, restricting bond angles to only a few discrete values [4]. Interactions are only counted between topological neighbours, that is between beads (representing amino acids) that lie adjacent to each other on the lattice, but which are not sequence neighbours [3]. The energies corresponding to the possible topological interactions are as follows:

$$\epsilon_{HH} = -1.0 \qquad \epsilon_{HP} = 0.0 \qquad \epsilon_{PP} = 0.0 \tag{1}$$

By summing over these local interactions, the energy of the model protein can be obtained:

$$E = \sum_{i<j} \epsilon_{ij} \Delta_{ij} \tag{2}$$

where

$$\Delta_{ij} = \begin{cases} 1 & \text{if i and j are topological neighbours,} \\ & \text{but are not sequence neighbours;} \\ 0 & \text{otherwise.} \end{cases}$$

The HP lattice model recognises only the hydrophobic interaction as the driving force in protein folding, with many native structures pro-

tecting the hydrophobic core with polar residues, resulting in a compact arrangement [11]. This idea reflects the repulsive nature of the interactions between the hydrophobic residues and the surrounding water molecules [3].

## 2.2 The Coordinate System

A previous study has illustrated how a local coordinate system offers better performance than a global one for studying protein folding [2]. In this work, a **local coordinate system** is used to define the folding conformation of the model proteins, that is the position of bead $j$ is defined relative to beads $(j-1)$ and $(j-2)$. As the energy is identical for rotationally related structures, the bond between the first two beads lies along the $x$-axis, with these beads having coordinates (0,0) and (1,0) respectively. As a result, the search spaced is halved. The bond joining the $(j-1)^{th}$ and $j^{th}$ beads can be left, right or straight ahead relative to the bond joining the $(j-2)^{th}$ and $(j-1)^{th}$ bead, corresponding to an integer representation of 0, 1 and 2 respectively. The protein conformation is therefore expressed as a **conformation vector**, containing a list of 0's, 1's and 2's.

For this study, a set of well investigated protein benchmark sequences have been considered: the tortilla HP benchmark sequences [12]. They range in length from eighteen to fifty beads and are listed in table 1. The table also includes the energy, $E^*$, of the putative global minimum (or conformations, since all of these structures have degenerate global minima) for each sequence.

*Table 1.* Benchmark HP sequences used in the present study [12]. The lowest energies that have been found for these sequences are indicated by $E^*$.

| Name | Length | $E^*$ | Sequence |
|---|---|---|---|
| HP-18a | 18 | -9 | $PHP_2HPH_3PH_2PH_5$ |
| HP-18b | 18 | -8 | $HPHPH_3P_3H_4P_2H_2$ |
| HP-18c | 18 | -4 | $H_2P_5H_2P_3HP_3HP$ |
| HP-20a | 20 | -10 | $H_3P_2(HP)_2HP_2(HP)_2HP_2H$ |
| HP-20b | 20 | -9 | $HPHP_2H_2PHP_2HPH_2P_2HPH$ |
| HP-24 | 24 | -9 | $H_2P_2(HP_2)_6H_2$ |
| HP-25 | 25 | -8 | $P_2HP_2(H_2P_4)_3H_2$ |
| HP-36 | 36 | -14 | $P_3H_2P_2H_2P_5H_7P_2H_2P_4H_2P_2HP_2$ |
| HP-48 | 48 | -23 | $P_2H(P_2H_2)_2P_5H_{10}P_6(H_2P_2)_2HP_2H_5$ |
| HP-50 | 50 | -21 | $H_2(PH)_3PH_4P(HP_3)_3P(HP_3)_2HPH_4(PH)_4H$ |

## 3. The Immune Algorithm

An immune algorithm [13] is inspired by the clonal selection principle employed by the human immune system. In this process, when an antigen enters the body, B and T lymphocytes are able to clone upon recognition and bind to it [13]. Many clones are produced in response and undergo many rounds of somatic hypermutation. The higher the fitness of a B cell to the available antigens, the greater the chance of cloning. Cells have a certain life expectancy, allowing a higher specific responsiveness for future antigenic attack [11].

The IA presented here includes the aging, cloning and selection operators used in a previous study by Cutello *et al.* [11], with modified constructor and mutation operators. The constructor employs a backtracking algorithm that records some of the possible mutations by testing bead placement during chain growth. These possibilities are exploited and updated during the mutation process, preventing an infeasible conformation from occurring based on the preceding self-avoiding structure for a particular point in the model protein chain. In retaining this information, infeasible mutations are not explored, allowing a greater number of constructive mutations to be investigated. Figure 1 illustrates the stages involved in placing two consecutive beads during the chain growth phase. Before committing a bead to the lattice, all possible directions are explored, 1(a), and from the valid options available, a random choice is made, 1(b). Again all possible choices are investigated, marking any infeasible options (note that choosing left will not result in a self avoiding conformation), 1(c), and a valid choice is selected from the remaining options, 1(d). Any remaining valid choices are left unmarked for use in the first mutation phase after the initial chain growth. Once a valid mutation has been made, the entire structure is reconstructed as before marking any infeasible directions as a result of the new conformation vector.

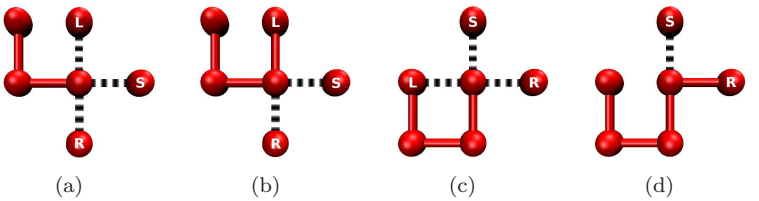



*Figure 1.* Stages of the chain growth algorithm investigating left (L), right (R) and straight (S) availability (a), random selection from all available directions (b), at the next locus investigation of L, R and S availability (c) and random selection from remaining available S and R directions (d).

In the basic IA set up, there are a maximum of 10,000,000 fitness evaluations, with the maximum number of generations set to 500,000. In order to estimate the optimal combination of parameters, we adopted the procedure used by Cutello et al., whereby the maximum B-Cell age and the number of clones were each varied from 1 to 10. Population sizes examined were 10, 25, 50 100 and 200. This provided a combination of 500 different parameter sets for each sequence, which was applied to all the benchmark sequences up to 25 beads in length. All fitness evaluations for the best success rates were collated and graded for overall performance. As a result of this preliminary testing, the results presented below were obtained using a maximum B-Cell age of 4, 3 clones and a population size of 10. All results quoted are averaged over 30 independent runs.

# 4. Results

## 4.1 Algorithm Comparison

With CPU time being hardware dependent, the number of fitness evaluations (together with the percentage success rate) have been used to assess the efficiency of the algorithm, as shown in table 2 for the benchmark sequences.

*Table 2.* Comparison of the percentage success and average number of structure evaluations with and without using memory B-Cells

| Sequence | No Memory B-Cells | | Memory B-Cells | |
|---|---|---|---|---|
| | %Success | No. Evaluations | %Success | No. Evaluations |
| HP-18a | 100 | 89,578 | 100 | 117,251 |
| HP-18b | 100 | 40,167 | 100 | 200,740 |
| HP-18c | 100 | 87,761 | 100 | 72,270 |
| HP-20a | 100 | 26,207 | 100 | 312,405 |
| HP-20b | 100 | 15,221 | 100 | 30,414 |
| HP-24 | 100 | 26,580 | 100 | 49,616 |
| HP-25 | 100 | 79,042 | 100 | 95,123 |
| HP-36 | 63 | 4,867,993 | 90 | 3,082,014 |
| HP-48 | 3 | 6,318,721 | 3 | 4,195,086 |
| HP-50 | 50 | 4,904,031 | 96 | 853,706 |

It is apparent from table 2 that, although the use of memory B-Cells [11] hinders the discovery of global minima for some of the smaller sequences, it enhances the search for the larger, more difficult to find sequences. The memory ability allows mid to high fitness conforma-

tions to remain in the population for a longer number of generations. For larger sequences, this allows a more detailed exploration in certain areas of the potential energy surface, permitting the memory B-Cells to converge towards the global solution much sooner. In contrast, for smaller sequences the mid to high fitness range is much smaller, thereby preventing a rapid exploration of the potential energy surface by retaining unfavourable segments of local structure for a larger number of generations. Generally, the use of memory B-Cells allows a more diverse inspection of the potential energy surface, due to a greater number of the degenerate conformations being found. This is achieved as favourable fragments of local structure are not rapidly disposed of during the retirement process, hindering efficiency as a consequence.

The algorithm presented here shows promising results, being comparable to the work of Cutello *et al.* [11]. While our success rates for the larger sequences (e.g. HP-48) are a little lower, in some cases our number of fitness evaluations show an improvement.

## 4.2 Analysis of Global Minima

The compact structural arrangement present in all global minima (GMs) is apparent from the example GMs shown in figure 2. With the driving force being the hydrophobic topological contact, it can be seen that compact hydrophobic cores give rise to high fitness conformations. Inspection of the HP-48 global minimum (i) allows us to understand the poor success rate for this sequence. The 5×5 hydrophobic core presents

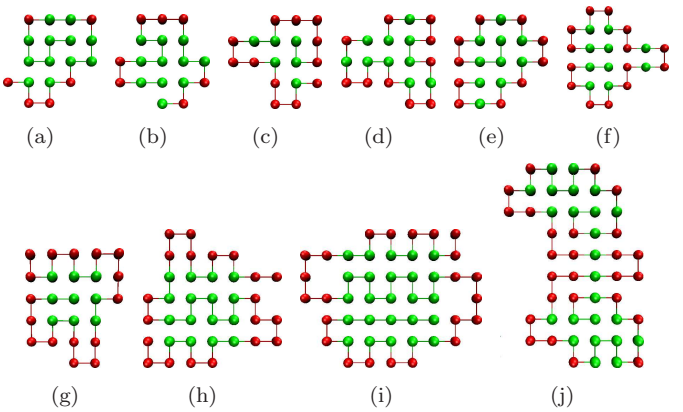
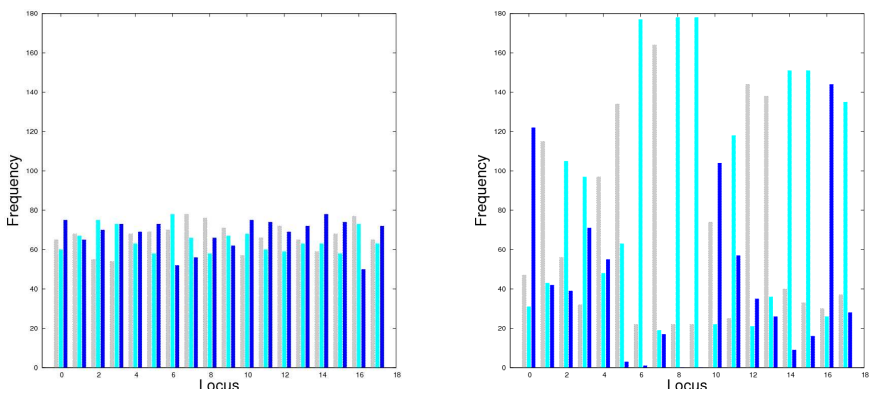



*Figure 2.* Examples of GM structures for the benchmark sequences: (a) HP-18a (b) HP-18b (c) HP-18c (d) HP-20a (e) HP-20b (f) HP-24 (g) HP-25 (h) HP-36 (i) HP-48 (j) HP-50.

a problem to the IA (or other optimization algorithms [3]) in achieving convergence, as a single misplaced hydrophobic bead will result in only a metastable conformation. The problem does not exist for the HP-50 sequence (j), due to the presence of two small hydrophobic cores coupled by a chain of hydrophobic beads, which explains the higher success rate and fewer average structure evaluations necessary for HP-50, compared with HP-48 and (when using memory B-cells) even the much shorter HP-36 sequence [3]. The work of Cutello *et al.* supports this idea [11], as similar magnitudes of the number of fitness evaluations for these problematic sequences can be seen, with a much lower success rate for HP-48 than for any other instance.

## 4.3 Tracking Population Diversity

The much larger populations required to ensure population diversity can be problematic for both GAs and IAs. In this section, a single run, with population size 200 for sequence HP-20a has been analyzed. The global minimum was found in generation 28, at which point the algorithm was terminated due to meeting the search criteria. In order to help us understand the progress of the optimization and ultimately to improve the methodology, monitoring population diversity and the progress of the algorithm is beneficial.

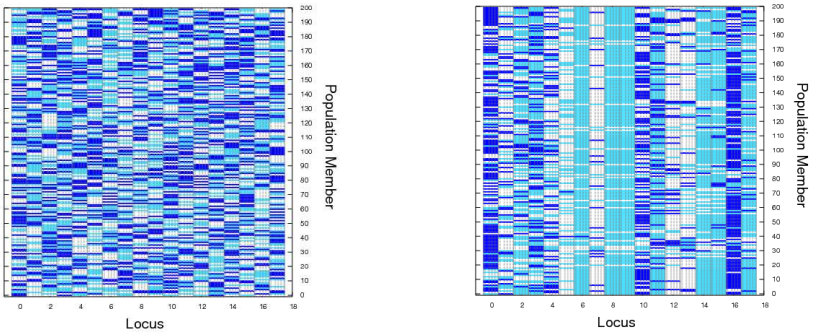



(a) (b)

*Figure 3.* The frequency of alleles at each locus along the model protein chain for the initial population (a) and the final population (b), left (grey), right (light blue) and straight ahead (dark blue).

Figure 3(a) assigns a colour to each of the three possible direction decisions (corresponding to alleles in a genetic sense) made when placing

each successive bead. It can be seen that initial structure generation, using the IA's constructor, is indeed statistically uniform, showing the frequency of left (grey), right (light blue) and straight ahead (dark blue) choices at each locus of the model protein chain to be very similar. In contrast, figure 3(b) illustrates how this statistical distribution is skewed in the final population (generation 28), in that the IA has concentrated its search to a much narrower region of the potential energy surface. It should also be noted that position 6 in the chain has a very low frequency of the straight ahead choice (dark blue), because (for most population members) previous direction decisions preclude (for structural and/or energetic reasons) this choice from being made at this chain position.



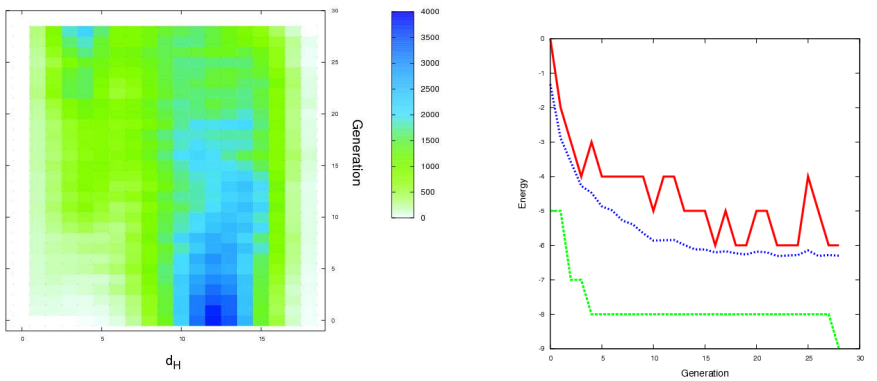

(a) (b)

*Figure 4.* Graphical representation of an initial population (a) and final population (b) of B-Cells, left (grey), right (light blue) and straight ahead (dark blue). Population members are sorted by descending fitness, with structures of the highest energy at the bottom of the plot.

Figure 4 shows a graphical representation of the initial and final populations of the calculation. By plotting the conformation vector for each population member, the population can be quickly compared for diversity. Population members are ordered by descending fitness and the colour scheme is similar to the allele frequency distribution shown in figure 3, but with white replacing grey for the left choice. It is clear that initially the population has high diversity (in agreement with the allele frequency plot shown above), with the algorithm preserving favourable regions of local structure (corresponding to schemata in a GA sense) as the calculation converges. More detailed analysis of the final population shows that there are often correlations (or anti-correlations) be-

tween directions at specific loci, with certain combinations giving rise to favourable energies or infeasible structures, repectively.

For simple protein models such as the HP lattice bead model, the Hamming distance ($d_H$, which is the number of bit differences between two conformation vectors) can be used as a simple measure of similarity between structures in the population. Figure 5(a) plots the frequency of the Hamming distances between all pairs of structures in the population as a function of generation. (As the population size is 200, there are a total of 19,900 pair Hamming distances). It can be seen how the diversity of the population changes as the calculation approaches the global minimum (which is found in generation 28). Combining this with a plot of the best, worst and average fitnesses in the population, as a function of generation (figure 5(b)), it should be noted that structural diversity shows a more uniform spread (beginning around generation 20) as favourable segments of local structure begin to dominate the population, with the search focussing on a much more concentrated area of the potential energy surface. It is also evident that the population diversity drastically decreases during the final stages of the calculation, not just in the final generation. This confirms that the calculation has not discovered the global minimum by chance, but a directed search strategy has been employed.




(a) (b)

*Figure 5.* (a) The density of pairwise Hamming distances, $d_H$, between population members throughout the calculation. (b) The change in energy throughout the calculation, showing the best (green), worst (red) and average (blue) energies in each generation.

## 5. Conclusions

Although implementation of a modified constructor for use in the mutation phase of the IA has not always given greater success rates (especially for more challenging sequences), it has allowed for a more efficient search to be performed in some cases, showing a descrease in the number of fitness evaluation performed. The use of population diversity tracking allows a greater understanding of the algorithm's ability to explore areas of the potential energy surface of these simple model proteins. Areas of favourable local structure along the chain can be assessed, illustrating the important allele combinations that give rise to the determination of global minima. We are currently applying these approaches to more realistic protein models.

## Acknowledgements

AJB thanks Dr Benjamin Curley for programming assistance and the University of Birmingham for PhD funding. Calculations were performed on the University of Birmingham's BlueBEAR 1500+ processor cluster, funded by the EPSRC (under SRIF3) and the University of Birmingham.

## References

[1] K.M. Merz, (editor). *The Protein Folding problem and Structure Prediction.* Birkhauser, Boston, MA, 1994.

[2] N. Krasnogor, W.E. Hart, J. Smith, and D. Pelta. Protein Structure Prediction with Evolutionary Algorithms. In *Proc. 1999 International Genetic and Evolutionary Computation Conference (GECCO99)*, pages 1569–1601, Orlando, Florida, USA, 1999.

[3] G.A. Cox and R.L. Johnston. Analyzing Energy Landscapes for Folding Model Proteins. *J. Chem. Phys.*, 124(20):204714, 2006.

[4] T. Beutler and K. Dill. A Fast Conformational Search Strategy for Finding Low Energy Structures of Model Proteins. *Protein Sci.* 5(10):2037–2043, 1996.

[5] R. Unger and J. Moult. Genetic Algorithms for Protein Folding Simulations. *J. Mol. Biol.*, 231:75–81, 1993.

[6] R. Ramakrishnan, B. Ramachandran, and J.F. Pekny. A Dynamic Monte Carlo Algorithm for Exploration of Dense Conformational Spaces in Heteropolymers. *J. Chem. Phys.*, 106(6):2418–2425, 1997.

[7] S. Kirkpatrick and C.D. Gellat. Optimization by Simulated Annealing. *Science*, 220:671–680, 1983.

[8] W. Liu and B. Schmidt. Mapping of Genetic Algorithms for Protein Folding onto Computational Grids. *TENCON 2005 IEEE Region 10*, 1–6, 2005.

[9] J. Song, J. Cheng, T. Zeng, and J. Mau. A Novel Genetic Algorithm for HP Model Protein Folding. *PDCAT 2005* 935–937, 2005.

[10] A. Shmygelska and H.H. Hoos. An Improved Ant Colony Optimization Algorithm for the 2D HP Protein Folding Problem. *Lect. Notes Comput. Sci.*, 2671:400–412, 2003.

[11] V. Cutello, G. Niscosia, M. Pavone, and J. Timmis. An Immune Algorithm for Protein Structure Prediction on Lattice Models. *IEEE T. Evol. Comput.*, 11(1):101–117, 2007.

[12] W.E. Hart, S. Istrail. HP Benchmarks. `www.cs.sandia.gov/tech_reports/compbio/tortilla-hp-benchmarks.html`.

[13] L.N. de Castro and J. Timmis. *Artifical Immune Systems and Their Applications*. Springer-Verlag, Berlin, Germany, 1999.

# EVOLUTIONARY JAZZ HARMONY: A NEW JAZZ HARMONY SYSTEM

Kjell Bäckman
*IT University*
*Gothenburg, Sweden*
kjell.backman@hv.se

**Abstract** Jazz harmony has during the jazz history mainly been functionally based on tonality principles derived from classical and romantic periods of the 18th and 19th centuries. By means of computer based evolutionary principles we introduce a function-less harmony system that somewhat changes the musical feeling in jazz compositions to more imitate the harmonic feeling of early experiments made by jazz musicians like Cecil Taylor, Herbie Hancock, Brecker Brothers etc. The new harmony system is not a ready-made product but offers a means to explore new harmonic areas. The main features of this new harmony system are chords not built on any specific base note, and chords that should be regarded as harmonic spectrums.

**Keywords:** Chord, Coloured chord, Harmonic spectrum, Evolutionary algorithm, MIDI note, Scale

## 1. Introduction

Jazz harmony has since the birth of jazz been functionally based, which means that each chord has been related to a base note and classified as minor or major, and optionally also enriched with colourization, such as:

Cm, Eb7, G13b9, A7#11

This situation has prevailed through jazz history with a few exceptions. The earliest experiments with other kinds of harmony were made in the 1950's by advanced and new-thinking musicians like Ornette Coleman, Cecil Taylor, Don Cherry and others. Experiments have also been made during the 60's and 70's by e.g. Herbie Hancock, Miles Davies and fusion musicians Brecker Brothers. Not to mention all experiments in the classical music domain during the entire 20th century from Schoenberg and onwards.

However, from the last quarter of the 20th century a stagnation of the harmonic development in jazz has ensued, and nothing harmonically essential has occurred. The Evolutionary Jazz Harmony project is an attempt to break the ice and open new dimensions to harmonic thinking.

The Evolutionary Jazz Harmony project uses a non-functional harmony philosophy (no specific base note and not necessarily connected to the major/minor concept), where the "chords", or rather harmonic spectrums, are built up by means of the computer using evolutionary principles.

The produced chord progressions are used by the automatic jazz composer function described in another paper to produce tunes, and by the generic jazz improvisation program to produce jazz solos based on this kind of harmony. These papers are still under work in progress and have not yet been published. However, there are some publications written by the author that provide valuable background information for this project [1, 2, 3].

Dahlstedt [4], Dean [6], Levinde [7], Manning [8], Rowe [9, 10] and Thywissen [12] have made valuable contributions in the same area and have been sources of inspiration for this project.

## 2. Background

Jazz harmony has since the birth of jazz during the first two decades of the 20th century been systematically organized around a tonal centre by fifth progressions. Blues and ragtime harmony mainly used simple major/minor triads at the distance of fifths. Swing music enriched the chords with sixths and ninths but the chord progressions were mainly the same. Bebop further enriched the chords with further colourizations such as b9, #9, #11, 13, b13 etc. and exchanged some chord progressions by inserting an extra subdominant parallel, e.g.

G7 - C was replaced by Dm7 - G7 - C

However the focus was still on major/minor and fifth progressions. The main harmonic contribution from cool jazz and hardbop during the 50's was further advanced chord colourizations. A few new-thinking musicians began at the end of the 50's to split up the harmonic foundation prevailing until then, and this development continued during the subsequent decades under stylistic classification into "modal jazz", "avant-garde", "free form" etc. Current jazz musicians have to some extent adopted this break-up tendency.

However, mostly you can in the "modern" jazz styles trace some remainders of the functional harmony principles and the fifth circle basis. When progressive or avant-garde musicians create compositions with

new harmony, there still is a risk to get stuck in conventions dictated by routine and learnt behaviour, idiomatic properties of the instrument and the musician's physical and muscular restrictions. The computer has no such restrictions but creates harmonies controlled by the algorithms having been programmed. The aim is to be able to free oneself from traditional thinking and create a completely new kind of harmony.

## 3. Artificial Evolution

The evolutionary algorithm process starts by, from a basic set of parameters (genome), creating a first random population of pictures, melodies, chord progressions or whatever. The fitness evaluation then takes place by examining the bred material (children) and selecting the best, optionally by giving each a score. The children with the highest score have the highest probability to become parents for the next generation. The breeding is done by combining the genome (parameter values) of two parents, optionally by applying a mutation somewhere in the genome. The mutation might imply a shift between two parameter values, or a slight modification of a parameter value.

The principle of using evolutionary algorithms to develop new artistic productions, enhance artistic thinking and stimulate creativity, first started on a broader scale in the digital graphics area, such as forerunner Karl Sims [11]. The evolutionary algorithms principle is well accommodated to that area because when using interactive evaluation of a created generation, as described by Dawkins [5], you can swiftly scan over a great number of pictures and select the best according to your personal preference. With audio material, however, the selection procedure is much slower since you will have to listen through each bred melody in a generation, one at a time. The first experiments in the music area have been made by Collin Johnson and Palle Dahlstedt [4].

The fitness selection and breeding is repeated generation by generation until you arrive at a genome good enough to be used for reproduction of artworks (pictures / melodies etc.).

This process is much the same as the genetic process of creating a new species generation in nature, only that it must be sped up considerably to have a chance to be completed in proper time. The number of generations used for one evolution session must be limited, the calculation of parameter values must be optimized and efficient to allow for a rapid development towards a good genome, and the fitness function must be user friendly to minimize tedious manual intervention. An automatic fitness function would be valuable, but requires programmatic coding of abstract items as tension, climax, phrasing, musicality etc. Such a

function has been developed for jazz improvisation solos by the author [3].

The genome in this project consists of parameter values specifying the internal structure of each chord and the progress from one chord to the next. For each new generation one parent chord progression is combined with another by selecting various portions of each of the parents' genomes. For each child different sections of the parents' genomes are selected, optionally also by performing a mutation which might consist of a slight modification of some genome parameter values.

At each step it is possible to save a parent/child genome, making it possible to pick up a previous situation or reuse an existing genome.

## 4. Method

There are a couple of parameters controlling the overall behaviour of the genetic evolution process:

- Number of notes per chord (4-5)
- Number of notes to change from chord to chord (1-5). A higher value gives abrupt chord changes, while a lower value gives a more homogenous chord sequence.
- Changing method, move notes in steps of half-notes, whole-notes or bigger intervals (random change with maximum 4 half-note steps). Also in this case, bigger intervals give more abrupt chord changes.

These parameter values can be manually set prior to starting an evaluation session. We have experimented with different settings, where the following seems to produce the best result: 4 notes per chord, 2 notes changed per chord, changing method = bigger intervals.

A genome consists of a number (given by "Number of notes per chord") of absolute MIDI pitches as the initial chord. The pitches are randomly created within a specific pitch range around middle C. For each chord change the genome holds the number of half-note steps per note (Fig. 1).

In this case the genome will be:

59 60 63 68 -1 0 +1 0 -1 0 -2 0 ...

When breeding two parents we combine different sections of the parents' genomes just like the process of combining DNA for species.

At the end of the breeding a mutation is made by amending a few values one step up or down, so -1 might be -2 or 0, etc.

The program code is written in C++, including the MIDI compiler function, which makes it possible to use any media player to listen to the

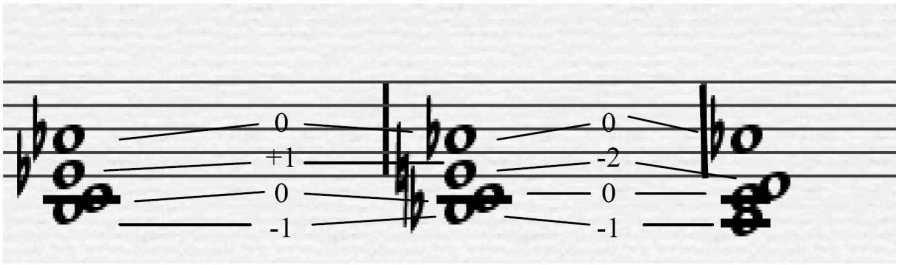



*Figure 1.* Chord genome.

produced MIDI files, and also import them into a note editing program, such as Sibelius. The resulting chord progression is also stored in an ASCII file in a format possible to copy to the project folder for jazz improvisation solos [1, 2, 3, 6, 7, 9].

The chord evaluation in this project was manually made. An automatic fitness function for chord evaluation has not been developed, only for jazz improvisation solos. An evaluation session could last about one hour. It is probably not meaningful to extend a session any longer due to the limitation of one's brain concentration.

## 5. Results

Chord progressions created this way provide the feeling of a continuous progress towards new heights without arriving at rest points, which is the case with traditional functional harmony, where some chords have a striving character to dissolve into tonics. Compare the chord sequence of a tune like 'Autumn Leaves':

Am7b5 D7b9 Gm7 G7b9 Cm7 F7b9 Bbmaj7

There is an intermediate rest point at the chord Gm7 and then a final rest point at Bbmaj7. These rest points provide a relaxation at various positions of the tune, which gives a periodic character. Such relaxation points are not found in tunes with the new kind of harmony. Whether this depends on what people have been used to since long time ago, or real built-in features of the functional harmony, is another topic not further discussed here. Anyway, our conclusion is that this new kind of harmony has an on-going forward-striving feature not prevalent in standard jazz harmony.

When jamming with a jazz group on tunes with this new harmony system, it has the effect on the soloist to continuously proceed towards a climax never completely reached. The soloist is compelled to go on and on and on. The listener will be involved in this forward-striving feeling

of wanting some more all the time. Whether this is good or bad I am not sure, but anyhow it is an interesting feature that some people might find valuable.

When I experimented with these ideas in a live jazz group, it turned out that the musicians had apparent difficulties in keeping chords and scales in their minds during their solos, since they had to learn completely new chords and scales. The harmony was of a kind that they could not apply their current knowledge and routine and not trust old learnt patterns and behaviour. Clever and experienced musicians appeared to be relative beginners, at least during the first rehearsals. Especially when playing tunes with odd periodicity where a chord could last for 3 bars and the next chord for 2 ? bar, etc., difficulties became obvious. So the time required for rehearsal tended to grow remarkably. For example the bassist, who normally bases his walking bass paths on a base note accentuated at the first beat of each bar and chord notes at the remaining beats, got into problems when there was no specific base note. Learning to play this new kind of music is a laboursome task that requires a new way of thinking and a lot of practicing patience.

Also, to find the most adequate way of playing, a lot of time of discussion and reflection has been used in the acoustic live jazz group. For instance, a great deal of cooperative work has been spent by accommodating the bassist's notes and the piano chord layout to each other.

Provided with this paper are a couple of sound examples with chord progressions:

`http://oden.ei.hv.se/kjell/comp/chords1.mid`
`http://oden.ei.hv.se/kjell/comp/chords2.mid`

A couple of pre-composed tunes based on the new harmony system have been prepared with a jazz combo orchestration, where the compositions and improvisations have been created by the referenced algorithms, and where I also play a couple of acoustic piano solos. Also these are provided with this paper:

`http://oden.ei.hv.se/kjell/comp/GeneticSamba.mp3`
`http://oden.ei.hv.se/kjell/comp/gate.mp3`
`http://oden.ei.hv.se/kjell/comp/chaos.mp3`
`http://oden.ei.hv.se/kjell/comp/MissPC.mp3`
`http://oden.ei.hv.se/kjell/comp/PureCode.mp3`

Finally, a couple of sound examples are given from the first rehearsals with the live jazz group:

`http://oden.ei.hv.se/kjell/live2/Chaos2.mp3`
`http://oden.ei.hv.se/kjell/live2/MissPC.mp3`

## 6. Conclusions

Do evolutionary algorithms provide any valuable artistic material? At least some sounding examples are of interest, maybe not of high professional musician class, but provide interesting and unexpected artistic output. A jazz tune composer often uses standard chord progressions learnt during a long time of practicing and concerting. He relies on routines built up through repeated usage of similar chord colourizations.

The new harmonic system presented in this paper provides a tool for creating a new kind of harmonic base by means of evolutionary algorithms. The resulting harmonic schemes can be used as a foundation for new jazz tunes and as a foundation for exploring the world of jazz improvisation.

The main purpose of using computer based support to produce jazz music is that it opens your mind to new ways of thinking and frees you from old habitual paces of reflection. Hopefully it can enrich your harmonic and improvisation style with new kinds of musical material.

However, introducing a new way of thinking revolutionizes the musical habits of experienced musicians. Such a new system requires considerable time of reflection and rehearsal, which has been proved by experience and discussions in the live jazz group.

## References

[1] K. Bäckman. A Generative Representation for the Evolution of Jazz Solos. In *Proc. EvoWorkshop Conference*, Napoli, Italy, 2008.

[2] K. Bäckman Evolutionary Jazz Improvisation. Master thesis, IT University of Gothenburg, Sweden, 2007. `http://oden.ei.hv.se/kjell/eji/thesis.htm`

[3] K. Bäckman. Automatic Fitness in Evolutionary Jazz Improvisation, In *Proc. International Computer Music Conference*, Belfast, UK, 2008.

[4] P. Dahlstedt. Sounds Unheard of - Evolutionary algorithms as creative tools for the contemporary composer. In *Proc. International Computer Music Conference*, La Habana, Cuba, 2001.

[5] R. Dawkins. *The Blind Watchmaker*. New York W. W. Norton & Company, Inc. New York, USA, 1986.

[6] T. Dean. *Hyperimprovisation: Computer-Interactive Sound Improvisation*. A-R Editions Inc., Middleton, Wisconsin, 2003.

[7] M. Levine. The Jazz Theory Book. Sher Music Co. Petaluma, CA, USA, 1995.

[8] P. Manning. *Electronic and Computer Music*. Oxford University Press, New York, USA, 2004.

[9] R. Rowe. *Interactive Music Systems*. The MIT Press, Cambridge, Massachusetts, USA, 1993.

[10] R. Rowe. *Machine Musicianship*. The MIT Press, Cambridge, Massachusetts, USA, 2001.

[11] K. Sims. Artificial Evolution for Computer Graphics. In Proc. ACM SIGGRAPH '91 Conference, Las Vegas, Nevada, July 1991.

[12] K. Thywissen. GeNotator: An environment for investigating the application of generic algorithms in computer assisted composition. In *Proc. International Computer Music Conference*, pages 274–277, Hong Kong, 1996.

# CONSTRAINED TRANSPORTATION SCHEDULING

Gregor Papa, Peter Korošec
*Computer Systems Department, Jožef Stefan Institute*
*Ljubljana, Slovenia*
{gregor.papa; peter.korosec}@ijs.si

**Abstract** This paper presents a comparison of two metaheuristic approaches in solving a constrained transportation scheduling problem. We compared Parameter-less Evolutionary Search and Ant Stigmergy Algorithm in solving the schedule optimization problem. The transportation scheduling problem is NP-hard, and is applicable to different real-life situations with high frequency of loading and unloading operations; like in depots, warehouses and ports.

**Keywords:** Ant-colony optimization, Evolutionary algorithm, Scheduling, Transportation

## 1. Introduction

Transportation is a main component of supply chain and distribution between depots. Assigning and scheduling vehicle routing is a crucial management problem. Despite numerous publications dealing with efficient scheduling methods for vehicle routing, very few address the transportation and load scheduling with multiple constraints on different load type, different vehicle capacities, loading/unloading capacities, and overall transportation time minimization.

The vehicle routing problem (VRP) is a combinatorial optimization and nonlinear programming problem seeking to service a number of customers with a fleet of vehicles. VRP is an important problem in the fields of transportation, distribution and logistics [1]. Often, the goal is delivering goods located at a central depot to customers who have placed orders. In our case we did not deal with routing of vehicles but with optimal schedules for loading and unloading of goods, at the depots.

To solve our transportation scheduling problem we used two metaheuristic techniques to solve this NP-hard optimization problem. The

first one was Parameter-Less Evolutionary Search (PLES) [6], and the second one was Ant Stigmergy Algorithm (ASA) [4]. PLES was already used in the process of the search for an optimal value of CEC2006 functions, and in the process of geometry optimization of an electrical motor stator and rotor. ASA was used in the process of geometry optimization of an electrical motor stator and rotor, geometry optimization of radial impeller, and geometry optimization of electrical motor case [5, 7].

The rest of the paper is organized as follows: in Section 2 we formally define the problem; in Section 3 we describe our approaches used for the search of the best schedule; in Section 4 we describe the experimentation results; and in Section 5 we list our conclusions.

## 2. Definitions

Our problem includes $v$ vehicles. For each vehicle we know the capacity for three types of the load, $w_{i1}, w_{i2}, w_{i3}$, where $i = 1, \ldots, v$. The capacity $w_{ij}$, where $j = 1, 2, 3$, is measured in a load-relevant unit, i.e., ton, cubic meter, number of containers, etc. If the vehicle $i$ is not capable of loading type $j$ of a load, then its capacity is $w_{ij} = 0$.

We assume that each vehicle can load one type of a load only, however in different runs the same vehicle can load different loads. We also assume that the vehicle is always fully loaded; unless the remaining load is smaller than the capacity of the vehicle.

For each vehicle the maximal velocity is known (for loaded and empty vehicle). With the distance between depot $A$ and $B$ we can calculate the driving times $t_{\mathrm{l}1}, \ldots, t_{\mathrm{lv}}$ from $A$ to $B$ for loaded vehicles, and $t_{\mathrm{e}1}, \ldots, t_{\mathrm{ev}}$ from $B$ to $A$ for empty vehicles.

At depot $A$ there are $m$ loading pads, each capable to load $a_{z1}, a_{z2}, a_{z3}$ of load per hour, $z = 1, \ldots, m$. If the loading pad is not appropriate for type $j$ of the load, then $a_{zj} = 0$. Similarly, at depot $B$ there are $n$ unloading pads, capable to unload $b_{u1}, b_{u2}, b_{u3}$ of load per hour, $u = 1, \ldots, n$. There can be only one vehicle at a time on each loading or unloading pad.

### 2.1 Experiment

The aim is to transport three types of load, $V_1, V_2$ and $V_3$, from depot $A$ to $B$. Since the load can not be transported in one run, vehicles have to come back from $B$ to $A$ and perform several runs (cycles). Each run starts in $A$ and consists of:

- waiting in queue till assigned loading pad is available,
- loading at loading pad,

- driving from $A$ to $B$,
- waiting in queue till assigned unloading pad is available,
- unloading at unloading pad,
- driving from $B$ to $A$.

The scheduling task consists of assigning (in each run) a type of a load for each vehicle, a loading pad and unloading pad. The aim is to find a schedule of load types and loading/unloading pads for all vehicles to finish the transportation of all loads in shortest possible time.

## 3. Algorithms

### 3.1 PLES

The main advantage of Parameter-Less Evolutionary Search [6] over the basic GA [3] is that PLES can set the control parameters, like population size, number of generations, probabilities of crossover and mutation, by itself, during the optimization process. The values of parameters depend on the statistical behavior and convergence of the solutions. Elitism, selection, crossover and mutation are implemented through forcing of better individuals and moving of individuals. The control parameters are never set in advance and are not constant.

**Problem encoding.** For $v$ vehicles the chromosome looks like the string of $7v$ values, in general. Here, one value encodes the load type, and the next three pairs of values encode loading and unloading pad for each type of a load. If a vehicle is not capable to transport some type of a load, then the corresponding pair of loading/unloading pad values is omitted in the chromosome. So, for $r$ runs with $v$ vehicles we have a chromosome with $r \sum_{i=1}^{v} (1 + 2l_i)$ positions, where $l_i$ is the number of possible loads for vehicle $i$.

**Optimization.** The population size is proportional to chromosome size, i.e., problem complexity. The number of generations depends on the convergence of the global best solution. Optimization is stopped when the global best solution is not improved for some number of generations ($Limit$). The number depends on ratio of population size ($PopSize$) and the number of generations since the last improvement ($Resting$) of the global best solution.

$$Limit = 5log_{10}(PopSize) + log_{10}(Resting + 1) \tag{1}$$

In every generation the worse solutions are replaced with slightly permutated better solutions. Mutation is realized by moving of some positions in the chromosome according to statistical properties. The number of the positions in the chromosome to be moved depends on the standard deviation of the solution fitness of the previous generation, and the maximal standard deviation of all generations. New value of the parameter is calculated upon the value of each parameter of the current solution, the value of parameter of the globally best solution, and the average value of the parameter in the previous generation.

**Algorithm 1** PLES

1: Set the initial population $S$.
2: Evaluate each individual of initial population.
3: Perform statistical calculations for initial population.
4: **while** stopping criterion, based on statistics, not met **do**
5: Force better individuals to replace worse.
6: Move each individual in the population.
7: Evaluate each individual in the population.
8: Statistical calculations over the population.
9: **end while**

**Statistical calculations.** Each population is statistically evaluated; the best, the worst, and average fitness value in the generation is found. Furthermore, the standard deviation of fitness values of all solutions in the generation, maximal standard deviation of fitness value over all generations, and average value of each parameter in the solution is calculated.

## 3.2 ASA

The Ant Stigmergy Algorithm [4] is an approach to solving multiparameter optimization problems. It is based on stigmergy, a type of collective work that can be observed in ant colonies. ASA consists of two phases:

**Search graph construction.** The problem parameters are transformed into a search graph where vertices represent all possible values of parameters. A vertex representing a parameter value is connected to all vertices representing the values of the next parameter. In our case, each vehicle run is represented with 3 parameters. First parameter represents types of load, second represents the possible loading pads and

third represents the possible unloading pads. Due to this way of presentation, we have to search for subset of possible loading and unloading pads according to the possible types of load for a vehicle. For example, if vehicle could only carry loads 1 and 3, than we need to select those loading and unloading pads that could manage both loads. Of course this means that quite some number of possible solutions are omitted. Otherwise the algorithm is not capable of finding the feasible solution in reasonable time. So, for $r$ runs with $v$ vehicles we need a search graph with $3rv$ parameters. In this way, the constrained multi-parameter optimization problem is transformed into a problem of finding the cheapest path.

**Algorithm 2** ASA

```
1: Construct the search graph from all parameters.
2: Initialize vertices with initial amount of pheromone.
3: while stopping criterion not met do
4:     for all ants do
5:         Find (using probability rule) the cheapest path.
6:     end for
7:     Update pheromone in all vertices visited by the ants.
8:     Additionally increase the pheromone on currently best path.
9:     Evaporate pheromone in all vertices.
10: end while
```

**Optimization.** Here the algorithm applies the optimization procedure based on ant colony optimization [2]. All ants simultaneously start from the starting vertex. The probability of choosing the next vertex depends on the amount of pheromone in the vertices. Ants repeat this action until they reach the ending vertex. The parameter values gathered on each ant's path represent a candidate solution which is then evaluated according to the given objective function. Afterwards, each ant returns to the starting vertex, on its way depositing pheromone in the vertices according to the evaluation result: the better the result, the more pheromone deposited. If the gathered parameter values form an infeasible solution, the amount of pheromone in the parameter vertices is slightly decreased. When the ants return to the starting vertex, two additional actions are performed. First, like in ant colony optimization, a "daemon action" is applied as a kind of elitism, i.e., the pheromone amount on the currently best path is additionally increased. Second, the pheromone in all vertices evaporates, i.e., in each vertex the amount of pheromone is decreased by some predetermined percentage.

## 4. Experiments

PLES and ASA algorithms were used to find the shortest time for transportation of a loads from depot $A$ to $B$. For this experiment we had three types of a load. We had 20 vehicles with different speeds and different load capacities. Not all vehicles were able to transport every type of load. Some vehicles also had self-loaders, therefore they did not need to wait for a free loading/unloading pad.

We determined the upper bound of the number of runs for all vehicles. This upper bound was used to narrow the search space and decrease the search time. The upper bound was set according to the capacity of all loads and capacities of all vehicles for each load. It is calculated as

$$r = 1.5 \left( \frac{V_1}{\sum_{i=1}^{v} w_{i1}} + \frac{V_2}{\sum_{i=1}^{v} w_{i2}} + \frac{V_3}{\sum_{i=1}^{v} w_{i3}} \right) \tag{2}$$

PLES is parameter-less algorithm, therefore no need to set any control parameters. Stopping criteria was automatically set according to the convergence progress and the population size, i.e., approximately 30.000 evaluations for our experiment, or when maximal number of evaluations is reached, i.e., 500.000 evaluations. The ASA parameters were set as follows: the number of ants was 50, $\rho = 0.0005$ and stopping criteria was determined by maximal number of evaluations or 50.000 evaluations without improvement.

Each algorithm was repeated 30 times. Table 1 presents the best, worst, average, and standard deviation value of the transportation time (measured in hours). The last row presents the average number of evaluations needed to finish the algorithm run.

*Table 1.* Performance of PLES and ASA

| | PLES | ASA |
|---|---|---|
| Best | 18.83 | 24.24 |
| Average | 21.48 | 26.28 |
| Worst | 23.40 | 28.25 |
| St. Dev. | 0.91 | 0.96 |
| Avg. Evaluations | 307,618 | 367,333 |

To obtain one solution the algorithms needed approximately 200.000 to 500.000 evaluations, which in time took 6 to 15 minutes. To transport the loads, all vehicles had to perform about 7 runs.

Figure 1 presents the performance of both algorithms, where the average, minimal and maximal values are presented in the graph. It is shown

that PLES converges faster and it returns lower optimal solutions than ASA. Even worst solution returned by PLES is better than the best solution returned by ASA.

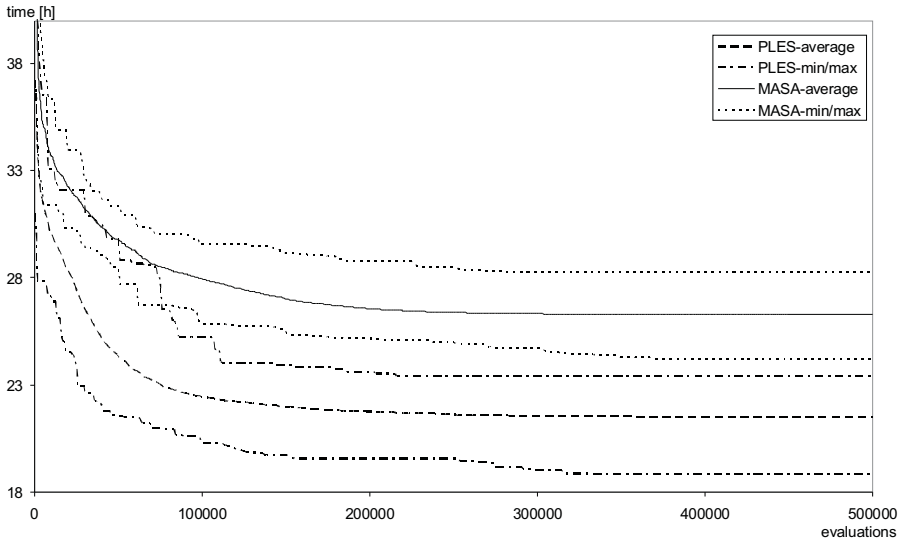



*Figure 1.* Performance of PLES and ASA.

## 5. Conclusions

The presented results show that PLES achieved better results than ASA. To get even better results with PLES the encoding of the problem should be improved, i.e., chromosome length should be further decreased. This would improve the statistical evaluation of solutions and improve the moves of solutions towards the optimal one. As a result the search power of the algorithm would be increased. From the ASA results one can see that limiting the solution space is not the proper way to go. The constraints imposed with this prevented the ASA to find solutions that would be even close to PLES. In the future work we will try to remove this limit with the use of some kind of different search graph.

## References

[1] G.B. Dantzig and J.H. Ramser. The Truck Dispatching Problem. *Manage. Sci.*, 6(1):80–91, 1959.

[2] M. Dorigo, G. Di Caro, and L.M. Gambardella. Ant algorithms for discrete optimization. *Artif. Life*, 5(2):137–172, 1999.

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

[4] P. Korošec and J. Šilc. The multilevel ant stigmergy algorithm: an industrial case study. In *Proc. 7th Int. Conf. Computational Intelligence and Natural Computing*, Salt Lake City, UT, 2005.

[5] K. Oblak, P. Korošec, F. Kosel, and J. Šilc. Multi-parameter numerical optimization of selected thin-walled machine elements using a stigmergic optimization algorithm. *Thin Wall. Struct.*, 45(12):991–1001, 2007.

[6] G. Papa. Parameter-Less Evolutionary Search. In *Proc. Genetic and Evolutionary Computation Conference (GECCO'08)*, Atlanta, GE, 2008.

[7] T. Tušar, P. Korošec, G. Papa, B. Filipič, and J. Šilc. A comparative study of stochastic optimization methods in electric motor design. *Appl. Intell.*, 27(2):101–111, 2007.

# PROTEIN FUNCTION PREDICTION USING ACO AND BAYESIAN NETWORKS

Mohammad Ehsan Basiri
*Department of Computer Engineering, Faculty of Engineering, University of Isfahan*
*Isfahan, Iran*
Basiri@eng.ui.ac.ir

Shahla Nemati
*Islamic Azad University Arsenjan Branch*
*Fars, Iran*
s.nemati@ec.iut.ac.ir

Nasser Ghasem-Aghaee
*Department of Computer Engineering, Faculty of Engineering, University of Isfahan*
*Isfahan, Iran*
Aghae@eng.ui.ac.ir

**Abstract** Feature selection and feature extraction are the most important steps in classification systems. The ASFS algorithm is a feature selection technique based on ACO algorithm which is a branch of newly developed form of artificial intelligence called Swarm Intelligence (SI). This paper empowers the ASFS algorithm by enabling the ASFS to select features for a Bayesian network algorithm; which is more sophisticated than the Nearest Neighbor classifier previously used by the original ASFS algorithm. This paper then compares the performance of the ASFS algorithm against the performance of a standard Binary PSO algorithm on the task of selecting features on Postsynaptic data sets. The criteria used for this comparison are (1) maximizing predictive accuracy; and (2) finding the smallest subset of features. Simulation results on Postsynaptic dataset show the superiority of the proposed algorithm.

**Keywords:** Ant Colony Optimization (ACO), Bayesian networks, Bioinformatics, Feature Selection, Particle Swarm Optimization (PSO)

## 1. Introduction

Feature Selection (FS) is a commonly used step in machine learning, especially when dealing with a high dimensional space of features. The objective of feature selection is to simplify a dataset by reducing its dimensionality and identifying relevant underlying features without sacrificing predictive accuracy. By doing that, it also reduces redundancy in the information provided by the selected features.

In real world problems FS is a must due to the abundance of noisy, irrelevant or misleading features. Feature selection is extensive and it spreads throughout many fields, including text categorization, data mining, pattern recognition and biometrics [9]. Given a feature set of size $n$, the FS problem is to find a minimal feature subset of size $m$ ( $m < n$) while retaining a suitably high accuracy in representing the original features.

Among too many methods which are proposed for FS, population-based optimization algorithms such as Genetic Algorithm (GA)-based method, Particle Swarm Optimization (PSO)-based method and Ant Colony Optimization (ACO)-based method have attracted a lot of attention. These methods attempt to achieve better solutions by application of knowledge from previous iterations.

Meta-heuristic optimization algorithm based on ant's behavior (ACO) was represented in the early 1990s by M. Dorigo and colleagues [7]. ACO algorithm is inspired by ant's social behavior. Ants have no sight and are capable of finding the shortest route between a food source and their nest by chemical materials called pheromone that they leave when moving [3].

The work in [2] proposed an ACO-based algorithm for the feature selection task of data mining. Hereafter, this algorithm will be referred to as Ant System for Feature Selection (ASFS). Although specifically designed for the task of feature selection in bioinformatics datasets, the ASFS is not limited to this type of application. It has been successfully applied to other domains such as text mining [1] and speaker verification systems [12].

This paper extends previous work reported in [2] by taking advantage of Bayesian classification. We enable ASFS to select features for a Bayesian network algorithm, which is more sophisticated than the Nearest Neighbor classifier previously used. Then it is applied to the problem of predicting whether or not a protein has a post-synaptic activity, based on features of protein's primary sequence.

The rest of this paper is organized as follows. Section 2 briefly addresses Bayesian networks. Feature selection and ASFS algorithm are

described in Sections 3. Section 4 reports computational experiments. It also includes a brief discussion of the results obtained and finally the conclusion is offered in the last section.

## 2. Bayesian Classification

Bayesian classifiers are statistical classifiers that can predict class membership probabilities, such as the probability that a given sample belongs to a particular class. Bayesian classification is based on the Bayes theorem [8].

A Bayesian network (BN), by contrast, detects probabilistic relationships among features and uses this information to help the feature selection process. Bayesian networks are a graph-based model for representing probabilistic relationships between random variables of a given problem domain [10]. This graphical representation is a Directed Acyclic Graph (DAG) in which nodes represent the variables of the problem and arcs represent conditional probabilistic independencies among them [6].

Learning the structure of a BN is an NP-hard problem [4]. Many algorithms developed to this end use a scoring metric and a search procedure. The scoring metric evaluates the goodness-of-fit of a structure to the data. The search procedure generates alternative structures and selects the best one based on the scoring metric.

To reduce the search space of networks, only candidate networks in which each node has at most $k$ inward arcs (parents) are considered $k$ is a parameter determined by the user. In the present work $k$ is set to 10 to avoid overly complex models.

A greedy search algorithm is used to generate alternative structures for the BN. Starting with an empty network, the greedy search algorithm adds into the network the edge that most increases the score of the resulting network. The search stops when no other edge addition improves the score of the network. In this work, We implement a generic greedy search algorithm to generate alternative structures for the BN which has been proposed in [6].

To evaluate the score of a network structure, we use a 10-fold cross-validation. We divide the data set into 10 equally sized folds. For all class levels each fold maintains roughly the same proportion of classes present in the whole data set before division (called stratified cross-validation). Eight of the ten folds are used to compute the probabilities for the bayesian network. The ninth fold is used as validation set and the tenth fold as test set. During the search for the network structure only the validation set is used to compute predictive accuracy. The score of the candidate networks is given by the predictive accuracy of

the classification of the proteins in the validation set. The network that shows the highest predictive accuracy on the validation set is then used to compute the predictive accuracy on the test set. Once the network structure is selected, the nine folds are merged and this merged data set is used to compute the probabilities for the selected Bayesian network. The predictive accuracy is then computed on the previously untouched test set fold. Every fold will be once used as validation set and once used as test set. A similar process is adopted for the computation of the predictive accuracy using the Nearest Neighbor classifier.

## 3. Feature Selection and ASFS Algorithm

The feature selection task may be reformulated into an ACO-suitable problem. ACO requires a problem to be represented as a graph. Here nodes represent features, with the edges between them denoting the choice of the next feature. The search for the optimal feature subset is then an ant traversal through the graph where a minimum number of nodes are visited that satisfies the traversal stopping criterion.

The heuristic desirability of traversal and edge pheromone levels are combined to form the so-called probabilistic transition rule, denoting the probability of ant $k$ at feature $i$ choosing to travel to feature $j$ at time $t$:

$$P_{ij}^{k}(t) = \begin{cases} \frac{[\tau_{ij}(t)]^{\alpha}.[\eta_{ij}]^{\beta}}{\sum_{l \in J_i^k} [\tau_{il}(t)]^{\alpha}.[\eta_{il}]^{\beta}} & \text{if } j \in J_i^k \\ 0 & otherwise \end{cases} \tag{1}$$

Where, $\eta_{ij}$ is the heuristic desirability of choosing feature $j$ when at feature $i$ ($\eta_{ij}$ is optional but often needed for achieving a high algorithm performance), $J_i^k$ is the set of neighbor nodes of node $i$ which have not yet been visited by the ant $k$. $\alpha > 0, \beta > 0$ are two parameters that determine the relative importance of the pheromone value and heuristic information (the choice of $\alpha$, $\beta$ is determined experimentally) and $\tau_{ij}(t)$ is the amount of virtual pheromone on edge $(i,j)$.

The overall process of ASFS algorithm can be seen in Fig. 2. The pheromone on each edge is updated according to the following formula:

$$\tau_{ij}(t+1) = (1-\rho).\tau_{ij}(t) + \sum_{k=1}^{m} \Delta_{ij}^{k}(t) \tag{2}$$

where:

$$\Delta_{ij}^{k}(t) = \begin{cases} \gamma'(S^k)/|S^k| & if\ (i,j) \in S^k \\ 0 & otherwise \end{cases} \tag{3}$$

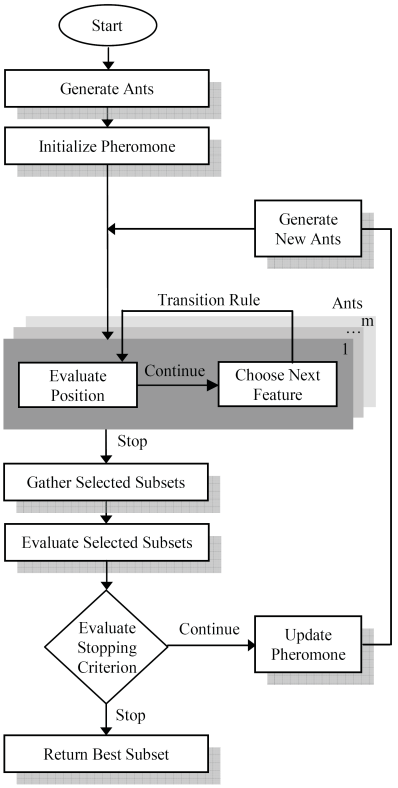



*Figure 1.* The ASFS feature selection algorithm.

The value $0 \leq \rho \leq 1$ is decay constant used to simulate the evaporation of the pheromone, $S^k$ is the feature subset found by ant $k$. The pheromone is updated according to both the measure of the "goodness" of the ant's feature subset ($\acute{\gamma}$) and the size of the subset itself. By this definition, all ants can update the pheromone.

## 4. Experimental Results

### 4.1 Postsynaptic Dataset

The dataset used in this paper is called the Postsynaptic dataset. It has been recently created and mined in [2, 5, 13]. The dataset contains 4303 records of proteins. These proteins belong to either positive or negative classes. Proteins that belong to the positive class have postsynaptic activity while negative ones don't show such activity. From the

4303 proteins on the dataset, 260 belong to the positive class and 4043 to the negative class.

This dataset has many features which makes the feature selection task challenging. More precisely, each protein has 443 PROSITE patterns, or features. PROSITE is a database of protein families and domains. It is based on the observation that, while there are a huge number of different proteins, most of them can be grouped, on the basis of similarities in their sequences, into a limited number of families (a protein consists of a sequence of amino acids). PROSITE patterns are small regions within a protein that present a high sequence similarity when compared to other proteins. In our dataset the absence of a given PROSITE pattern is indicated by a value of 0 for the feature corresponding to that PROSITE pattern which its presence is indicated by a value of 1 for that same feature [5].

## 4.2 Experimental Methodology

As mentioned earlier, the computational experiments involved a ten-fold stratified cross-validation method [14]. First, the 4303 records in the Postsynaptic dataset were divided into 10 almost equally sized folds. There are three folds containing 431 records each one and seven folds containing 430 records each one. In each of the 10 iterations of the cross-validation procedure, the predictive accuracy of the classification is assessed by 3 different methods:

- Using all the 443 original features: all possible features are used by the Nearest Neighbor classifier and the Bayesian network.
- Standard binary PSO algorithm: only the features selected by the best particle found by the binary PSO algorithm are used.
- Proposed ASFS algorithm: only the features selected by the best ant found by the ASFS algorithm are used.

Various values were tested for the parameters of ASFS algorithm. The results show that the highest performance is achieved by setting the parameters to values shown in Table 1.

*Table 1.* Standard binary PSO and ASFS parameter settings.

| | *Population* | *Iteration* | *Initial pheromone* | $c1$ | $c2$ | $w$ | $\alpha$ | $\beta$ | $\rho$ |
|---|---|---|---|---|---|---|---|---|---|
| PSO | 30 | 50 | - | 2 | 2 | 0.8 | - | - | - |
| ASFS | 30 | 50 | 1 | - | - | - | 1 | 0.1 | 0.2 |

Where, $w$ is inertia weight and $c1$ and $c2$ are acceleration constants of standard binary PSO algorithm. The choice of the value of this parameter was based on the work presented in [5]. For ASFS, parameter values were empirically determined in our preliminary experiments for leading to better convergence; but we make no claim that these are optimal values. Parameter optimization is a topic for future research.

### 4.3 Performance Measure

In this work we use the following measurement which has also been used before in [2, 13].

$$Predictive\ accuracy\ rate = TPR \times TNR \tag{4}$$

Where, $TPR$ and $TNR$ are defined az follows:

$$TPR = \frac{TP}{TP + FN}, TNR = \frac{TN}{TN + FP}. \tag{5}$$

Where, $TP$ (true positives) is the number of records correctly classified as positive class and $FP$ (false positives) is the number of records incorrectly classified as positive class. $TN$ (true negatives) is the number of records correctly classified as negative class and $FN$ (false negatives) is the number of records incorrectly classified as negative class.

### 4.4 Results

Table 2 gives the optimal selected features for each method. As discussed earlier the experiments involved 200 runs of ASFS and standard binary PSO, 10 cross-validation folds times 20 runs with different random seeds. Presumably, those 200 runs selected different subsets of features. So, the features which have been listed in Table 2 are the ones most often selected by ASFS and standard binary PSO across all the 20 runs. Both ACO-Based and PSO-Based methods significantly reduce the number of original features, however; ACO-Based method, ASFS, chooses fewer features.

Also, the results of both algorithms for all of the 10 folds are summarized in Table 3. The classification quality and feature subset length are two criteria which are considered to assess the performance of algorithms. Comparing these criteria, we noted that ASFS and standard binary PSO algorithms did very better than the baseline algorithm (using all features). Furthermore, for all of the 10 folds the ASFS algorithm selected a smaller subset of features than the standard binary PSO algorithm.

As we can see in Table 3, the average number of selected features for standard binary PSO algorithm was equal to 15.4 with the average

*Table 2.* Selected features of standard binary PSO and ASFS algorithms

| Method | Selected Features | Number of Selected features |
|---|---|---|
| Binary PSO | 134,162,186,320,321<br>333,342,351,352,353 | 10 |
| ASFS | 352,381,419,353,342 | 5 |

*Table 3.* Comparison of obtained results for ASFS and standard binary PSO

| | Using all the 443 original attribute | Standard Binary PSO algorithm | | Proposed ASFS Algorithm | |
|---|---|---|---|---|---|
| Fold | TPR × TNR | TPR × TNR | No. of Features selected | TPR × TNR | No. of Features selected |
| 1 | 1.00 | 1.00 | 16 | 1.00 | 4 |
| 2 | 1.00 | 1.00 | 19 | 1.00 | 2 |
| 3 | 1.00 | 1.00 | 21 | 1.00 | 4 |
| 4 | 0.73 | 0.76 | 14 | 0.85 | 3 |
| 5 | 0.00 | 0.63 | 17 | 0.76 | 5 |
| 6 | 0.00 | 0.62 | 18 | 0.87 | 5 |
| 7 | 0.92 | 0.88 | 11 | 0.95 | 2 |
| 8 | 1.00 | 0.69 | 15 | 1.00 | 6 |
| 9 | 0.73 | 0.73 | 9 | 0.79 | 3 |
| 10 | 0.42 | 0.42 | 14 | 0.54 | 4 |
| **AVG** | **0.68** | **0.77** | **15.4** | **0.88** | **3.8** |

predictive accuracy of 0.77 and the average number of selected features for ASFS algorithm was equal to 3.8 with the average predictive accuracy of 0.88. Furthermore, in [5] a new discrete PSO algorithm, called DPSO, has been introduced for feature selection. DPSO has been applied to Postsynaptic dataset and the average number of features selected by that was 12.70 with the average predictive accuracy of 0.74. Comparison of these three algorithms shows that ASFS tends to select a smaller subset of features than the standard binary PSO algorithm and DPSO. Also, the average predictive accuracy of ASFS is higher than that of the standard binary PSO algorithm and DPSO.

*Table 4.* Standard binary PSO and ASFS. Paired two-tailed t-test for the predictive accuracy and number of selected features with significance level 0.05.

| | Nearest Neighbor | Bayesian network |
|---|---|---|
| Predictive Accuracy | t(9) = 2.206, p = 0.053 | t(9) = 3.200, p = 0.023 |
| No. of Selected Features | t(9) = 7.635, p = 4.8E-5 | t(9) = 8.266, p = 1.6E-6 |

Furthermore, the ASFS algorithm did slightly better than the standard binary PSO algorithm. Nevertheless, the difference in the predictive accuracy performance between these algorithms is, in some cases, statistically insignificant. Table 4 shows the results of a paired two-tailed t-test for the predictive accuracy of the standard binary PSO versus the predictive accuracy of the ASFS at a significance level of 0.05.

Table 4 shows that, using Nearest Neighbor as classifier, there is no statistically significant difference in performance (in terms of predictive accuracy) between the standard binary PSO and ASFS algorithms. By contrast, using Bayesian networks as classifier the difference in performance is statistically significant. Nevertheless, the discriminating factor between the performance of these algorithms is on the number of selected features criterion. The ASFS not only outperformed the standard binary PSO in predictive accuracy, but also did so using a smaller subset of attributes. Moreover, when it comes to effectively pruning the set of features, the difference in performance between the standard binary PSO and the ASFS is always statistically significant, as Table 4 shows.

## 5. Conclusion

Experimental results show that the use of unnecessary Features decrease classifiers' performance and hurt classification accuracy and FS is used to reduces redundancy in the information provided by the selected features. Using only a small subset of selected features, the Binary PSO and the ASFS algorithms obtained better predictive accuracy than the baseline algorithm using all Features. Previous work had already shown that the ASFS algorithm outperforms the Binary PSO in the task of Featur selection [2].

The ASFS clearly enhances computational efficiency of the classifier by selecting fewer features than the standard Binary PSO algorithm. Therefore, when the difference in predictive accuracy is insignificant, ASFS is still preferable.

Also as we expected, computational results show the clear difference in performance between Nearest Neighbor and Bayesian networks classi-

fiers. The Bayesian networks approach outperformed the Nearest Neighbor approach in all experiments.

**Acknowledgments** The authors wish to thank the Office of Graduate studies of the University of Isfahan for their support.

## References

[1] M.H. Aghdam, M.E. Basiri, and N. Ghasem-Aghaee. Application of Ant Colony Optimization for Feature Selection in Text Categorization. In *Proc. IEEE World Congress on Computational Intelligence (WCCI 2008)*, pages 2872–2878, Hong Kong, 2008.

[2] M.E. Basiri, N. Ghasem-Aghaee, and M.H. Aghdam. Using Ant Colony Optimization-Based Selected Features for Predicting Post-synaptic Activity in Proteins. *Lect. Notes Comp. Sc.*, 4973:12–23, 2008.

[3] E. Bonabeau, M. Dorigo, and G. Theraulaz. *Swarm Intelligence: From Natural to Artificial Systems*. Oxford University Press, 1999.

[4] D.M. Chickering, D. Geiger, and D. Heckerman. Learning Bayesian networks is NP-hard. Technical Report MSR-TR-94-17, Microsoft Research, 1994.

[5] E.S. Correa, A.A. Freitas, and C.G. Johnson. A new discrete particle swarm algorithm applied to attribute selection in a bioinformatics dataset. In *Proc. Genetic and Evolutionary Computation Conference (GECCO-2006)*, pages 35–42, Seattle, 2006.

[6] E.S. Correa, A.A. Freitas, and C.G. Johnson. Particle Swarm and Bayesian Networks Applied to Attribute Selection for Protein Functional Classification. In *Proc. Genetic and Evolutionary Computation Conference (GECCO-2007)*, pages 2651–2658, London, UK, 2006.

[7] M. Dorigo and G. Di Caro. Ant Colony Optimization: A New Meta-heuristic. In: Proc. IEEE Congress on Evolutionary Computation (CEC 1999), pages 1470–1477, Washington, D.C., 1999.

[8] W. Feller. *An introduction to probability theory and its applications*. John Wiley & Sons, Inc., New York, 1971.

[9] R. Jensen. Combining rough and fuzzy sets for feature selection. Ph.D. Thesis, University of Edinburgh, 2005.

[10] F.V. Jensen. *Bayesian networks and decision graphs*. Springer-Verlag, 1st edition, 2001.

[11] B. Liu, H.A. Abbass, and B. McKay. Classification Rule Discovery with Ant Colony Optimization. In: Proc. IEEE/WIC International Conference on Intelligent Agent Technology (IAT 2003), pages 83–88, 2003.

[12] S. Nemati, R. Boostani, and M.D. Jazi. A Novel Text-Independent Speaker Verification System Using Ant Colony Optimization Algorithm. *Lect. Notes Comput. Sc.*, 5099:421-429, 2008.

[13] G.L. Pappa, A.J. Baines, and A.A. Freitas. Predicting post-synaptic activity in proteins with data mining. *Bioinformatics*, 21(2):19–25, 2005.

[14] I.H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, San Francisco, 2005.

# THE APPLICATION OF SMART CONTROL TECHNIQUE IN SCHEDULING OF ELECTRICAL POWER GENERATION

Nadjamuddin Harun, Yusri S. Akil
*Department of Electrical Engineering, Engineering faculty, Hasanuddin University Makassar, South Sulawesi, Indonesia*
{n_harun; yakil}@unhas.ac.id

**Abstract** This study aims to find a new and better method in solving the electrical power generation scheduling to minimize the generation cost. The method of scheduling used is smart control technique of fuzzy logic system with a patron derived from the output of LaGrange method. This system uses once fuzzification process, and the amount of each input and output variable is one (SISO). Initial condition of the power system analyzed is obtained by using Newton Raphson method and input-output characteristic equations determined with using Least Square method. Scheduling of power generation analyzed are thermal plants interconnected on 30 kV, 70 kV and 150 kV line transmission at South Sulawesi electrical power system. The results of scheduling of generation of thermal plants load range (90.3 MW - 120.30 MW) by using MATLAB software is to give highest for 116.60 MW and lowest efficiency for 90.30 MW load. The economizing of operational cost / total cost efficiency used smart control technique are average of Rp.4,675.75/MWh or 23.4%.

**Keywords:** Scheduling of power generation, Smart control

## 1. Introduction

An electrical power system consists of power generation, transmission line and distribution system [4]. These element in its operation formed interconnection each other. Power plant center converts primary energy resources to electric energy, generally in application consist of some unit generators [7]. The profile of operation in serving electrical power system load collectively require a control on scheduling and the amount of generated power on every generator unit, so that a minimum cost operation can be accomplished.

Electrical power generation operation in a scheduled electrical power system requires interconnection among generation centers and load centers through the transmission line. Usually in a power plant system, generator units with different characteristic are operated simultaneously [4]. This operation situation, insist on an excellent management electrical power plant operation in order to optimize the existing electrical tools. One way to could be done by optimizing thermals power plant operation cost till minimum level by scheduling. The optimization however must concern with the expected electrical energy quality.

Beside that, big load of electrical power system is random. The big load can not be predicted and some generator unit characteristic cause the coordinated power output control is very important to assure generation to a balance load so that system frequency close to the nominal operation value.

Based on this condition, the problem of automatic generation control is developed from the steady state point of view. The coordinated generation scheduling can be conducted by smart control technique of fuzzy logic system. This system could process uncertain data, imprecision, and could be implemented with cheap cost. In taking human mind as a model, enables to solve problem where mathematic formula can not be made clearly in getting wanted output. This technique is based on the artificial intelligent, which consist of a number field of study and one of them is fuzzy logic system. In application, automatic generation control works automatically based on the designed and saved program in the computer memory.

## 2. Methodology

The research object is South Sulawesi electrical power system that covers 23 generator units that grouped into 7 power plants center with 20 load center and 27 power transmission lines that connecting load and power plant center. The research of scheduling focuses on non-private thermal power plant. Simulation software used is MATLAB (Matrix Laboratory) and actual data used in analysis consist of one line diagram of South Sulawesi system, data of electric power system, input-output data and thermal power plant units, power flow for several loading condition, data of transmission line constant.

The research start with load flow study used polar coordinate Newton Raphson method to determine initial condition or normality of electrical power system analyse. Load flow study conducted for several load condition at every bus agree with pattern of South Sulawesi electrical power system. Initial condition is as voltage per unit for every bus,

power losses and phase angle of South Sulawesi system. Next it is to determine input-output characteristic equation for every generator unit based on input-output data (operation) by using Least Square method. This equation aims to schedule as objective function and could be calculated with using formula:

$$F_m = a_m P_m^2 + b_m P_m + c_m \tag{1}$$

where $m$ is the $m$-th plant.

The next step is optimization by using LaGrange method for load in working interval of thermal plant peak load time. Based on daily load curve on the date of May 14, 2001 and losses calculation at the peak load in South Sulawesi interconnected electrical power system, the average load of 198.5 MW with maximum thermal generator of 123.79 MW and minimum load 90.3 MW are obtained. From the above data, the load range can be determined which will be set to the fuzzy smart control input variable. Optimization output with LaGrange method is patron for smart control of fuzzy logic system. In this case ones fuzzification process is employed, using input and output variable with each has the sum of unity or can be categorized single-input single-output (SISO) fuzzy logic. The input variable refers to thermal generator load information and the output variable refers to the amount of economic load of each thermal generator unit. The input membership degree utilized is functional definition. On the input and output variable, a trapezium shape is employed for the 1st and $n$-th membership function and a triangle shape is employed for the 2nd to $(n-1)$-th membership function. Peak load information from the load center is obtained and processed by five implication rules.

The fuzzy system smart control technique design consist of three stages, that is: fuzzification stage, fuzzy inference, defuzzification stage. The early design stage (fuzzification strategy) the membership function of input and output variable are made overlap. This is to assure that no crisp value is not included in the existing sets and to make possible that more then one rule involve in determining output. For instance the $n$-th membership function is a triangle function that owned $a$, $b$ and $c$ values as LOW, MEDIUM, and HIGH values. The LOW value from the $(n-1)$ membership function must take the MEDIUM value from the $(n-2)$ membership function and the HIGH value must take MEDIUM value from $n$ membership function, whereas the MEDIUM value is HIGH value from membership function $(n-2)$ and also is a LOW value from n membership value. To control the thermal generator operation on the studied system, the optimized objective function is the operation cost function with constraint function which is electrical load balance on ev-

*Table 1.* Linguistic part and function data of input variable membership

| No | Linguistic Part | LOW (MW) | MED (MW) | HIGH (MW) |
|---|---|---|---|---|
| 1 | VERY LOW | 90 | 91.5 | 99.25 |
| 2 | LOW | 91.5 | 99.25 | 107 |
| 3 | MEDIUM | 99.25 | 107 | 114.75 |
| 4 | HIGH | 107 | 114.75 | 122.5 |
| 5 | VERY HIGH | 114.75 | 122.5 | 124 |

*Table 2.* Linguistic part and function data of output variable membership of PLTG GE I generator unit

| No | Linguistic Part | LOW (MW) | MED (MW) | HIGH (MW) |
|---|---|---|---|---|
| 1 | NSMALL | 13.901 | 15.388 | 23.073 |
| 2 | SMALL | 15.388 | 23.073 | 30.759 |
| 3 | MEDIUM | 22.073 | 30.759 | 32 |
| 4 | BIG | 30.759 | 32 | 32.5 |
| 5 | PBIG | 32 | 32.5 | 33 |

ery thermal generator unit. Input and output variable setting (PLTG GE 1 generator unit) are divided into 5 linguistic parts, as can be observed in Table 1 and 2. While 4 other thermal plant units are PLTU 2, PLTG WESTCAN, PLTG ALSTOM 1 and MIRRLES 1, its output variable setting is determined with the same way before (PLTG GE 1).

For the inference stage or coordination strategy in calculating fire strength that make possible for the studied system, i.e. fulfill the determined rules. The next step is the defuzzification to find the crisp value from the output variable. The defuzzification method that is employed in this study is the method of Center of Maximum (COA). All the steps are used in order to generation output can be optimized with minimum cost for every load change.

## 3. Results

Practically such in South Sulawesi, load of an electrical power system always changed. The big this load changes is influenced by a number of factors for each point so that its difficult to predict exactly how big each time [3]. In reality, this changes bases to power output control coordi-

*Table 3.* Rules for thermal generation operation control

| IF Load | THEN Power |
|---|---|
| VERY LOW | NSMALL |
| LOW | SMALL |
| MEDIUM | MEDIUM |
| HIGH | BIG |
| VERY HIGH | PBIG |

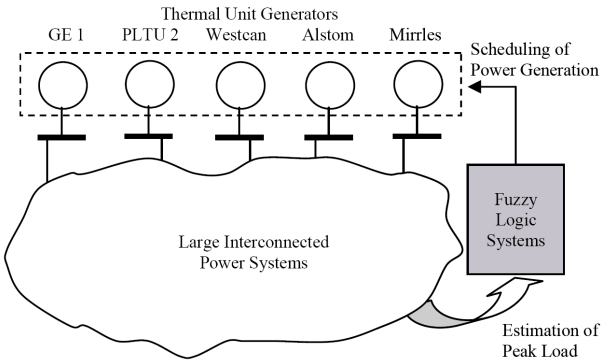



*Figure 1.* Scheduling of power generation using smart control technique of fuzzy logic system.

nated, in which, is very needed to assure generation to a balance load so that system frequency close to the nominal operation value. Economic generation scheduling coordinated could be conducted by using smart control technique base on artificial intelligent. Economic scheduling application involves analysis procedures that need system electrical data input and generation [2].

In the previous part pointed out that research focus is non private thermal plants interconnected in the South Sulawesi System. The existing thermal plants and fuel types used covers 8 units namely: GE 1 (HSD), GE 2 (HSD), MITSUBISHI (MFO), SWD (MFO), PLTU 2 (MFO), WESTCAN (HSD), ALSTOM (HSD), MIRRLES (MFO). The price fuel in the year were studied conducted namely HSD (High Speed Diesel) is Rp.600.00/liter and MFO (Marine Fuel Oil) is Rp.400.00/liter. Thermal units GE 2, MITSUBISHI and SWD must generated to be maximum when peak load time begin (based on data) so that the three units are not used smart control. Range of peak load analysed up 18.00 to 20.00 time with range of thermal load up 90.30 to 120.30 MW.

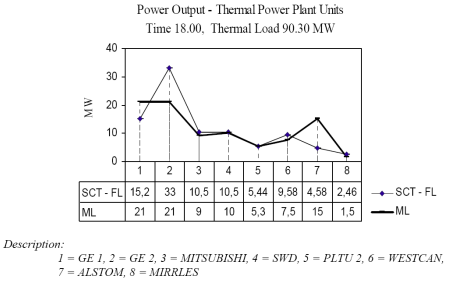



*Figure 2.* Load distribution on thermal plant units using smart control technique of fuzzy logic system (SCT- FL) with thermal load 90.30 MW and merit loading (ML) in South Sulawesi.

## 3.1 Thermal Plant Scheduling at Time 18.00–20.00 with Thermal Load Range 90.30–120.30 MW

The result of generation scheduling with smart control especially at time 18.00 such in the Fig. 2 shows big load that must be supplied by each thermal plant units are 15.2 MW for GE 1; 33 MW for GE 2; 10.50 MW for MITSUBISHI; 10.50 MW for SWD; 5.44 MW for PLTU 2; 9.58 MW for WESTCAN; 4.58 MW for ALSTOM and 2.46 MW for MIRRLES. While load distribution must be supplied by thermal plant units is up 18.30 to 20.00 time shown in the Fig. 4.

The arrangement of generation scheduling uses smart control technique of fuzzy logic system is more efficient compare with conventional method applied during the time. The increasing of efficiency could be caused by softcomputing method result which applied during scheduling process, in which, has higher respond capacity and automatic toward load changes occuring randomly. Beside the increasing of efficiency stated above, other more important advantage of fuzzy control is easily implemented [1].

In this system model, a fuzzy logic controller is a closed circle system, where there is not operator in its part of controlled circle system [6].

Basically, output of this system should pass at censor system and than to be compared with reference value. If the result points to difference, so input variables system (E) should be mapped into fuzzy singleton in the linguistic parts with fuzzification interface (FI). Furthermore, this process is continued to decision making logic (DML) that should yield fuzzy conclusion in getting action from fuzzy control by evaluating a number of fuzzy rules (Knowledge Base) for each fuzzified input. For output, defuzzification interface (DFI) is a part of the last step to draw

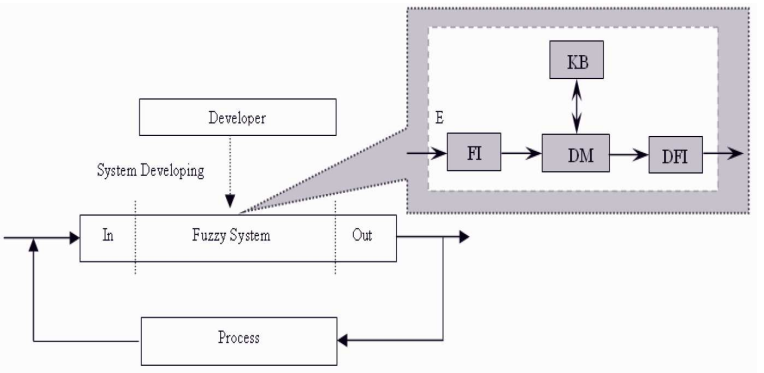



*Figure 3.* Smart control technique of fuzzy logic system (SCT- FL).

fuzzy conclusion. This part includes giving heavy and combination of several fuzzy set that giving crisp fuzzy singleton from each output.

The result of smart control load distribution that its input patron comes from LaGrange method cause chosen automatically thermal plant unit with certain big load that must be supplied. LaGrange function is needed to establish the necessary condition for an extreme value of the objective function, add the constraint function to the objective function after constraint function has been multiplied by an undetermined multiplier [5]. If total thermal load reaches 90.30 - 120.30 MW so optimal scheduling control for all existing thermal units interconnected on transmission line 30 kV, 70 kV, and 150 kV will operate. On the time of 18.00, PLTG GE 1 - 150 kV has capacity 33.00 MW that supplying power 15.20 MW. The big supply is based on input patron used for this control is derived from LaGrange method that giving limitation of the highest value 33 MW for load 114.75 - 124 MW and the lowest 13.901 for load 90 MW. PLTU 2 interconnected at Tello 30 kV supplying power 9.58 MW with capacity 11.50 MW.

The limitation of control for range load is set in this unit patron maximally 11.50 MW in the load 124 MW and minimally 5.44 MW for load 90 MW. PLTG WESTCAN - 30 kV supplies 9.58 MW. The limitation of this unit is the highest for load 124 MW namely 12.5 MW and the lowest for load 90 MW namely 9.54 MW. PLTG ALSTOM - 30 kV supplies 4.58 MW with capacity 10 MW. Control range for this unit is up 4.58 MW to 10 MW for load 90 MW and 124 MW respectively. PLTD MIRRLES unit - 30 kV supplies 2.46 MW. This unit is the lowest capacity (3 MW) for the others units that supplying peak load with output control limitation is up 2.43 MW to 3 MW. While the result of scheduling control for others time period (time 18.30 - 20.00) could

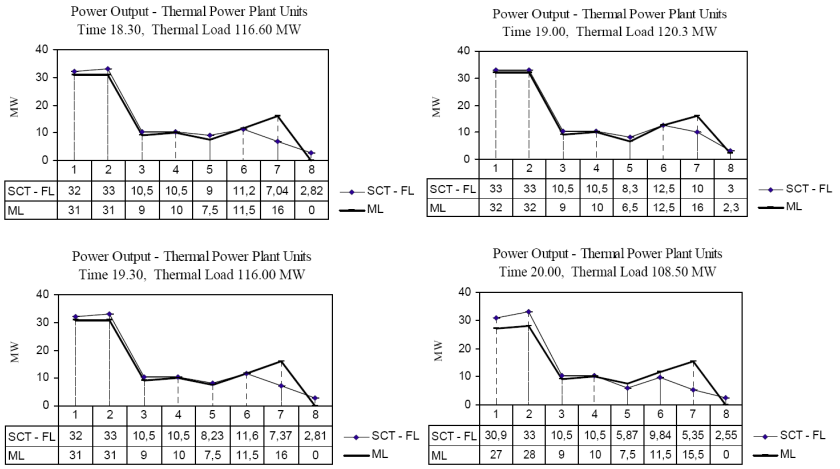



*Figure 4.* Load distribution on thermal plant units Time 18.30 - 20.00 using smart control fuzzy logic system (SCT-FL) with thermal load total 108.50–120.30 MW compared with merit loading (ML) scheduling in South Sulawesi.

be explained such before. Load distribution for each unit scheduled is shown in the Fig. 4.

Based on the whole time period observed (time 18.00 - 20.00), most parts of the highest power supply occured at the time of 19.00 (except PLTU 2 at the time of 18.30 namely 9.00 MW) for each unit: PLTG GE 1 is amount of 33.00 MW, WESTCAN is amount of 12.50 MW, ALSTOM is amount of 10.00 MW and MIRRLES is amount of 3.00 MW. This condition is caused by the highest thermal load occured at the time of 19.00 so that most thermal units operating on the its maximum capacity with output control comes from set patron.

## 3.2 Optimization and Operational Cost Efficiency Using SCT-FL Technique

As the comparison parameter to fuzzy system smart control technique application, an operation cost comparison table which is obtained with this technique application and thermal generator operation scheduling with Merit Loading method (ML) are as follows:

Table shown above is the amount of cost and efficiency are achieved by using smart control. The amount of cost for every plant unit is yielded with fuel cost equation (Rp./Hour) for each unit. This equation is started with forming input-output equation (Liter/Hour), and the next

*Table 4.* Optimization result using fuzzy logic system smart control technique

| Load (MW) | Operational Cost (Rp./MWh) | |
|---|---|---|
| | Scheduling by ML | Optimization Results |
| 90.30 | 210,142.90 | 206,760.86 |
| 116.60 | 197,157.21 | 191,560.23 |
| 120.30 | 195,157.41 | 192,111.52 |
| 116.00 | 198,176.99 | 192,567.54 |
| 108.50 | 201,170.98 | 195,683.59 |

*Table 5.* Operational cost efficiency using fuzzy logic system smart control technique

| Load (MW) | Cost Effciency | |
|---|---|---|
| | (Rp./MWh) | Percentage (%) |
| 90.30 | 3,382.04 | 16.094 |
| 116.60 | 5,596.98 | 28.388 |
| 120.30 | 3,302.90 | 16.902 |
| 116.00 | 5,609.45 | 28.305 |
| 108.50 | 5,487.39 | 27.277 |

is processed by using Least Square method. And than, this input-output equation is multiplied with fuel price for each unit.

Thermal plant scheduling consists of 5 units, 2 of them used MFO fuel (PLTU 2 and MIRRLES) and others units used HSD fuel (GE 1, WESTCAN and ALSTOM). In the lowest load thermal is 90.30 MW, almost each thermal units operated far from its capacity such as GE 1 supplies 15.20 MW (capacity 33.00 MW) and PLTU 2 supplies 5.44 MW (capacity 11.50 MW). Beside that, power supply is given to each of MFO fuel units (PLTU 2, MIRRLES) are not to big yet. Those factors caused operational cost efficiency achieved for this load is the lowest namely 16.094%. While the load range analyze is thermal load 116.60 MW that yielding the highest cost efficiency namely as amount of 28.388%. The highest efficiency is caused by almost each unit operating in maximum capacity, and especially for MFO fuel units such in the Fig. 4. The big load must be supplied automatically for each units, in which, is the output of scheduling control (output control based on units patron respectively)

Based on the result in Table 4 and 5, for every load employing smart control technique, it can be observed that generation cost can be mini-

mized. This shows the technique application can offer a meaningful efficiency in term of operational cost of thermal generator units in South Sulawesi electrical power system.

## 4. Conclusion

This paper investigates the possibility of fuzzy logic system smart control technique application in the scheduling of South Sulawesi electrical power generation on normal operation. The result of simulation is to give the highest for 116.60 MW and the lowest efficiency for 90.30 MW load. The economizing of operational cost / total cost efficiency used smart control technique are average of Rp.4,675.75/MWh or approximately 23.4%. This method application is simpler, more accurate, and can be scheduled automatically and economically, compare to that of employing conventional method with many graphical analysis and analytics. Based on the above excess, the smart control technique can be applied to the scheduling of generation operation in an electrical power system.

## Acknowledgments

The authors gratefully acknowledge to Gassing, Makmur, R., Waris, T., and Indrajaya, for their valuable comments. The author also wish to thank staff's PT. PLN (Persero) of South Sulawesi and South-East Sulawesi Area in providing data to completing the purpose of this paper.

## References

[1] E.F. Camacho, F.R. Rubio, and M. Berenguel. Application of Fuzzy Logic Controllers to a Solar Power Plant. In W. Mielczarski, (editor), *Fuzzy Logic Techniques in Power Systems*, Studies in Fuzziness and Soft Computing, Vol. 11, pages 136–174, Springer-Verlag, 1998.

[2] L.K. Kirchmayer. *Economic Operation of Power Systems*. Wiley Eastern Limited, New Delhi, 1958.

[3] A.C. Liew and D. Srinivasan. Fuzzy Logic Approach to load Modelling and Forecasting. In W. Mielczarski, (editor), *Fuzzy Logic Techniques in Power Systems*, Studies in Fuzziness and Soft Computing, Vol. 11, pages 241–276, Springer-Verlag, 1998.

[4] W.D. Stevenson. *Analisis Sistem Tenaga Listrik*. Penerbit Erlangga, Edisi Keempat, Jakarta, 1990.

[5] A.J. Wood and B.F. Wollenberg. *Power Generation, Operation and Control*. John Wiley & Sons, Inc., Singapore, 1984.

[6] A.T. Zebua and W. Wahab. *Teknologi sistem fuzzy*. Elektro Indonesia, 1995.

[7] Zuhal. *Dasar Teknik Tenaga Listrik dan Elektronika Daya*. Penerbit PT. Gramedia Pustaka Umum, Cetakan Keenam, Jakarta, 2000.

# INFORMATION SOCIETY 2008

In its 11th year, the Information Society Multiconference (`http://is.ijs.si`) continues as one of the leading conferences in Central Europe gathering scientific community with a wide range of research interests in information society. In 2008, we organized eight independent conferences forming the Multiconference. Information society displays a complex interplay of social, economic, and technological issues that attract attention of many scientific events around Europe. The broad range of topics makes our event unique among similar conferences. The motto of the Multiconference is synergy of different interdisciplinary approaches dealing with the challenges of information society. The major driving forces of the Multiconference are search and demand for new knowledge related to information, communication, and computer services. We present, analyze, and verify new discoveries in order to prepare the ground for their enrichment and development in practice. The main objective of the Multiconference is presentation and promotion of research results, to encourage their practical application in new ICT products and information services in Slovenia and also broader region.

The Multiconference is running in parallel sessions with 300 presentations of scientific papers. The papers are published in the conference proceedings, and in special issues of two journals. One of them is Informatica with its 31 years of tradition in excellent research publications.

The Information Society 2008 Multiconference consists of the following conferences:

- BIOMA 2008 - Bioinspired Optimization Methods and their Applications
- Cognitive Sciences
- Collaboration, Software and Services in Information Society
- Data Mining and Data Warehouses (SiKDD 2008)
- Education in Information Society
- Intelligent Systems
- Language Technologies
- Slovenian Demographic Challenges in the 21st Century

The Multiconference is co-organized and supported by several major research institutions and societies, among them ACM Slovenia, i.e. the Slovenian chapter of the ACM. We would like to express our appreciation to the Slovenian Government for cooperation and support, in particular through the Ministry of Higher Education, Science and Technology.

In 2008, the Programme and Organizing Committees decided to award one Slovenian for his/her outstanding contribution to development and promotion of information society in our country. With the majority of votes, this honor went to Prof. Dr. Ivan Rozman. Congratulations!

On behalf of the conference organizers we would like to thank all participants for their valuable contribution and their interest in this event, and particularly the revieewers for their thorough reviews.

Franc Solina, Programme Committee Chair

Matjaž Gams, Organizing Committee Chair