BIOINSPIRED OPTIMIZATION METHODS AND THEIR APPLICATIONS

BIOINSPIRED OPTIMIZATION METHODS AND THEIR APPLICATIONS

Proceedings of the Fifth International Conference on Bioinspired Optimization Methods and their Applications, BIOMA 2012

24–25 May 2012, Bohinj, Slovenia

Edited by
BOGDAN FILIPIČ
JURIJ ŠILC

Jožef Stefan Institute, Ljubljana

Jožef Stefan Institute
Ljubljana, Slovenia

Editors: Bogdan Filipič, Jurij Šilc

Cover design by Studio Design Demšar, Škofja Loka
Logo design by Gregor Papa
Printed by Tiskarna Artelj, Ljubljana
Published by Jožef Stefan Institute, Ljubljana, Slovenia

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

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

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

INTERNATIONAL Conference on Bioinspired Optimization Methods and their Applications (5 ; 2012 ; Bohinj)
Bioinspired optimization methods and their applications : proceedings of the Fifth International Conference on Bioinspired Optimization Methods and their Applications - BIOMA 2012, 24-25 May 2012, Bohinj, Slovenia / edited by Bogdan Filipič, Jurij Šilc. - Ljubljana : Jožef Stefan Institute, 2012

ISBN 978-961-264-043-9
1. Gl. stv. nasl. 2. Filipič, Bogdan

261676288

BIOMA 2012 is partially financed by the Slovenian Research Agency.

Contents

Preface

Natural phenomena, like the evolution of species, emergent behavior of biological societies, and functioning of the vertebrate immune system, have inspired computer scientists to design advanced problem-solving techniques, known as evolutionary algorithms, ant colony optimization, and artificial immune systems. These and many other bioinspired algorithms that continue to emerge, overcome many limitations of traditional algorithms and have been widely accepted in science, engineering and business. Solving intricate optimization problems in these domains is of particular interest to both designers and practitioners of bioinspired algorithms.

This volume contains recent theoretical and practical contributions to the field of bioinspired optimization presented at the Fifth International Conference on Bioinspired Optimization Methods and their Applications (BIOMA 2012), held in the Alpine resort of Bohinj, Slovenia, on 24 and 25 May 2012.

BIOMA continues its mission of bringing together theoreticians and end users to present their new achievements and promote their field of interest to wider audience. It also sticks to the tradition of a single-track event with a lot of opportunities for discussions and exchange of ideas. Unlike the previous BIOMA conferences that have been held biennially since 2004 at the Jožef Stefan Institute, Ljubljana, BIOMA 2012 takes places at a new location with picturesque mountain surroundings.

This year we received the highest number of papers so far. Each submitted paper was reviewed by three members of the international program committee. In the reviewing procedure, 28 papers were selected for presentation at the conference, which is 68% of the submissions. Together with two invited contributions the conference proceedings contain 30 papers by 70 (co)authors from 13 countries.

The BIOMA 2012 invited keynote speaker is Ágoston E. Eiben, the head of the Computational Intelligence Group at the VU University of Amsterdam. He is one of the European pioneers in evolutionary computing and the coauthor of the popular textbook Introduction to

Evolutionary Computing published by Springer. Tuning of evolutionary algorithms is of his primary research interest and the subject of his BIOMA presentation.

The BIOMA 2012 keynote speaker is Xin-She Young, a senior research scientist at National Physical Laboratory in Teddington, London, who is also the editor-in-chief of International Journal of Mathematical Modelling and Numerical Optimization. He focuses on metaheuristics for hard optimization problems and his talk illustrates their potentials in industrial applications using the newly proposed bat algorithm.

Theoretical and algorithmic studies presented at the conference include several new or improved bioinspired optimization algorithms, parameter tuning and control in evolutionary algorithms, exploitation and exploration measures, and dealing with premature algorithm convergence. Reports on applications come from domains such as waste management, resource allocation in communication networks, tracking of unmanned aerial vehicles, automated neural network design, text categorization, solving mathematical problems with genetic programming, generation of facial composites for criminal investigations, optimization of stock market investment strategies, and scheduling of flexible energy production and consumption.

BIOMA 2012 is sponsored by the Slovenian Research Agency. Technical sponsors of the conference are the World Federation on Soft Computing, the Slovenian Artificial Intelligence Society (SLAIS), and the Jožef Stefan Institute.

We are grateful to the conference sponsors, members of the program and organizing committees, the invited speakers, paper presenters and other participants for contributing their parts to the conference. We wish you an inspiring meeting and a pleasant stay in Bohinj.

Ljubljana, 10 May 2012

Bogdan Filipič and Jurij Šilc

Program Committee

Bogdan Filipič, *Chair*, Jožef Stefan Institute, Ljubljana, Slovenia
Jurij Šilc, *Chair*, Jožef Stefan Institute, Ljubljana, Slovenia
Thomas Bäck, Leiden Institute of Advanced Computer Science, The Netherlands
Christian Blum, Technical University of Catalonia, Barcelona, Spain
Sadok Bouamama, University of Carthage, Tunis, Tunisia
Janez Brest, University of Maribor, Slovenia
Dirk Büche, MAN Diesel & Turbo, Zürich, Switzerland
Carlos A. Coello Coello, CINVESTAV-IPN, Mexico
David W. Corne, Heriot-Watt University, Edinburgh, UK
Kusum Deep, Indian Institute of Technology Roorkee, India
Marco Dorigo, Université Libre de Bruxelles, Belgium
Rolf Drechsler, University of Bremen, Germany
Peter Korošec, Jožef Stefan Institute, Ljubljana, Slovenia
Barbara Koroušić Seljak, Jožef Stefan Institute, Ljubljana, Slovenia
Shih-Hsi "Alex" Liu, California State University, Fresno, USA
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
Nadia Nedjah, State university of Rio de Janeiro, Brasil
Gregor Papa, Jožef Stefan Institute, Ljubljana, Slovenia
Mario Pavone, University of Catania, Italy
Günter Rudolph, University of Dortmund, Germany
Franciszek Seredynski, Polish Academy of Sciences, Warsaw, Poland
El-Ghazali Talbi, University of Lille, France
Jim Tørresen, University of Oslo, Norway
Rasmus Kjar Ursem, Aarhus University, Denmark

Organizing Committee

Contributing Authors

Ana Maria de Almeida received his Ph.D. degree in Applied Mathematics/Computer Science from the University of Coimbra, Portugal, in 2004. She is currently a researcher at the Center for Informatics and Systems of the University of Coimbra. She is also an expert consultant at the IPN - Instituto Pedro Nunes, Association for Innovation and Development in Science and Technology and for EACI. Her research interests include modeling, combinatorial optimization, signal processing. She is a member of ACM, Apdio and SPM.

Maja Atanasijević-Kunc received Ph.D. (1997) degrees from the Faculty of Electrical Engineering, University of Ljubljana, Slovenia, where she is currently associate professor. Her research interests include modeling and simulation of dynamical systems and control systems analysis and design, specially of MIMO-systems.

Torben Bach Pedersen received his Ph.D. degree in Computer Science from Aalborg University, Denmark, in 2000. He is currently a professor at the Center for Data-Intensive Systems (Daisy) at Aalborg University. His research interests include business intelligence, data warehousing, and multidimensional databases, focusing on advanced application areas like energy, logistics, and Location-based mobile services. He is a member of ACM, IEEE, and IEEE CS.

Dražen Bajer received his M.Sc. degree in Computer Science from the Josip Juraj Strossmayer University of Osijek, Croatia, in 2010. He is currently a research and teaching assistant at the Josip Juraj Strossmayer University of Osijek, Faculty of Electrical Engineering. His research interests include applications of artificial intelligence methods and combinatorial optimization. He is a IEEE graduate student member.

Uroš Bole has a business background (MBA and various leadership positions in the engineering, retail, education and manufacturing industries). He became passionate about issues relating to the integration of analytics in organizations after joining a startup analytics consultancy in 2006. Through his involvement in analytics projects in the metallurgy, retail and entertainment industries he sensed that a major obstacle to wider acceptance of analytics in organizations is the difference in the way of seeing problems, solutions and communicating the ideas involved between business and analytics people. In 2008 he started a Ph.D. at Jožef Stefan International Postgraduate School, Ljubljana, Slovenia with the aim of proposing a methodology which would facilitate the integration of analytics in organizations.

Janez Brest received his Ph.D. degree in Computer Science from the University of Maribor, Slovenia, in 2000. He is currently a full professor at the University of Maribor, Faculty of Electrical Engineering and Computer Science. His research interests include evolutionary computing, artificial intelligence, optimization, programming languages, web-oriented programming, parallel and distributed computing research. He is a member of IEEE, and ACM.

Sergey Burakov received his B.Sc. degree in Applied Mathematics and Computer Science from the Siberian Federal University in 2007 and Ph.D. degree in System Analysis, Control and Data Processing from the Siberian State Aerospace University, Krasnoyarsk, Russia, in 2012. He is currently a research fellow at the Siberian State Aerospace University, Department of System Analysis and Operations Research. His research interests include computational intelligence and applications.

Fabio Caraffini received the M.Sc. degree in Computer and Electronic Engineering from the University of Perugia, Italy, in 2011. He is currently a Ph.D. student in Computer Science at the University of Jyväskylä, Finland. His research interests include computational intelligence optimization, robotics, and embedded systems.

Adam Cardenas is attending California State University, Fresno, USA. He is working on his B.Sc. in Computer Science and graduating in the Spring of 2012. Along with school, he is currently a Java developer at the software company called Bixly. In the Fall of 2012, he will be starting his Ph.D. in Computer Science at the University of Colorado Boulder.

Hamouda K. Chantar is a Ph.D. student at the Department of Computer Science, Heriot-Watt University, Edinburgh, UK. His research interests are in optimization algorithms, data mining, and applications in text analysis and document categorization, especially in the Arabic language.

Marko Corn is a Ph.D. student at Faculty of Electrical Engineering, University of Ljubljana, Slovenia. He is currently working at INEA d.o.o. company for automation, process control and manufacturing informatics in Ljubljana. His research interests include evolutionary computation, smart grids and process controls.

David Wolfe Corne is a professor of Computer Science at Heriot-Watt University, Edinburgh, UK. He leads the Intelligent Systems Lab, and his research interests span the fundamentals and applications of population based optimisation algorithms, machine learning algorithms, and, in particular, multiobjective optimization.

Péter Cserti received his M.Sc. degree in Computer Science from the University of Pannonia, Veszprem, Hungary, in 2010. He is currently a Ph.D. student at the University of Pannonia, Faculty of Information Technology. His research interests include machine learning, global optimizers and GPU based parallelization. He is a member of John von Neumann Computer Society.

Gregor Černe received his Ph.D. degree in Nuclear Engineering from University of Ljubljana, Slovenia, in 2001. Currently he is a project leader in department for Energy and Ecology in INEA d.o.o. He is working on the area of smart grids, energy managements systems and control of the energy devices. He is a member of the Slovenian smart grid platform association.

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

Donald Davendra received his Ph.D. degree in Technical Cybernetics from the Tomas Bata University in Zlín, Czech Republic, in 2009. He is currently an assistant professor at the Technical University of Ostrava, Faculty of Electrical Engineering and Computer Science. His research interests include evolutionary algorithms, differential evolution, CUDA technologies and theory of chaos.

Luca De Filippis is a graduate research assistant affiliated with the Department of Mechanics and Aerospace of Politecnico di Torino, Italy. He received his Ph.D. degree in 2012 on Path Planning and Collision Avoidance Algorithms for UAVs. His research interests include path planning and collision avoidance systems for unmanned vehicles, control-system design, space dynamics and control, optimization techniques and evolutionary algorithms, human supervising and human machine interfaces.

Erik Dovgan received his B.Sc. degree in Computer Science from the University of Ljubljana, Slovenia, in 2008. He is currently a research assistant at the Jožef Stefan Institute, Department of Intelligent Systems. His research interests include evolutionary algorithms, multiobjective optimization, and machine learning methods for pattern recognition.

Ágoston Endre Eiben is head of the Computational Intelligence Group at the VU University Amsterdam, The Netherlands. He is one of the European early birds of Evolutionary Computing, his first EC paper dates back to 1989. Since then he has published over 100 research papers, and he co-authored the first comprehensive introductory book on evolutionary computing (*Introduction to Evolutionary Computing*, Springer, 2003). His current research interests include parameter calibration, evolutionary swarm robotics, and evolutionary art.

Roxanne T. Evering received her M.Sc. degree in Modern Applications of Mathematics from the University of Bath, UK, in 2011, graduating with distinction. She currently works for CORDA as a graduate consultant, specialising in decision support services.

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

terests include stochastic optimization, evolutionary computation and intelligent data analysis. He published over 30 papers in international scientific journals and was a principle investigator in national and international projects dealing with production processes optimization, energy efficiency, and cultural heritage preservation. He serves as a member of editorial boards of several scientific journals and a programme committee member for the Genetic and Evolutionary Computation Conference (GECCO) and Congress on Evolutionary Computation (CEC).

Iztok Fister received his Ph.D. degree in Computer Science from the University of Maribor, Slovenia, in 2007. He is currently a teaching assistant at the University of Maribor, Faculty of Electrical Engineering and Computer Science. His research interests include computer architectures, programming languages, operational researches, artificial intelligence, evolutionary computing and swarm intelligence. He is a member of IEEE.

Iztok Fister Jr. received his B.Sc. degree in Computer Science from the University of Maribor, Slovenia, in 2010. He is currently a post graduate M.Sc. student at the University of Maribor, Faculty of Electrical Engineering and Computer Science. His research interests include evolutionary computing, swarm intelligence and programming languages. He is a Student member of IEEE.

Balázs Gaál received his Ph.D. degree in Computer Science from the University of Pannonia, Veszprem, Hungary, in 2010. He is currently an associate professor at the University of Pannonia, Faculty of Information Technology. His research interests include genetic algorithms, global optimization, medical information systems.

Pavel Galushin received his M.Sc. degree in System Analysis and Control from the Siberian State Aerospace University, Krasnoyarsk, Russia, in 2008 and Ph.D. degree in System Analysis, Control and Data Processing from the Siberian State Aerospace University in 2012. He is currently a research fellow at the Siberian State Aerospace University, Department of System Analysis and Operations Research. His research interests include pseudo-boolean optimization and effective evolutionary algorithms implementation.

Stuart J. Gibson received his Ph.D. degree in Physics from the University of Kent, UK, in 2006. He is currently lecturer in Forensic Science at the University of Kent, Faculty of Sciences. He is also a Director of Visionmetric Ltd, a company supplying facial identification software to police forces. His research interests centre on vision and image processing. He is a member of the British Machine Vision Association, the IEEE and the Institute of Physics.

Giorgio Guglieri is an associate professor of flight mechanics at Politecnico di Torino, Italy, where he belongs to the steering committee for the doctoral course in aerospace engineering. He is currently the quality manager for the EASA Part66 certification (Italian Aviation Authority - ENAC) for the Aerospace Engineering degree. Author of several articles and reviewer for several aerospace international journals. He is involved in several research activities on flight control system design, HMI interfaces design and tele-operations, UAV design, optimal path planning & collision avoidance, advanced multi-task PC-based HMI, optimal control design based on genetic algorithms (bounded by handling qualities requirements and mission constraints), simulation of spacecraft reentry vehicle dynamics, lunar landing, and rendez-vous/docking, including GNC modelling.

Muhannad Hijaze is a Ph.D. student at the Department of Computer Science, Heriot-Watt University, Edinburgh, UK. His research interests are in parallel computation, in particular the design of population based optimization algorithms for implementation over clusters of communicating processors.

Ronald Hochreiter received his Ph.D. degree in Business Informatics from the University of Vienna, Austria, in 2005. He is currently an assistant professor at the WU Vienna University of Economics and Business, Department of Finance, Accounting and Statistics. He is also a visiting professor at the University of Bergamo, Italy. His research interests include computational management science, business analytics and risk management. He is a member of ACM, INFORMS, ISI/IASC, OeGOR, and OeSG.

Jiří Hološka received his Ph.D. degree in Engineering Informatics from the Tomas Bata University in Zlín, Czech Republic, in 2012. His research interests include steganalysis by means of neural networks.

Giovanni Iacca received his Ph.D. degree in Computer Science from the University of Jyväskylä, Finland, in 2011. He is currently a post-doctoral researcher at INCAS3, Assen, The Netherlands, in the Systems & Controls research group. His research interests include evolutionary optimization, memetic computing, memory-saving algorithms, robotics, real-time systems, wireless sensor networks and distributed computing. He is a member of IEEE and IEEE CIS.

Giorgos Karafotias received his M.Sc. degree in Artificial Intelligence from Vrije Universiteit Amsterdam, The Netherlands, in 2010. He is currently a Ph.D. student at VU Amsterdam working with the Computational Intelligence Group. His research focuses on evolutionary algorithms, parameter calibration and self-adaptation and their applications to on-line adaptive systems.

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

György Kozmann received his Ph.D. degree in Computer Science from the University of Technology and Economics, Budapest, Hungary, in 1981. He is currently a professor at the University of Pannonia, Faculty of Information Technology. His research interests include medical informatics. He is a governing member of International Society of Electrocardiology and International Medical Informatics Association.

Erkki Laitinen received his Ph.D. degree in Applied Mathematics from the University of Jyväskylä, Finland, in 1989. He is a now a lecturer in Applied Mathematics at the Department of Mathematical Sciences of the Faculty of Science at the University of Oulu, and a docent in Computer Science at the Department of Mathematical Information Sciences at the University of Jyväskylä. His current research interests are in nonlinear optimization techniques, numerical analysis, modeling and simulation. He has published several papers in journals and conference proceedings,

and participated in many industrial projects dealing, among others, with optimal water spray control in the steel continuous casting process.

Shih-Hsi "Alex" Liu received his Ph.D. degree in Computer Science from the University of Alabama at Birmingham, USA, in 2007. He is currently an assistant professor at California State University, Fresno. His research interests include domain-specific languages, evolutionary computation, software product lines, and service-oriented computing. He is a member of ACM, IEEE, and UPE.

Goran Martinović received his Ph.D. degree in Computer Science from the University of Zagreb, Croatia, in 2004. He is currently an associate professor at the Josip Juraj Strossmayer University of Osijek, Faculty of Electrical Engineering. His research interests include distributed computer systems, real-time systems, artificial intelligence and medical informatics. He is a member of IEEE, ACM, IACIS, Cognitive Science Society, KOREMA and member of the IEEE SMC Technical Committee on Distributed Intelligent Systems.

Luca Masi received his M.Sc. degree in Aerospace Engineering from Università di Pisa, Italy, in 2009. He is currently Ph.D. student at the University of Strathclyde, Faculty of Aerospace & Mechanical Engineering, Glasgow, UK. His research interests include multi criteria decision making, bio-inspired single and multiobjective optimisation and space system engineering.

Marjan Mernik received the M.Sc. and Ph.D. degrees in Computer Science from the University of Maribor, Slovenia, in 1994 and 1998, respectively. He is currently professor of Computer Science at the University of Maribor. He is also visiting professor of Computer and Information Sciences at the University of Alabama at Birmingham, USA, and at the University of Novi Sad, Faculty of Technical Sciences, Serbia. His research interests include programming languages, compilers, domain-specific (modeling) languages, grammar-based systems, grammatical inference, and evolutionary computations. He is a member of the IEEE, ACM and EAPLS.

Ernesto Mininno received his first Ph.D. degree in Electrical Engineering from the Technical University of Bari, Italy, in 2007. He got a second Ph.D. in Computer Science from University of Jyväskylä, Finland,

in 2011. He is currently a Postdoctoral fellow of the Finnish Academy of Science at University of Jyväskylä. His research interests include evolutionary algorithms, metaheuristic methods, multi-objective multidisciplinary design optimization. He is an IEEE member since 2004.

Ferrante Neri received the M.Sc. and Ph.D. degrees in Electrical Engineering from the Technical University of Bari, Italy, in 2002 and 2007, respectively, and the Ph.D. degree in Computer Science from the University of Jyvskyl, Finland, in 2007. Since 2010 he is Adjunct Professor in Computational Intelligence at the University of Jyväskylä. Currently, he is a research fellow with the Academy of Finland at the Department of Mathematical Information Technology, University of Jyväskylä, Finland. His current research interests include computational intelligence optimisation and more specifically memetic computing, differential evolution, noisy and large scale optimisation, and compact and parallel algorithms.

Rui Ferreira Neves is a professor at Instituto Superior Técnico since 2005. He received the B.Sc. and Ph.D. degrees in Electrical and Computer Engineering from the Instituto Superior Técnico, Technical University of Lisbon, Portugal, in 1993 and 2001, respectively. In 2006 he joined Instituto de Telecomunicações (IT) as a research associate. His research activity deals with evolutionary computation applied to the financial markets, sensor networks, embedded systems and mixed signal integrated circuits. During his research activities he has collaborated/coordinated several EU and national projects.

Orlando O. A. A. N. Oliveira received his Ph.D. degree in Science from the University of Edinburgh, UK, in 1996. He is currently a professor at the University of Coimbra, Faculty of Science and Technology, Portugal. His research interests include non-perturbative techniques for quantum field theory, computational physics and mathematical physics.

Zuzana Oplatková received her Ph.D. degree in Technical Cybernetics from the Tomas Bata University in Zln, Czech Republic, in 2008. She is currently an assistant professor at the same university, Faculty of Applied Informatics. Her research interests include neural networks, evolutionary computation, symbolic regression, deterministic chaos and others.

Gregor Papa is a researcher at the Computer Systems Department, Jožef Stefan Institute, Ljubljana, Slovenia, and an assistant professor at the Jožef Stefan International Postgraduate School, Ljubljana. He received his M.Sc. and Ph.D. in Electrical Engineering from the University of Ljubljana 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.

José Matias Pinto is Ph.D. student at Instituto Superior Técnico. He received the B.Sc. degree in Electronics and Telecommunications Engineering from the Instituto Superior de Engenharia de Lisboa, Instituto Politécnico de Lisboa, Portugal, in 1995. In 2010 he joined Instituto de Telecomunicações as a research student. His research activity deals with evolutionary computation applied to the financial computing.

Michal Pluháček received his M.Sc. degree in Informatics from the Tomas Bata University in Zlín, Czech Republic, in 2011. He is currently a Ph.D. student at the Tomas Bata University in Zlín, Faculty of Applied Informatics. His research interests include evolutionary algorithms.

Ilpo Poikolainen received his M.Sc. degree from the University of Jyväskylä, Finland, in 2011. He is currently a Ph.D. student at the University of Jyväskylä, Faculty of Information Technology. His research interests include computational intelligence optimisation and more specifically memetic computing, differential evolution, noisy and large scale optimisation, and compact and parallel algorithms.

Martin B. Reed received his Ph.D. degree in Applied Mathematics from the University of Bath, UK, in 1980. He is currently a Lecturer at the University of Bath, Department of Mathematical Sciences. His research interests include numerical optimisation and finite element methods for geotechnical engineering.

Rui Salgueiro received his M.Sc. degree in Mathematics (specialization in Applied Mathematics) from the University of Coimbra, Portugal, in 2009. He is currently a Information Technology technician at the University of Coimbra and a junior researcher for CISUC - Center

for Informatics and Systems of the University of Coimbra. His research interests include algorithms and network optimization.

Eugene Semenkin received his Ph.D. degree in Computer Science from the University of Leningrad (Sankt Petersburg), Russia, in 1989 and D.Sc. degree in Engineering from the Siberian State Aerospace University, Krasnoyarsk, Russia, in 1997. He is currently a professor at the Siberian State Aerospace University, Faculty of Computer Science and Telecommunication. His research interests include complex systems modeling and optimization, computational intelligence and decisions support systems.

Maria Semenkina received her B.Sc. degree in Applied Mathematics from the Siberian Federal University in 2010 and M.Sc. degree in System Analysis and Control from the Siberian State Aerospace University, Krasnoyarsk, Russia, in 2012. She is currently a teaching assistant at the Siberian State Aerospace University, Department of Higher Mathematics. Her research interests include computational intelligence and applications.

Selmar Kagiso Smit is a former Ph.D. student from the Vrije Universiteit Amsterdam, The Netherlands, currently working at TNO Defense Safety and Security. He did his Ph.D. on parameter tuning of evolutionary algorithms and experimental methodology. His research interests include evolutionary algorithms, artificial life, model-based optimization, parameter tuning and parameter control.

Chris J. Solomon received his Ph.D. degree in Physics from the University of London, UK, in 1989. He is currently reader in Physics at the University of Kent, Faculty of Sciences. He is also a chief executive officer of Visionmetric Ltd, a company supplying facial identification software to police forces. His research interests centre on stochastic search methods and image processing. He is a member of the Optical Society of America and the IEEE.

Szabolcs Szondi is currently a bachelor student at the University of Pannonia, Veszprem, Hungary, at the Faculty of Information Technology. His research interests include stochastic optimizers, machine learning and GPU based parallelization.

Roman Šenkeřík received his Ph.D. degree in Technical Cybernetics from the Tomas Bata University in Zlín, Czech Republic, in 2008. He is currently an assistant professor at the Tomas Bata University in Zlín, Faculty of Applied Informatics. His research interests include evolutionary computation, neural networks and theory of chaos.

Laurynas Šikšnys received his M.Sc. degree in Computer Science from Aalborg University, Denmark, in 2009. He is currently a Ph.D. student at the University of Aalborg, Faculty of Engineering and Science. His research interests include energy data management, energy databases, and smart-grids.

Jurij Šilc received his Ph.D. in Electrical Engineering from the University of Ljubljana, Slovenia, in 1992. In 1980 he joined the Jožef Stefan Institute, where he is now a senior researcher. At the Institute, he served as the head of the Computer Architecture Laboratory from 1986 to 1994. He is presently the deputy head of the Computer Systems Department and an assistant professor at the Jožef Stefan International Postgraduate School, Ljubljana. His research interest include parallel processing, computer architectures, and bio-inspired optimization algorithms. He is a member of the IEEE.

Riste Škrekovski is an associate professor at the Department of Mathematics of the University of Ljubljana, Slovenia, where he also obtained his Ph.D. degree in Graph Theory. His research interest are various classical topics in graph theory as well as graph applications in chemistry and networks.

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

Tea Tušar received her M.Sc. degree in Computer Science from the University of Ljubljana, Slovenia, in 2007. She is currently a research assistant at the Department of Intelligent Systems of the Jožef Stefan Institute and a Ph.D. student at the Jožef Stefan International Post-

graduate School. Her research interests include evolutionary algorithms for singleobjective and multiobjective optimization, and machine learning methods for text processing and outlier detection. She is a student member of IEEE and a member of SLAIS.

Massimiliano Vasile received his Ph.D. degree in Aerospace Engineering from the Politecnico di Milano, Italy, in 2000. Since 2010, he is a reader in Aerospace Engineering at the University of Strathclyde, Glasgow, UK. His research interests include multi-objective optimisation, evolutionary algorithms, robust space mission design, asteroid deflection methods, orbital dynamics and trajectory optimisation. He is a member of AIAA, IEEE, IAF Space Power Committee.

István Vassányi received his Ph.D. degree in Computer Science from the Budapest University of Technology and Economics, Budapest, Hungary, in 2000. He is currently an associate professor at the University of Pannonia, Faculty of Information Technology. His research interests include data modeling, database technology, data mining and medical information systems.

Risto Vesanen received his M.Sc. degree in Applied Mathematics from the University of Oulu, Finland, in 2011 and he is a postgraduate student at the same university. He is now working in Metallurgy laboratory at the Department of Materials Science and Engineering at the Aalto University, Helsinki. His current work is related to developing and implementing algorithms for solving continuous casting problems. His current research interests are in steel solidification modeling and continuous casting process simulation.

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

Christoph Waldhauser received his M.A. degree in Political Science from the University of Vienna, Austria, in 2009. He is currently a Ph.D. candidate at the WU Vienna University of Economics and Business, Department of Finance, Accounting and Statistics. His research interests

include evolutionary computation, election forecasts and data mining. He is a member of ACM.

Matthieu Weber received his M.Sc. degree in Informatics from the Institut National des Sciences Appliquées of Lyon, France, in 1999 and his Ph.D. degree in Information Technology from the University of Jyväskylä, Finland, in 2010. He is currently working as a postdoctoral researcher at the Faculty of Information Technology of the University of Jyväskylä, Finland. His research interest are and evolutionary computing, and more specifically differential evolution, large scale problems and parallel algorithms.

Xin-She Yang received his Ph.D. degree in Applied Mathematics from the University of Oxford, then worked at Cambridge University for a few years and now is a Senior Research Scientist at National Physical Laboratory, Teddington, London, UK. He has authored/edited a dozen books and published more than 140 papers. He is the editor-in-chief of International Journal of Mathematical Modelling and Numerical Optimisation, serves as an editorial board member of several international journals, including Elsevier's Journal of Computational Science. He is also a vice chair of the IEEE CIS task force.

Ivan Zelinka received his Ph.D. degree in Technical Cybernetics from the Tomas Bata University in Zlín, Czech Republic, in 2001. He is currently a professor at the Technical University of Ostrava, Faculty of Electrical Engineering and Computer Science. His research interests include evolutionary computation, complex networks, bioinformatics and theory of chaos.

Federico Zuiani received his M.Sc. degree in Space Engineering from the Politecnico di Milano, Italy, in 2009. Since December 2009, he is a Ph.D. candidate in Aerospace Engineering at the University of Glasgow, UK. He is also a visiting researcher at the University of Strathclyde, Glasgow. His research interests include multi-objective optimisation, evolutionary algorithms, orbital dynamics and trajectory optimisation.

I

INVITED CONTRIBUTIONS

TUNING EVOLUTIONARY ALGORITHMS AND TUNING EVOLUTIONARY ALGORITHM CONTROLLERS

Giorgos Karafotias, Selmar Kagiso Smit, Ágoston Endre Eiben
Faculty of Sciences, Vrije Universiteit Amsterdam, The Netherlands
g.karafotias@vu.nl; selmar@gmail.com; gusz@cs.vu.nl

Abstract In the last few years various powerful parameter tuning methods have been presented for evolutionary algorithms. These can nicely handle the tuning problem, delivering good parameter values that remain constant during an EA run. In this paper we conduct a feasibility study into using one of these tuning methods to solve the control problem. The idea is to specify a parameterized mechanism to control EA parameters on-the-fly and optimize the parameters of the control mechanism with our tuning algorithm. We perform proof-of-concept experiments with 3 parameters of an evolution strategy on 5 different objective functions and find that on all problems the EA using the tuned controller is better or at least as good as the EA whose parameters were tuned directly.

Keywords: Evolutionary algorithms, Parameter control, Parameter tuning.

1. Introduction

When defining an evolutionary algorithm (EA) one needs to configure various settings: choose components (such as variation and selection mechanisms) and set numeric values (e.g. the probability of mutation or the tournament size). These configurations largely influence performance making them an important aspect of algorithm design. The field of evolutionary computing (EC) traditionally distinguishes two approaches for setting parameter values [5, 8]. *Parameter tuning*, where parameter values are fixed in the initialization stage and do not change while the EA is running. *Parameter control*, where parameter values are given an initial value when starting the EA and undergo changes while the EA is running. The capability of parameter control to use adequate parameter values in different stages of the search is an advantage, because the run of an EA is an intrinsically dynamic process. It

is intuitively clear –and for some EA parameters theoretically proven– that different values may be optimal at different stages of the evolution. This implies that the use of static parameters is inherently inferior to changing parameter values on-the-fly.

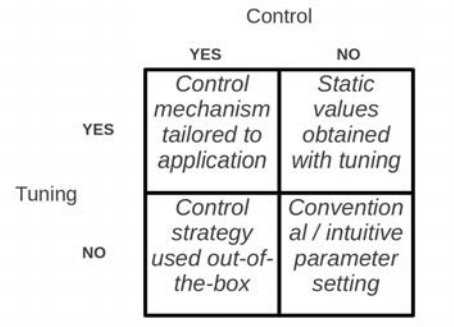


Figure 1. Tuning vs control.

The distinction between tuning and control can be lifted if we consider the control mechanism as an integral part of the EA. In this case the EA and its parameter control mechanism (that may be absent) are considered as one entity. This composed entity may or may not be tuned before being applied to a new problem. The resulting matrix of four options is shown in Fig. 1. The combinations in the top row have the advantage of enhanced performance at the cost of the tuning effort [17]. The options in the left column offer the benefits of time varying parameter values mentioned above with a trade-off of increased complexity.

Here, we focus on the top left cell of the matrix, i.e. control that is tailored to a specific problem. This approach combines on-line parameter adjustment (control) and off-line configuration (tuning). The EA incorporates a parameter control mechanism, but this controller itself has certain parameters that can be configured for a problem through an off-line tuning process. This tunable control mechanism is proposed as an alternative to tuning static parameter values: *instead of spending time to search for good EA parameter values, the same effort can be spent to make a good calibration of a mechanism that performs on-line control of these parameters.* Such a method has the advantage of enhanced performance thanks to varying parameter values, and the possibility to be tuned to a specific problem. The objective of this paper is to introduce such a generic mechanism and to answer the following questions:

- Is such an approach viable?

- What is the added value of this approach, when compared to static parameter values?

- On what kind of feedback from the search process should such a parameter control mechanism base its decisions?

To answer these questions we introduce a generic framework that helps understand the decisions behind the design of a control mechanism and conduct experiments with three parameters of an evolution strategy.

2. Related Work

Parameter control is an increasingly popular topic in the field of evolutionary algorithms [14]. The outline of the most commonly used methods is quite similar: one of the parameter values is altered based on some specific evidence. Most often these methods are designed for specific parameters. The most popular parameter-specific control methods focus on mutation probability [9], mutation step size [21] and operator selection [26] but methods also exist for the selection pressure [27], the population-size [25], the fitness function [15] and the encoding [22].

The mutation step size of evolution strategies was one of the first parameters that was considered for control. In the field of Evolution Strategies, controlling the mutation step size was one of the key ingredients to its success. Analysis of the simple corridor and sphere problems in large dimensions in the early 70's led to Rechenberg's 1/5th success rule that changes the mutation step size, based on the feedback about its success. Not much later, self-adaptation of *sigma* was introduced. Self-adaptive control is not based on such deterministic rules to change parameter values. Instead, it encodes the σ into the chromosomes and co-evolves them with the problem parameters. In general, this is much less interpretable since the immediate effect on the parameter-values is not very clear. Furthermore, because in each run these values are 're-learned', the information gained about the appropriate parameter values for different situations, cannot be reused in another run or another EA.

The control of the population size of evolutionary algorithms has been previously examined in several works. One approach taken to the population size parameter was to eliminate it all together by introducing the notion of individuals' lifetimes as was done in GAVaPS [1]. Another approach was to approximate a good population size on-line: the parameter-less GA [11] automatically performs an on-line search for a good population size by creating and running populations of progressively larger size. Direct control of the population size parameter has also been studied deterministically [12], adaptively [6] and self-adaptively [4].

Some generic control methods for numeric parameters also exist. In [28] an adaptive mechanism is proposed that works in alternating epochs, first evaluating parameter values in a limited set and then applying them probabilistically. In the end of every such pair of epochs the set of possible parameter values is updated according to some heuristic rule. In Lee and Takagi [13] an adaptive control mechanism based on fuzzy logic is proposed. Instantiation of the rule set of the controller is achieved through an off-line calibration process using a GA. Lee and Takagi concluded that such an approach was very beneficial, and led to a much better performance than using the fixed parameter values. However, the fixed values used in this study were the ones commonly used at that time, based on the early work of DeJong, rather than found using parameter tuning. A two-layer approach to generic numeric control is presented in [3] and [16]: the lower layer adaptively controls EA parameters driven by an upper level that enforces a user-defined schedule of diversity or exploration-exploitation balance (though these are not parameters per se). The algorithm in [16] includes a built-in learning phase that calibrates the controller to the EA and problem at hand by associating parameter values to diversity and mean fitness using random samples. In [3], the lower control level is an adaptive operator selection method that scores operators according to the diversity-fitness balance they achieve as compared to a balance dictated by the upper level user defined schedule. However, neither of the two make a comparison against static parameter-values found using parameter tuning. There are a few studies that compare control strategies with tuned parameter values [10, 18, 19, 20], however these typically arrive to a conclusion opposite to that of [3, 13, 16].

Extensive literature reviews on parameter control can be found in [7] and [2].

3. Parameter Control Road-map

In this section we present a simple framework for parameter control mechanisms. The purpose of this framework is not to provide any theoretical grounding or proofs but to serve as a road-map that helps in designing and positioning one's mechanism.

We define a parameter control mechanism as a combination of three components:

1 A choice of parameters (i.e. *what* is to be controlled).

2 A set of observables that will be the input to the control mechanism (i.e. what *evidence* is used).

3 A technique that will map observables to parameter values (i.e. *how* the control is performed).

These components are briefly described in the following paragraphs. However, they are based on the definition of the state of an evolutionary algorithm, which is therefore introduced first.

EA State. We define the state S_{EA} of an evolutionary algorithm as:

$$S_{EA} = \{G, \bar{p}, \mathcal{F}\}, \tag{1}$$

where G is the set of all the genomes in the population, $\bar{p}$ is the vector of current parameter values, and $\mathcal{F}$ is the fitness function.

A triple S_{EA} uniquely specifies the state of the search process for a given evolutionary algorithm (the design and specific components of the EA need not be included in the state since they are the same during the whole run) in the sense that S_{EA} fully defines the search results so far and is the only observable factor that influences the search process from this point on (though not fully defining it, given the stochastic nature of EA operators). Time is not part of S_{EA} as it is irrelevant to the state itself; it introduces an artificial uniqueness and a property that is unrelated to the evolution. Of course, state transitions are not deterministic.

3.1 Parameters

The starting point when designing a control mechanism is the parameter to be controlled (as well as choices such as when and how often the parameter is updated). The importance of various parameters and the effect or merit of controlling each of them are subjects that will not be treated here (we refer to [2]). Instead, here we will only distinguish between *numeric* (e.g. population size, crossover probability) and *symbolic* (e.g. recombination operator) parameters.

3.2 Observables

The observables are the values that serve as inputs to the controller's algorithm. Each observable must originate from the current state S_{EA} of the EA since, as defined above, it is the only observable factor defining how the search will proceed from this point on.

However, the raw data in the state itself are unwieldy: if we were to control based on state S_{EA} directly, that would imply that the control algorithm should be able to map every possible S_{EA} to proper parameter values. Consequently, preprocessing is necessary to derive some useful abstraction, similar to the practise of dataset preprocessing in the field

of data mining. We define such an observable derivation process as the following pipeline:

$$Source \rightarrow (Digest) \rightarrow (Derivative) \rightarrow (History).$$

Parentheses denote that steps can be bypassed.

i. *Source*: As stated above, the source of all observables is the current state of the EA, i.e. the set of all genomes, the current parameter values and the fitness function.

ii. *Digest*: A function $D(S_{EA}) = v$ that maps an EA state to a value, e.g. best fitness or population diversity.

iii. *Derivative*: Instead of using directly a value v we might be more interested in its speed or acceleration (e.g. to make the observable independent to the absolute values of v or to determine the effect of the previous update as the change observed in the most recent cycle).

iv. *History*: The last step in defining an observable is maintaining a history of size W of the value received from the previous step. This step includes a decision on the sliding window size W and the definition of a function $F_H(v_1, v_2, ..., v_W)$[1] that, given the last W values, provides a final value or vector (e.g. the minimum value, the maximum increase between two consecutive steps, the whole history as is etc.).

The above observable derivation is meant to be a conceptual framework and not an implementation methodology. For example, the current success ratio (in the context of Rechenberg's 1/5 rule) can in theory be derived from a state S_{EA} by applying the selection and variation operators to G and calculating the fitnesses of the results though obviously that would be a senseless implementation.

3.3 Mapping Technique

Any technique that translates a vector of observable values to a vector of parameter values can be used as an mapping for the control mechanism, e.g. a rule set, an ANN or a function are all valid candidates. The choice of the proper technique seems to bear some resemblance to choosing an appropriate machine learning technique given a specific task or dataset. Whether EA observables display specific characteristics that make certain biases and representations more suitable is a question that needs to be investigated. In any case, it is obvious that given the type of

parameter controlled (i.e. numeric or nominal) different techniques are applicable.

Here we distinguish between two main categories of control techniques, regardless the algorithm and representation used, based on a fundamental characteristic of the controller: whether it is static or it adapts itself to the evolutionary process.

i. *Static*: A static controller remains fixed during the run, i.e. given the same observables input it will always produce the same parameter values output. In other words, the values produced only depend on the current observables input:

$$\vec{p} = c(\vec{o}) \text{ and } \vec{o_1} = \vec{o_2} \Rightarrow c(\vec{o_1}) = c(\vec{o_2}),$$

where $\vec{o} \in O$, $\vec{p} \in P$ are the vectors of observables and parameter values respectively and $c : O \mapsto P$ is the mapping of the controller.

ii. *Dynamic*: A dynamic controller changes during the run, i.e. the same observables input can produce different parameter values output at different times. This implies that the controller is stateful and that the values produced depend on both the current observables input and the controller's current state:

$$\vec{p} = c_p(\vec{o}, S_C) \text{ and } S_C^{t+1} = c_S(\vec{o_t}, S_C^t),$$

where $\vec{o} \in O$, $\vec{p} \in P$ are the vectors of observables and parameter values respectively, $S_C \in \mathcal{S}$ is the state of the controller and $c_p : O \times \mathcal{S} \mapsto P$, $c_S : O \times \mathcal{S} \mapsto \mathcal{S}$ are the mappings of the controller.

According to this classification, a time-scheduled mechanism is a trivial case of a dynamic controller; it maintains a changing state (a simple counter) but is "blind" to the evolutionary process since it does not use any observables. It should be noted that we do not consider control mechanisms necessarily as separate and distinct components, e.g. we classify self-adaptation in ES as a dynamic controller since it implicitly maintains a state influenced by the evolutionary process.

4. Generic and Tunable Control

The motivation of this paper is to provide a proof of the concept that a tunable and generic controller can have an advantage over tuning static parameter values. For this purpose we need to compare the performance of the two approaches. To facilitate this comparison, though, we need concrete representatives of the two methods. For the case of tuned static parameters this is trivial, however, we require a tunable

generic controller as a representative of tunable control. We describe such a controller in this section; notice that the described controller is merely one possible implementation used for the experiments and we do not suggest it as an 'optimal' controller design.

The controller is required to be generic enough to be able to handle any numeric parameter, capable of problem-specific calibration and compatible with any evolutionary algorithm. Below we present a design for an EA-independent, parameter-independent and tunable controller based on the framework presented in Section 3.

4.1 Parameters

Any numeric parameter can be controlled and the value of a controlled parameter is updated in each generation. To control a parameter, a range $[min, max]$ of values must be specified which will be the output range of the controller.

4.2 Observables

The observables that we currently use as input to the controller, are based on the current parameter values, diversity and fitness. However, in principle any observable that provides useful information about the current state of the algorithm can be used.

Fitness based. We use two observables based on fitness. Both use the best fitness digest of the current population: $f_B = max\mathcal{F}(g), g \in G$. The simpler fitness-based observable f_N is just the normalized value of the best fitness found in the current population: $f_N = norm(f_B), f_N \in [0, 1]$. This observable is only applicable when the bounds of the fitness function are known. The second fitness-based observable Δf provides information about the change of the best fitness observed in the last generations. Instead of using the derivative step to describe this change we use a history of length W and the history function $f_H(f_B^1, ..., f_B^W) = \frac{f_B^W - f_B^{W/2}}{f_B^W - f_B^1}$. We choose to use this history setting to measure change instead of the derivative step so to make the controller robust to shifting and stretching of the fitness landscape.

To summarize the two fitness-based observables are:

$$f_N = norm(f_B), \quad f_N \in [0, 1],$$

$$\Delta f = \frac{f_B^W - f_B^{W/2}}{f_B^W - f_B^1}, \quad \Delta f \in [0, 1], \quad W = 100,$$

where $f_B = max\mathcal{F}(g), g \in G$. In order to be able to use Δf from the beginning, the value of W grows from 2 to 100 with each generation. Notice that these observables are not used together but are two alternatives.

Diversity based. Diversity is observed using the Population Diversity Index (PDI) [24] as the digest function and bypassing derivatives and history. The Population Diversity Index is a measure that uses an entropy like approach for measuring the genotypic diversity in a real-valued population. It can identify clusters, and always returns a value between 0 and 1 indicating a fully converged (0), a uniformly distributed (1) population, or anything in between.

Current parameter values. The current values of the controlled parameters $\vec{p_c}$ are also observed and input to the controller when the Δf observable is used. The reason is that if changes in fitness are observed then changes in the parameter value should be output, thus the old value must be available. Each value in $\vec{p_c}$ corresponds to the current value of a controlled parameter and is normalized to $[0, 1]$ (this is possible because the range of all controlled parameters is known as described in 4.1).

Given two choices for observing fitness and the choice to include the diversity observable or not yields four sets of observables: $\{f_N\}$, $\{f_N, PDI\}$, $\{\Delta f, \vec{p}\}$ and $\{\Delta f, PDI, \vec{p}\}$.

4.3 Control Method

As a control method we chose a neural network (NN) as a generic method for mapping real valued inputs to real valued outputs. We use a simple feed-forward network without a hidden layer. The structure of the nodes is fixed and the weights remain static during an EA run and are set by the off-line tuning process. Thus, the calibration of the controller consists of finding appropriate weights for a given problem or application type.

All inputs are, as defined above, in the range $[0, 1]$. The weights are tuned $w \in [-1, 1]$. The activation of the neurons is a sigmoid function, horizontally compressed so as to reach very close to its asymptotes given a domain of $[0, 1]$.

When multiple parameters are controlled simultaneously a separate NN controls each parameter. If the current parameter values $\vec{p_c}$ are used as input, then each NN uses only the value of the corresponding parameter as input. This may seem oversimplifying considering parameter interaction but it also simplifies the controller and significantly decreases the number of weights that need to be calibrated.

5. Experimental Setup

We conducted experiments to evaluate the tunable control approach (represented by the generic controller described in the previous section) and how it compares with the tuned static approach. Using the same EA, solving the same problems and spending the same effort for off-line tuning/calibration, we compared the performance achieved when keeping parameter values static to the performance achieved when using the controller.

The evolutionary algorithm used, the relevant parameters, the method used for tuning and the overall experimental setup are described in the following subsections.

5.1 Evolutionary Algorithm and Parameters

The EA used, is a simple $(\mu + \lambda)$ ES with Gaussian mutation. It has no recombination and uses uniform random parent selection. This EA has three parameters: the population size μ, the generation gap $g = \frac{\lambda}{\mu}$ and the mutation step size σ. We run five distinct experiment sets controlling each parameter alone, controlling μ and g together (considering their obvious interaction) and controlling all parameters together. In the experiments where one of the parameters is not controlled/tuned then it is always set to a value chosen based on conventional EC knowledge and intuition (see Table 1). These values also serve as initial values when a parameter is controlled.

Table 1. Default values of parameters when not controlled or tuned.

μ	10
g	7
σ	(Sphere) 0.1, (Corridor) 0.1, (Ackley) 0.6, (Rosenbrock) 0.04, (Fletcher & Powell) 0.07, (Rastrigin) 0.1

5.2 Off-Line Tuning

Both controller calibrations and static values are found through an off-line tuning process using Bonesa [23]. Bonesa is an iterative model-based search procedure based on an intertwined searching and learning loop. The search loop is a generate-and-test procedure that iteratively generates new vectors, pre-assesses their quality using a surrogate model

of the performance landscape, and finally tests the most promising vectors by executing an algorithm run with these specific values. In its turn, the learning loop uses the information obtained about the quality of the tested vectors to update the model of the performance surface. Furthermore, it uses a kernel filter to reduce the noise caused by the stochasticity of the algorithm.

5.3 Overall Setup

For each problem, the four versions of the controller (using the four different observables sets described in 4.2) are calibrated using Bonesa. The same tuner, and effort is also spent on finding good static values for the problem, which adheres to classical parameter tuning. The resulting EAs (with controlled and static parameters) are run 100 times to validate the outcomes, to fairly compare their performances. The same experiment is repeated for controlling each of the parameters individually, μ together with g and all together. The experimental setup is summarized in Table 2.

Table 2. Experimental setup.

EA	$(\mu + \lambda)$-ES with: Gaussian mutation, no recombination and uniform random parent selection, limit of 10000 evaluations, default parameter values shown in Table 1
Parameters	$\mu \in [1, 100]$, $g \in [0, 14]$, $\sigma \in [0, 1]$ (individually, μ with g and all together)
Instantiation	Bonesa with a budget of 3000 tests to calibrate weights $w_i \in [-1, 1]$
Problems	Sphere, Corridor, Ackley, Rosenbrock, Rastrigin, Fletcher & Powell
Controller observables	$\{f_N\}$, $\{f_N, PDI\}$, $\{\Delta f, \vec{p}\}$, $\{\Delta f$, PDI, $\vec{p}\}$

6. Results

Results are shown in Tables 3 to 7. For all tables, bold values denote a performance that is not significantly worse than the best on that specific problem, while underlined values denote performance that is significantly better than the EA with static parameter values for the specific problem. In all cases, significance is verified using a t-test with 95% confidence.

Tuning and solving for the Corridor problem failed in all cases, so it will be completely disregarded in the subsequent analysis.

6.1 Performance

Tables 3, 4 and 6 show that attempting to control solely the population size, the generation gap or their combination was unsuccessful. But, controlling only σ (Table 5) was successful for the controlled EA. It outperforms the static EA on three out of five problems, while not being significantly worse in the other two.

Table 3. Performance results for μ. For each column, bold values denote performance not significantly different to the best and underlined values denote performance significantly better than static.

	Sph	Crdr	Rsbk	Ack	Rsg	F&P
(f_N,D)	**0.09607**	**9.303**	4.868	**5.75**	36.98	7602
(f_N)	**0.09607**	**9.303**	4.868	**5.75**	36.77	6731
(Δf, D, $\vec{p}$)	**0.09607**	**9.303**	4.868	**5.75**	36.98	6408
(Δf,$\vec{p}$)	**0.09607**	**9.303**	4.868	**5.75**	38.97	5528
Static	**0.09901**	**9.303**	**4.329**	**5.776**	**34.29**	**1335**

Table 4. Performance results for g. For each column, bold values denote performance not significantly different to the best and underlined values denote performance significantly better than static.

	Sph	Crdr	Rsbk	Ack	Rsg	F&P
(f_N,D)	**0.1009**	**9.303**	**5.115**	**5.716**	**55.15**	1.113e+04
(f_N)	**0.1009**	**9.303**	**5.115**	**5.716**	**55.15**	1.065e+04
(Δf, D, $\vec{p}$)	**0.1009**	**9.303**	6.142	**5.752**	**54.4**	1.065e+04
(Δf,$\vec{p}$)	**0.1009**	**9.303**	**5.115**	**5.716**	**55.32**	1.065e+04
Static	**0.1003**	**9.303**	**5.226**	**5.778**	79.35	**5770**

However, controlling all parameters (Table 7) at the same time, seems to be the most beneficial; the controlled EA outperforms the static in four problems while it is not significantly worse in the fifth. Even better (although not shown here), on most of the problems, the controller using all parameters outperform those for a single parameter or μ, g combination.

Table 5. Performance results for σ. For each column, bold values denote performance not significantly different to the best and underlined values denote performance significantly better than static.

	Sph	Crdr	Rsbk	Ack	Rsg	F&P
(f_N,D)	0.04848	**9.303**	**7.473**	0.26	29.06	6848
(f_N)	**0.01609**	**9.303**	**6.995**	**0.1085**	**23.32**	**5622**
(Δf, D, $\vec{p}$)	0.0528	**9.303**	**8.05**	**0.3153**	27.08	6089
(Δf,$\vec{p}$)	0.0353	**9.303**	**8.111**	0.617	**25.85**	8238
Static	0.05745	**9.303**	**7.066**	3.684	37.4	**4710**

Table 6. Performance results for μ,g. For each column, bold values denote performance not significantly different to the best and underlined values denote performance significantly better than static.

	Sph	Crdr	Rsbk	Ack	Rsg	F&P
(f_N,D)	0.09869	**9.303**	6.89	**5.633**	38.26	5841
(f_N)	**0.09382**	**9.303**	4.789	**5.591**	38.66	6432
(Δf, D, $\vec{p}$)	0.09869	**9.303**	6.89	**5.633**	38.19	4299
(Δf,$\vec{p}$)	**0.09382**	**9.303**	4.789	**5.756**	**36.22**	3465
Static	**0.09475**	**9.303**	**3.834**	**5.575**	**34.08**	**762.3**

Table 7. Performance results for all. For each column, bold values denote performance not significantly different to the best and underlined values denote performance significantly better than static.

	Sph	Crdr	Rsbk	Ack	Rsg	F&P
(f_N,D)	1.102	**9.303**	26.81	3.337	29.28	7824
(f_N)	0.007457	**9.303**	11.54	**0.5488**	**23.99**	5999
(Δf, D, $\vec{p}$)	5.387	**9.303**	82.6	18.5	36.52	2.055e+05
(Δf,$\vec{p}$)	**0.005169**	**9.303**	**7.358**	2.275	30.48	**2028**
Static	0.03565	**9.303**	**7.025**	2.024	35.9	2828

In general, for the EA and problems tested, the controller performs better or at least as good as the tuned, but static parameter values.

6.2 Parameter Behavior

Analyzing the behavior of the controlled parameters provides some important insight.

Controlling μ and g: As the performance results show (Tables 3, 4 and 6) there is no improvement when controlling μ and g. In fact, in most cases, the calibration of control creates controllers that maintain constant parameter values. When controlling μ alone, controllers almost always maintain a constant value, either set to 1 (Sphere, Ackley and Rosenbrock) or simply linearly increasing to its maximum (Rastrigin). Similarly, when controlling g alone, in almost all cases values are kept constant either to its minimum so that each generation has only one offspring (Sphere, Ackley, Rosenbrock) or its maximum (Fletcher & Powell, Rastrigin). In the few cases where parameter values vary according to the inputs there is no improvement in performance. These findings could mean that controlling these parameters for the specific EA and problems does not have an intrinsic value or that it is very difficult for the tuning process to find good controller weights.

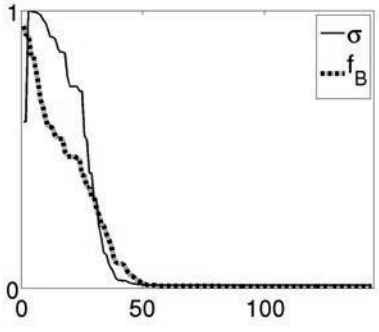


Figure 2. Example of the behavior of parameter σ over time (Ackley problem with $\{f_N\}$). All values are normalized between 0 and 1.

Controlling σ: A much more interesting behavior is observed in the values of σ. In most cases, σ shows an increase in the beginning and, subsequently, drops and stabilizes or oscillates around a value. Such a gradual decrease is a good strategy for climbing a hill and approximating an optimum. An example of this σ behavior is shown in Fig. 2. Of course, a preferable behavior would be to re-increase σ when a population is stuck to a local optimum. Such a situation can be detected based on the diversity and Δf observables used, however, this desired behavior does not occur in our results. A different control strategy is observed in Fig. 3: σ oscillates rapidly while the range of this oscillation seems

related to current diversity. Despite its apparent oddity, this controller performs better than tuned static.

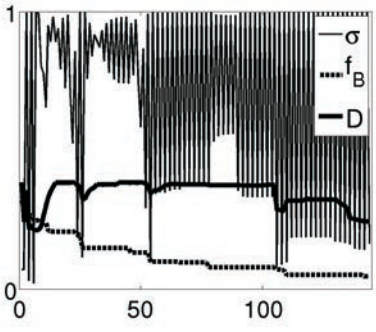


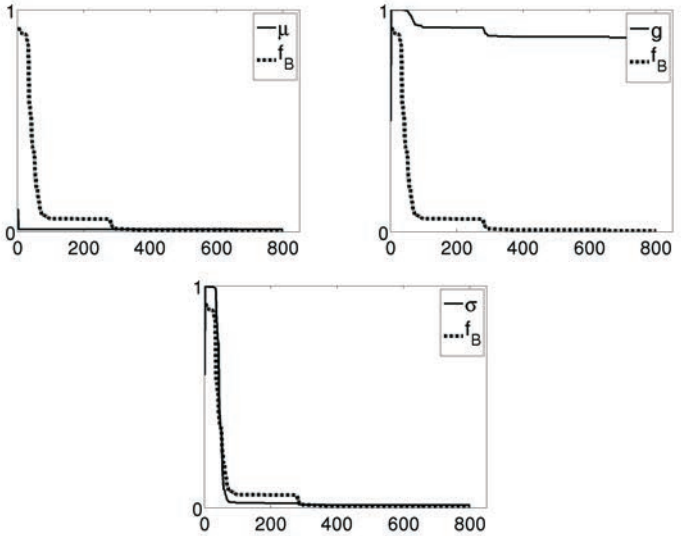
Figure 3. Example of the behavior of parameter σ over time (Rastrigin problem with $\{\Delta f$, D, $\vec{p}\}$). All values are normalized between 0 and 1.

Controlling all combined: Combined control of all parameters produces the best results with calibrated controllers generally demonstrating a more complex behavior. Values of μ and g still often remain constant, especially for the Rosenbrock and Sphere problems (this is not necessarily a drawback considering that these two problems are unimodal and can be solved with a (1+1)ES). Control of σ shows a behavior similar to what is described in the previous paragraph. Examples of parameters' behavior are illustrated in Fig. 4. In one case μ is set to a constant value, however, g and σ are controlled according to the inputs. In the first case, σ shows one of the two behaviors described in the previous paragraph while in the second it simply follows diversity.

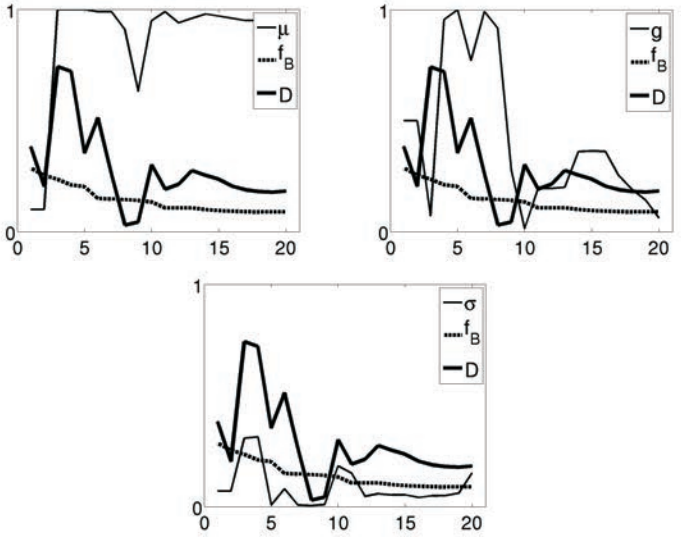
6.3 Observables

The best results are acquired using only fitness based observables, either $\{f_N\}$ or $\{\Delta f$, $\vec{p}\}$. Between these two we cannot distinguish a clear winner.

Though choosing between f_N or Δf is mostly a matter of feasibility (calculating normalized fitness is not possible if the bounds of the fitness values are not known beforehand), including a diversity observable or not, is a more fundamental question. Contrary to our initial expectations, adding diversity to the input does not yield better performance even for multimodal problems (including the irregular and asymmetric Fletcher & Powell function). In fact, keeping all other factors fixed, using diversity as an input produces a significant improvement only in 2 out of 50 cases.



(a) Ackley problem with $\{f_N\}$



(b) Rastrigin problem with $\{\Delta f,\ \mathrm{D},\ \vec{p}\}$

Figure 4. Example of parameter behavior over time with combined control. All values are normalized between 0 and 1.

6.4 Discussion on Selection

Three important points were observed in the results and analysis of this section:

- the failure to calibrate a meaningful controller for μ,
- the absence of control strategies that increase σ late in the run to escape local optima,
- and the fact that using diversity as an observable does not improve performance even for multimodal problems.

A plausible assumption is that these points might be due to the survivor selection used by the ES in the experiments ("plus" selection). All above points are related to maintaining a diverse population, either by accommodating enough individuals, by performing enough exploration when necessary or by screening the current status of diversity. However, using a "plus" selection could negate an effort to increase diversity since new and "diverse" individuals would be of inferior fitness and, thus, discarded while newly grown populations would be taken over by the existing best individuals.

7. Conclusions and Future Work

Based on our results, the questions stated in the introduction can be easily answered. The main conclusion that can be drawn is that the generic approach taken in this paper is viable and fruitful. In contrast to previous work, we were able to find a problem-tailored control mechanism that outperforms the tuned (but static) parameter values for each of the problems. More specific, either using the best fitness value directly, or the combination of δ fitness and the current parameter values, outperforms the tuned but static parameter values in most problems.

Inevitably, this conclusion depends on the experimenters design decisions. In our case, the most important factors (beyond the underlying algorithm itself) are the following:

- The observables chosen as inputs to the control strategy.
- The parameters to be controlled
- The technique that maps a vector of observable values to a vector of parameter values

Changing either of these can, in principle, lead to a different conclusion. With respect to the observables chosen, we can conclude that these

indeed highly influence the results. Remarkably, adding diversity as an input appears to have hardly any added value for controlling σ.

Regarding the future, we expect more studies along these ideas, exploring the other possible implementations of the most important factors. Most enticing is the possibility of applying it to other algorithms and applications. If these could be controlled with the same ease, this opens up the possibilities for a problem tailed and high performing algorithm for any particular problem.

Notes

1. Notice that indices have no relation to time but merely indicate a sequence of W elements.

References

[1] J. Arabas, Z. Michalewicz, and J. J. Mulawka. Gavaps - a genetic algorithm with varying population size. In *Proc. International Conference on Evolutionary Computation*, pages 73–78, 1994.

[2] K. De Jong. Parameter setting in EAs: a 30 year perspective. In F. Lobo, C. Lima, and Z. Michalewicz, editors, *Parameter Setting in Evolutionary Algorithms*, pages 1–18. Springer, 2007.

[3] G. di Tollo, F. Lardeux, J. Maturana, and F. Saubion. From adaptive to more dynamic control in evolutionary algorithms. In *Proc. 11th European Conference on Evolutionary Computation in Combinatorial Optimization*, pages 130–141, 2011.

[4] Á. E. Eiben, M. Schut, and A. de Wilde. Is self-adaptation of selection pressure and population size possible? - a case study. *Lect. Notes Comput. Sc.*, 4193:900–909, 2006.

[5] Á. E. Eiben, R. Hinterding, and Z. Michalewicz. Parameter control in evolutionary algorithms. *IEEE T. Evolut. Comput.*, 3(2):124–141, 1999.

[6] Á. E. Eiben, E. Marchiori, and V. A. Valko. Evolutionary algorithms with on-the-fly population size adjustment. *Lect. Notes Comput. Sc.*, 3242:41–50, 2004.

[7] Á. E. Eiben, Z. Michalewicz, M. Schoenauer, and J. E. Smith. Parameter control in evolutionary algorithms. In Lobo et al. [14], pages 19–46.

[8] Á. E. Eiben and S. K. Smit. Parameter tuning for configuring and analyzing evolutionary algorithms. *Swarm and Evolutionary Computation*, 1(1):19–31, 2011.

[9] T. C. Fogarty. Varying the probability of mutation in the genetic algorithm. In *Proc. 3rd International Conference on Genetic Algorithms*, pages 104–109, 1989.

[10] G. Francesca, P. Pellegrini, T. Stützle, and M. Birattari. Off-line and on-line tuning: A study on operator selection for a memetic algorithm applied to the qap. *Proc. 11th European Conference on Evolutionary Computation in Combinatorial Optimization*, 2011.

[11] G. R. Harik, F. G. Lobo, and D. E. Goldberg. The conpact genetic algorithm. In *IEEE T. Evolut. Comput.*, 3(4):287–297, 1999.

[12] V. K. Koumousis and C. P. Katsaras. A saw-tooth genetic algorithm combining the effects of variable population size and reinitialization to enhance performance. *IEEE T. Evolut. Comput.*, 10(1):19–28, 2006.

[13] M. A. Lee and H. Takagi. Dynamic control of genetic algorithms using fuzzy logic techniques. In *Proc. 5th International Conference on Genetic Algorithms*, pages 76–83, 1993.

[14] F. Lobo, C. Lima, and Z. Michalewicz, editors. *Parameter Setting in Evolutionary Algorithms*. Springer, 2007.

[15] M. Majig and M. Fukushima. Adaptive fitness function for evolutionary algorithm and its applications. In *Proc. International Conference on Informatics Research for Development of Knowledge Society Infrastructure*, pages 119–124, 2008.

[16] J. Maturana and F. Saubion. On the design of adaptive control strategies for evolutionary algorithms. *Lect. Notes Comput. Sc.*, 4926:303–315, 2008.

[17] V. Nannen, S. K. Smit, and Á. E. Eiben. Costs and benefits of tuning parameters of evolutionary algorithms. *Lect. Notes Comput. Sc.*, 5199:528–538, 2008.

[18] M. Pedersen and A. Chipperfield. Simplifying particle swarm optimization. *Appl. Soft Comput.*, 10(2):618–628, 2010.

[19] M. E. H. Pedersen and et al. Parameter tuning versus adaptation: proof of principle study on differential evolution. Technical report, Hvass Laboratories, 2008.

[20] P. Pellegrini, T. Stützle, and M. Birattari. Off-line vs. on-line tuning: A study on max–min ant system for the TSP. *Lect. Notes Comput. Sc.*, 6234:239–250, 2010.

[21] I. Rechenberg. *Evolutionstrategie: Optimierung Technisher Systeme nach Prinzipien des Biologischen Evolution*. Fromman-Hozlboog Verlag, Stuttgart, 1973.

[22] N. N. Schraudolph and R. K. Belew. Dynamic parameter encoding for genetic algorithms. *Mach. Learn.*, 9:9–21, 1992.

[23] S. K. Smit and Á. E. Eiben. Multi-problem parameter tuning using bonesa. In J. Hao, P. Legrand, P. Collet, N. Monmarché, E. Lutton, and M. Schoenauer, editors, *Artificial Evolution*, pages 222–233, 2011.

[24] S. K. Smit, Z. Szláavik, and Á. E. Eiben. Population diversity index: a new measure for population diversity. In *Proc. Genetic and Evolutionary Computation Conference*, pages 269–270, 2011.

[25] R. Smith and E. Smuda. Adaptively resizing populations: Algorithm, analysis and first results. *Complex Systems*, 9(1):47–72, 1995.

[26] W. M. Spears. Adapting crossover in evolutionary algorithms. In *Proc. 4th Annual Conference on Evolutionary Programming*, pages 367–384, 1995.

[27] P. Vajda, Á. E. Eiben, and W. Hordijk. Parameter control methods for selection operators in genetic algorithms. *Lect. Notes Comput. Sc.*, 5199:620–630, 2008.

[28] Y.-Y. Wong, K.-H. Lee, K.-S. Leung, and C.-W. Ho. A novel approach in parameter adaptation and diversity maintenance for genetic algorithms. *Soft Comput.*, 7(8):506–515, 2003.

METAHEURISTIC OPTIMIZATION WITH APPLICATIONS: DEMONSTRATION VIA BAT ALGORITHM

Xin-She Yang
National Physical Lab, Teddington, London, UK
xin-she.yang@npl.co.uk

Abstract Metaheuristics have become powerful and popular for solving tough optimization problems with a wide range of industrial applications. More than a dozen new algorithms such as PSO, firefly algorithm and cuckoo search have appeared in the last twenty years. In this paper, we will first discuss the role of exploitation and exploration in algorithms and how to best balance these two key components. Then, we use a new bat algorithm as an example to show how such optimal combination can be achieved. Finally, we apply the bat algorithm to solve design benchmarks in engineering and industry.

Keywords: Algorithm, Bat algorithm, Cuckoo search, Firefly algorithm, Metaheuristics, Particle swarm optimization.

1. Introduction

Many design optimization problems in engineering and industry are highly nonlinear under stringent constraints, and thus often NP-hard. Therefore, to find optimal solutions to such optimization problems is usually very challenging if not impossible. In the last two decades, more than a dozen new algorithms such as particle swarm optimization, differential evolution, ant and bee algorithms, bat algorithm, harmony search, firefly algorithm and cuckoo search have appeared and they have shown great potential in solving tough engineering optimization problems [8, 9, 16, 18].

Metaheuristic algorithms, especially those based on swarm intelligence, form an important part of contemporary global optimization algorithms [4, 7, 11, 16]. Good examples are particle swarm optimization [11] and firefly algorithm [16, 17]. They work remarkably efficiently and have many advantages over traditional, deterministic methods and al-

gorithms, and thus they have been applied in almost all area of science, engineering and industry [8, 18]. There are more than a dozen swarm-based algorithms using the so-called swarm intelligence. For a detailed introduction, please refer to [12, 18].

2. Key Components in Metaheuristics

Metaheuristics can be considered as an efficient way to produce acceptable solutions by trial and error to a complex problem in a reasonably practical time. The complexity of the problem of interest makes it impossible to search every possible solution or combination, the aim is to find good feasible solution in an acceptable timescale. There is no guarantee that the best solutions can be found, and we even may not know whether an algorithm will work and why if it does work, though we may know the basic components that can help to work. The idea is to have an efficient but practical algorithm that will work most the time and is able to produce good quality solutions. Among the found quality solutions, it is expected some of them are nearly optimal, though there is often no guarantee for such optimality.

However, many metaheuristics can typically have global convergence properties, and thus they usually can find the global optimality in practice within a relatively limited number of function iterations or evaluations. This makes it particularly suitable for metaheuristic algorithms to solve global optimization problems. For example, firefly algorithm [17] can deal with multimodal optimization problems naturally due to its ability to automatically subdivide the swarming population into subgroups and each group can search each mode locally and in parallel. On the other hand, cuckoo search uses not only a search technique with good convergence but also a randomization technique with more efficient Lévy flights [20].

In principle, for a metaheuristic algorithm to be efficient, it has to have some special capabilities. One of such is to be able to generate new solutions that can usually be more likely to improve the previous/existing solutions and also be able to cover most important search areas where the global optimum may lie. Another capability is that an algorithm should be able to escape any local optimum so that it cannot be stuck in a local mode. A good combination may lead to good efficiency under appropriate conditions, and this often requires to balance two important components of any metaheuristics: exploration and exploitation. However, this itself is an unresolved optimization task.

The main components of any metaheuristic algorithms are: intensification and diversification, or exploitation and exploration [1, 4, 16, 18].

Diversification means to generate diverse solutions so as to explore the search space on the global scale, while intensification means to focus on the search in a local region by exploiting the information that a current good solution is found in this region. This is in combination with the selection of the best solutions.

2.1 Balance of Exploration and Exploitation: Empirical Observations

Exploration often uses randomization [4, 16], which enables an algorithm to have the ability to jump out of any local optimum so as to explore the search space globally. Randomization can also be used for local search around the current best if steps are limited to a local region. When the steps are large, randomization can explore the search space on a global scale. Fine-tuning the right amount of randomness and balancing local search and global search are crucially important in controlling the performance of any metaheuristic algorithm.

Randomization techniques can be a very simple method using uniform distributions and/or Gaussian distributions, or more complex methods as those used in Monte Carlo simulations. They can also be more elaborate, from Brownian random walks to Lévy flights [20].

Exploitation is the use of local knowledge of the search and solutions found so far so that new search moves can concentrate on the local regions or neighborhood where the optimality may be close; however, this local optimum may not be the global optimality. Exploitation tends to use local information such as gradients, the shape of the mode such as convexity, and the history of the search process. A classic technique is the so-called hill-climbing which uses the local gradients or derivatives intensively.

Empirical knowledge from observations and simulations of the convergence behaviour of common optimization algorithms suggests that exploitation tends to increase the speed of convergence, while exploration tends to decrease the convergence rate of the algorithm. On the other hand, too much exploration increases the probability of finding the global optimality, while strong exploitation tends to make the algorithm being trapped in a local optimum. Therefore, there is a fine balance between the right amount of exploration and the right degree of exploitation. Despite its importance, there is no practical guideline for this balance. So essentially each algorithm uses different degree of exploitation and exploration, often far from optimal [4]. Some algorithms may have intrinsically better balance among these two important com-

ponents that other algorithms, that is one of the reasons why they may perform betters [16, 18].

2.2 Theoretical Results

Even there is no guideline in practice, some preliminary work on the very limited cases exists in the literature and may provide some insight into the possible choice of parameters so as to balance these components [2, 3, 15]. Intermittent search strategies concern an iterative strategy consisting of a slow phase and a fast phase [2, 3]. Here the slow phase is the detection phase by slowing down and intensive, static local search techniques, while the fast phase is the search without detection and can be considered as an exploration technique. For example, the static target detection with a small region of radius a in a much larger region b where $a \ll b$ can be modelled as a slow diffusive process in terms of random walks with a diffusion coefficient D.

Let τ_a and τ_b be the mean times spent in intensive detection stage and the time spent in the exploration stage, respectively, in the 2D case [3]. The diffusive search process is governed by the mean first-passage time satisfying the following equations

$$D\nabla_r^2 t_1 + \frac{1}{2\pi\tau_a}\int_0^{2\pi}[t_2(r) - t_1(r)]d\theta + 1 = 0, \tag{1}$$

$$\boldsymbol{u}\cdot\nabla_r t_2(r) - \frac{1}{\tau_b}[r_2(r) - t_1(r)] + 1 = 0, \tag{2}$$

where t_1 and t_2 are mean first-passage times during the search starting from slow and fast stages, respectively, and $\boldsymbol{u}$ is the search speed [3].

After some lengthy mathematical analysis, the optimal balance of these two stages can be estimated as

$$r_{\text{optimal}} = \frac{\tau_a}{\tau_b^2} \approx \frac{D}{a^2}\frac{1}{[2 - \frac{1}{\ln(b/a)}]^2}. \tag{3}$$

Assuming that the search steps have a uniform velocity u at each step on average, the minimum times required for each phase can be estimated as

$$\tau_a^{\min} \approx \frac{D}{2u^2}\frac{\ln^2(b/a)}{[2\ln(b/a) - 1]} \tag{4}$$

and

$$\tau_b^{\min} \approx \frac{a}{u}\sqrt{\ln(b/a) - \frac{1}{2}}. \tag{5}$$

When $u \to \infty$, these relationships lead to the above optimal ratio of two stages.

It is worth pointing out that the above result is only valid for 2D cases, and there is no general results for higher dimensions, except in some special 3D cases [2]. Now let us use this limited results to help choose the possible values of algorithm-dependent parameters in bat algorithm [19], as an example.

3. Bat Algorithm

A new bat-inspired algorithm was formulated in 2010 by Yang [19], which were based on the fascinating echolocation characteristics of microbats. Bats are the only mammals with wings and they also have advanced capability of echolocation. It is estimated that there are about 996 different species which account for up to 20% of all mammal species. Their size ranges from tiny bumblebee bats (of about 1.5 to 2 g) to the giant bats with wingspan of about 2 m and weight up to about 1 kg. Microbats typically have forearm length of about 2.2 to 11 cm. Most bats uses echolocation to a certain degree; among all the species, microbats are a famous example as microbats use echolocation extensively while megabats do not [14].

If we idealize some of the echolocation characteristics of microbats, we can develop various bat-inspired algorithms or bat algorithm [19].

- All bats use echolocation to sense distance, and they also 'know' the difference between food/prey and background barriers in some magical way;
- Bats fly randomly with velocity $\boldsymbol{v}_i$ at position $\boldsymbol{x}_i$ with a frequency $f_{\min}$, varying wavelength λ and loudness A to search for prey. They can automatically adjust the wavelength (or frequency) of their emitted pulses and adjust the rate of pulse emission $r \in [0, 1]$, depending on the proximity of their target;
- Although the loudness can vary in many ways, we assume that the loudness varies from a large (positive) A_0 to a minimum constant value $A_{\min}$.

Another obvious simplification is that no ray tracing is used in estimating the time delay and three dimensional topography. Though this might be a good feature for the application in computational geometry, however, we will not use this feature, as it is more computationally extensive in multidimensional cases. In addition to these simplified assumptions, we also use the following approximations, for simplicity. In general the frequency f in a range $[f_{\min}, f_{\max}]$ corresponds to a range of wavelengths $[\lambda_{\min}, \lambda_{\max}]$.

3.1 Bat Motion

For the bats in simulations, we have to define the rules how their positions $\boldsymbol{x}_i$ and velocities $\boldsymbol{v}_i$ in a d-dimensional search space are updated. The new solutions $\boldsymbol{x}_i^t$ and velocities $\boldsymbol{v}_i^t$ at time step t are given by

$$f_i = f_{\min} + (f_{\max} - f_{\min})\beta, \tag{6}$$

$$\boldsymbol{v}_i^t = \boldsymbol{v}_i^{t-1} + (\boldsymbol{x}_i^{t-1} - \boldsymbol{x}_*)f_i, \tag{7}$$

$$\boldsymbol{x}_i^t = \boldsymbol{x}_i^{t-1} + \boldsymbol{v}_i^t, \tag{8}$$

where $\beta \in [0, 1]$ is a random vector drawn from a uniform distribution. Here $\boldsymbol{x}_*$ is the current global best location (solution) which is located after comparing all the solutions among all the n bats at each iteration t. As the product $\lambda_i f_i$ is the velocity increment, we can use f_i (or λ_i) to adjust the velocity change while fixing the other factor λ_i (or f_i), depending on the type of the problem of interest. In our implementation, we will use $f_{\min} = 0$ and $f_{\max} = O(1)$, depending on the domain size of the problem of interest. Initially, each bat is randomly assigned a frequency which is drawn uniformly from $[f_{\min}, f_{\max}]$.

For the local search part, once a solution is selected among the current best solutions, a new solution for each bat is generated locally using random walk

$$\boldsymbol{x}_{\text{new}} = \boldsymbol{x}_{\text{old}} + \boldsymbol{\epsilon}\, A^t, \tag{9}$$

where $\boldsymbol{\epsilon}$ is a random number vector drawn from $[-1, 1]$, while $A^t = <A_i^t>$ is the average loudness of all the bats at this time step.

The update of the velocities and positions of bats have some similarity to the procedure in the standard particle swarm optimization, as f_i essentially controls the pace and range of the movement of the swarming particles. To a degree, BA can be considered as a balanced combination of the standard particle swarm optimization and the intensive local search controlled by the loudness and pulse rate.

3.2 Loudness and Pulse Emission

Furthermore, the loudness A_i and the rate r_i of pulse emission have to be updated accordingly as the iterations proceed. As the loudness usually decreases once a bat has found its prey, while the rate of pulse emission increases, the loudness can be chosen as any value of convenience. For simplicity, we can also use $A_0 = 1$ and $A_{\min} = 0$, assuming $A_{\min} = 0$ means that a bat has just found the prey and temporarily stop emitting any sound. Now we have

$$A_i^{t+1} = \alpha A_i^t, \quad r_i^{t+1} = r_i^0[1 - \exp(-\gamma t)], \tag{10}$$

where α and γ are constants.

3.3 Choices of Parameters

In fact, α is similar to the cooling factor of a cooling schedule in the simulated annealing. For any $0 < \alpha < 1$ and $\gamma > 0$, we have

$$A_i^t \to 0, \quad r_i^t \to r_i^0, \quad \text{as} \quad t \to \infty. \tag{11}$$

In the simplest case, we can use $\alpha = \gamma$, and we have used $\alpha = \gamma = 0.9$ in our simulations.

If we use the simple, isotropic random walks for local exploration, then we have

$$D \approx \frac{s^2}{2}, \tag{12}$$

where s is the step length with a jump during a unit time interval or each iteration step. From Eq. (3), the optimal ratio of exploitation and exploration in a special case of $b \approx 10a$ becomes

$$\frac{\tau_a}{\tau_b^2} \approx 0.2. \tag{13}$$

In case of $b/a \to \infty$, we have $\tau_a/\tau_b^2 \approx 1/8$. which implies that more times should spend on the exploration stage. It is worth pointing out that the naive guess of 50-50 probability in each stage is not the best choice. More efforts should focus on the exploration so that the best solutions found by the algorithm can be globally optimal with possibly the least computing effort.

In the case studies to be described below, we have used the bat algorithm to find the optimal solutions to two design benchmarks. If we set $\tau_b = 1$ as the reference timescale, we found that the optimal ratio is between 0.18 to 0.24, which are roughly close to the above theoretical result. This may imply that bat algorithm has an intrinsic ability of balancing exploration and exploitation close to the true optimality.

4. Applications

There are a wide range of design benchmarks, and it is not possible to include even a good fraction of these benchmarks in a short paper. Therefore, we will use a few well-selected case studies to demonstrate how metaheuristic algorithms perform for nonlinear global optimization problems in engineering design.

4.1 Standing-Wave Function

Let us first use a multimodal test function to see how to find the fine balance between exploration and exploration in an algorithm for a given

task. A standing-wave test function can be a good example [18]

$$f(\boldsymbol{x}) = 1 + \Big\{ \exp[-\sum_{i=1}^{d} (\frac{x_i}{\beta})^{10}] - 2\exp[-\sum_{i=1}^{d} x_i^2] \Big\} \cdot \prod_{i=1}^{d} \cos^2 x_i, \quad (14)$$

which is multimodal with many local peaks and valleys. It has a unique global minimum at $f_{\min} = 0$ at $(0, 0, ..., 0)$ in the domain $-20 \le x_i \le 20$ where $i = 1, 2, ..., d$ and $\beta = 15$. In this case, we can estimate that $R = 20$ and $a \approx \pi/2$, this means that $R/a \approx 12.7$, and we have in the case of $d = 2$

$$\tau_{\text{optimal}} \approx \frac{1}{2[2 - 1/\ln(R/a)]^2} \approx 0.19. \quad (15)$$

This indicate that the algorithm should spend about 80% of its computational effort on global explorative search, and about 20% of its effort on local intensive search.

For bat algorithm, we have used $n = 15$ and 1000 iterations. We then count the number functional evolutions at local modes and global long-distance moves, so as to calculate the fraction or ratio p of local moves to global moves. A set of 25 numerical experiments have been carried out and the results are summarized in Table 1.

Table 1. Variations of p and its effect on the solution quality.

p	0.4	0.3	0.2	0.1	0.05
$f_{\min}$	8.9e-11	1.1e-12	2.3e-14	7.7e-12	8.1e-11

This table clearly shows that $p \approx 0.2$ provides the optimal balance of local exploitation and global exploration, which is consistent with the theoretical estimation.

Though there is no direct analytical results for higher dimensions, we can expect that more emphasis on global exploration is also true for higher dimensional optimization problems. Let us look two design benchmarks.

4.2 Welded Beam Design

The so-called welded beam design is a standard test problem for constrained design optimization tasks [13, 5, 10]. The problem has four design variables: the width w and length L of the welded area, the depth h and thickness h of the main beam. The objective is to minimize the overall fabrication cost, under the appropriate constraints of shear

stress τ, bending stress σ, buckling load P and maximum end deflection δ. In short, the problem can be written as

$$\text{minimise} \quad f(\boldsymbol{x}) = 1.10471w^2L + 0.04811dh(14.0 + L), \tag{16}$$

subject to

$$\begin{array}{l} g_1(\boldsymbol{x}) = w - h \le 0, \quad g_2(\boldsymbol{x}) = \delta(\boldsymbol{x}) - 0.25 \le 0, \\ g_3(\boldsymbol{x}) = \tau(\boldsymbol{x}) - 13,600 \le 0, \quad g_4(\boldsymbol{x}) = \sigma(\boldsymbol{x}) - 30,000 \le 0, \\ g_5(\boldsymbol{x}) = 0.10471w^2 + 0.04811hd(14 + L) - 5.0 \le 0, \\ g_6(\boldsymbol{x}) = 0.125 - w \le 0, \quad g_7(\boldsymbol{x}) = 6000 - P(\boldsymbol{x}) \le 0, \end{array} \tag{17}$$

where

$$\begin{array}{ll} \sigma(\boldsymbol{x}) = \frac{504{,}000}{hd^2}, & Q = 6000(14 + \frac{L}{2}), \\ D = \frac{1}{2}\sqrt{L^2 + (w+d)^2}, & J = \sqrt{2}\, wL[\frac{L^2}{6} + \frac{(w+d)^2}{2}], \\ \delta = \frac{65{,}856}{30{,}000hd^3}, & \beta = \frac{QD}{J}, \\ \alpha = \frac{6000}{\sqrt{2}wL}, & \tau(\boldsymbol{x}) = \sqrt{\alpha^2 + \frac{\alpha\beta L}{D} + \beta^2}, \\ P = 0.61423 \times 10^6 \ \frac{dh^3}{6}(1 - \frac{d\sqrt{30/48}}{28}). & \end{array} \tag{18}$$

The simple limits or bounds are $0.1 \le L, d \le 10$ and $0.1 \le w, h \le 2.0$.

This is a nonlinear global optimization problem with nonlinear constraints. We can use BA to find its optimal solution. In order to do some parametric studies, we varied the parameters r, A, α and γ so that we can tune the ratio of local moves to global moves, and we found that when this ratio is about 0.2, the convergence rate is highest, as observed from simulations. With such settings, we have obtained the following optimal solution $\boldsymbol{x}_* = (w, L, d, h) = (0.2057296397, 3.4704886656, 9.0366239104, 0.2057296398)$ with

$$f_{\min}(\boldsymbol{x}*) = 1.7248523086. \tag{19}$$

This solution is essentially the same as the solution obtained by Cagnina et al. [5]

$$f_* = 1.724852 \quad \text{at} \quad (0.205730, 3.470489, 9.036624, 0.205729). \tag{20}$$

4.3 Speed Reducer Design

Optimal design of a speed reducer or a gearbox is another benchmark design problem with seven design variables [5, 10], including the face width (b), module of the teeth (h), the number of teeth on pinion (z), the length (L_1) of the first shaft between bearing, the length (L_2) of the second shaft between between bearings, the diameter (d_1) of the first

shaft, and the diameter (d_2) of the second shaft. The main objective is to minimize the total weight of the speed reducer, subject to 11 constraints such as bending stress, deflection and various limits on stresses in shafts. This optimization problem can be written as [10]

$$f(b,h,z,L_1,L_2,d_1,d_2) = 0.7854bh^2(3.3333z^2+14.9334z-43.0934)$$
$$-1.508b(d_1^2+d_2^2)+7.4777(d_1^3+d_2^3)+0.7854(L_1d_1^2+L_2d_2^2), \qquad (21)$$

subject to

$$g_1 = \frac{27}{bh^2z}-1\le 0, \quad g_2 = \frac{397.5}{bh^2z^2}-1\le 0, \quad g_3 = \frac{1.93L_1^3}{hzd_1^4}-1\le 0,$$
$$g_4 = \frac{1.93L_2^3}{hzd_2^4}-1\le 0, \quad g_5 = \frac{1}{110d_1^3}\sqrt{(\frac{745L_1}{hz})^2+16.9\times 10^6}-1\le 0,$$
$$g_6 = \frac{1}{85d_2^3}\sqrt{(\frac{745L_2}{hz})^2+157.5\times 10^6}-1\le 0, \quad g_7 = \frac{hz}{40}-1\le 0,$$
$$g_8 = \frac{5h}{b}-1\le 0, \quad g_9 = \frac{b}{12h}-1\le 0, \quad g_{10} = \frac{1.5d_1+1.9}{L_1}-1\le 0,$$
$$g_{11} = \frac{1.1d_2+1.9}{L_2}-1\le 0. \qquad (22)$$

In addition, the simple bounds are $2.6\le b\le 3.6, 0.7\le h\le 0.8$, $17\le z\le 28, 7.3\le L_1\le 8.3$, $7.8\le L_2\le 8.3, 2.9\le d_1\le 3.9$, and $5.0\le d_2\le 5.5$. z must be integers.

This design benchmark is a mixed variable optimization problem because z only takes the values of integers. As the number of integer variables are small (just one in this case), we can use the standard branch-and-bound method to deal with the integer constraint.

For this case study, we used the same setting as obtained in the previous case study for welded beam design. Then, the best solutions obtained by the bat algorithm with $n = 25$ after 5000 iterations are

$$b = 3.5,\ h = 0.7,\ z = 17,\ L_1 = 7.3,\ L_2 = 7.8,$$
$$d_1 = 3.34336445,\ d_2 = 5.285350625, \quad f\min = 2993.7495888, \qquad (23)$$

which are better than $f_* = 2996.348165$ by others [5].

We have seen that, for both benchmark problems, BA has found the optimal solutions which are either better than or the same as the solutions found so far in the literature.

5. Discussions

Optimal balance of local exploitation and global exploration is key to any metaheuristic algorithm, and the efficiency of an algorithm is largely controlled by this factor. In fact, search for such optimal balance is itself a challenging optimization problem which remains unsolved. In general,

there is no generic methodology that can find such optimal balance for a given class of algorithms. Consequently, such is done mainly by trial and error and the methods used in the literature are often algorithm-specific, and thus algorithm-dependent parameters can also be problem-specific. This may be linked with the fact that no-free-lunch theorems can be valid for some class of problems, though free lunches are possible [1, 6]

Even with limited results in the literature, they are only valid for small scale problems and for lower dimensions.

For higher-dimensional problems, even limited results do not exist. One possible extension is to use extrapolation to get an estimate. Based on the results on 2D and 3D cases [3], we can estimate that for any d-dimensional cases $d \geq 3$

$$\frac{\tau_1}{\tau_2} \sim O\Big(\frac{D}{a^2}\Big), \quad \tau_m \sim O\Big(\frac{b}{V}(\frac{b}{a})^{d-1}\Big), \tag{24}$$

where τ_m the mean search time or average number of iterations. This extension may not be good news for higher dimensional problems, as the mean number of function evaluations to find optimal solutions can increase exponentially as the dimensions increase. However, in practice, we do not need to find the guaranteed global optimality, we may be satisfied with suboptimality, and sometimes we may be lucky to find such global optimality even with a limited/fixed number of iterations. This may indicate there is a huge gap between theoretical understanding and the observations as well as run-time behaviour in practice. More studies are highly needed to address these important issues.

6. Conclusions

Nature-inspired metaheuristic algorithms have gained popularity, which is partly due to its ability of dealing with nonlinear global optimization problems. In this paper, we have first highlighted the importance of key components of exploration and exploitation in metaheuristics, and then provided a simple, yet still practical estimate for the ratio of search times or efforts of exploitation and exploration stages. Then, we extended the recently proposed bat algorithm to study two well-known design benchmarks in engineering applications. Then, we confirmed that bat algorithm can indeed provide a good balance of exploitation and exploration, which leads to excellent results.

References

[1] A. Auger and O. Teytaud. Continuous lunches are free plus the design of optimal optimization algorithms. *Algorithmica*, 57(1):121–146, 2010.

[2] O. Bénichou, C. Loverdo, M. Moreau, and R. Voituriez. Two-dimensional intermittent search processes: An alternative to Lévy flight strategies. *Phys. Rev.*, E74:020102(R), 2006.

[3] O. Bénichou, C. Loverdo, M. Moreau, and R. Voituriez. Intermittent search strategies. *Review of Modern Physics*, 83(1):81–129, 2011.

[4] C. Blum and A. Roli. Metaheuristics in combinatorial optimisation: Overview and conceptural comparision. *ACM Comput. Surv.*, 35(3):268–308, 2003.

[5] L. C. Cagnina, S. C. Esquivel, and C. A. Coello Coello. Solving engineering optimization problems with the simple constrained particle swarm optimizer. *Informatica*, 32(3):319–326, 2008.

[6] D. Corne and J. Knowles. Some multiobjective optimizers are better than others. In *Proc. IEEE Congress on Evolutionary Computation*, volume 4, pages 2506–2512, 2003.

[7] M. Dorigo and T. Stütle. *Ant Colony Optimization*. MIT Press, 2004.

[8] C. A. Floudas and P. M. Pardolos. *Encyclopedia of Optimization*, 2nd Edition. Springer, 2009.

[9] Z. W. Geem. *Music-Inspired Harmony Search Algorithm: Theory and Applications*. Springer, 2009.

[10] A. H. Gandomi, X. S. Yang, and A. H. Alavi. Cuckoo search algorithm: a metaheuristic approach to solve structural optimization problems. *Eng. Comput.*, DOI 10.1007/s00366-011-0241-y, in press, 2011.

[11] J. Kennedy and R. C. Eberhart. Particle swarm optimization. In *Proc. IEEE International Conference on Neural Networks*, volume 4, pages 1942–1948, 1995.

[12] R. S. Parpinelli and H. S. Lopes. New inspirations in swarm intelligence: a survey. *Int. J. Bio-Inspired Computation*, 3(1):1–16, 2011.

[13] K. Ragsdell and D. Phillips. Optimal design of a class of welded structures using geometric programming. *J. Eng. Ind.*, 98(3):1021–1025, 1976.

[14] P. Richardson. *Bats*. Natural History Museum, London, 2008). Also: P. Richardson. The secrete life of bats. `http://www.nhm.ac.uk`.

[15] M. F. Shlesinger. Random searching. *J. Phys. A: Math. Theor.*, 42(43):434001, 2009.

[16] X. S. Yang. *Nature-Inspired Metaheuristic Algorithms*. Luniver Press, 2008.

[17] X. S. Yang. Firefly algorithms for multimodal optimisation. *Lect. Notes Comput. Sc.*, 5792:169–178, 2009.

[18] X. S. Yang. *Engineering Optimization: An Introduction with Metaheuristic Applications*. John Wiley & Sons, 2010.

[19] X. S. Yang. A new metaheuristic bat-inspired algorithm. In *Proc. Nature Inspired Cooperative Strategies for Optimization*, pages 65–74, 2010.

[20] X. S. Yang and S. Deb. Cuckoo search via Lévy flights. In *Proc. World Congress on Nature & Biologically Inspired Computing*, pages 210–214, 2009.

II

THEORY AND ALGORITHMS

REVOLUTIONARY ALGORITHMS

Ronald Hochreiter, Christoph Waldhauser
Institute for Statistics and Mathematics, WU Vienna University of Economics and Business, Austria
{ronald.hochreiter; christoph.waldhauser}@wu.ac.at

Abstract The optimization of dynamic problems is both widespread and difficult. When conducting dynamic optimization, a balance between reinitialization and computational expense has to be found. There are multiple approaches to this. In parallel genetic algorithms, multiple subpopulations concurrently try to optimize a potentially dynamic problem. But as the number of sub-population increases, their efficiency decreases. Cultural algorithms provide a framework that has the potential to make optimizations more efficient. But they adapt slowly to changing environments. We thus suggest a confluence of these approaches: revolutionary algorithms. These algorithms seek to extend the evolutionary and cultural aspects of the former to approaches with a notion of the political. By modeling how belief systems are changed by means of revolution, these algorithms provide a framework to model and optimize dynamic problems in an efficient fashion.

Keywords: Cultural algorithm, Dynamic optimization, Genetic algorithm.

1. Introduction

In evolutionary computing, the treatment of dynamic problems is troublesome. Dynamic problems are characterized by their constant changing of the optimum of a target function sought to be optimized. Unfortunately, many problems in the real world are dynamic: optimal routing solutions for mail delivery as new pieces are arrive [6], finding the best path through a volatile landscape [11], portfolio optimization while markets are trading [13], election forecasting while new polling stations are being counted [9]. These are a few examples of dynamic problems. Clearly, if the update frequency of any such problem is low, traditional evolutionary methods can be used to find any number of successive optima, simply by restarting the optimization process on the basis of new data. However, as the update frequency increases, this ap-

proach becomes more and more impractical. Once the data is updated at an interval that is smaller than the time frame needed to complete the optimization, a breakdown point is reached and other ways of obtaining the successive optima need to be derived.

A possibility of postponing the advent of the breakdown point is the application of algorithms that can be parallelized [3]. By executing the optimization process on multiple nodes concurrently, the time required for an optimization can be reduced by a factor almost proportionally to the number of nodes. In practice, this is often realized by using multiple populations in a genetic algorithm. The drawback of these parallel algorithms is that the quality of the overall solution suffers, as multiple smaller populations also limit the gene pool for any instance of that algorithm. Thus, the risk of becoming trapped in local optima is increased.

One way to mitigate the problem of local optima is the use of cultural algorithms [15]. There, knowledge about the search is stored in a belief space that is influenced by the most successful solutions and which in turn influences all children. However, the implementation of cultural algorithms in a parallel, multi-population setting is not trivial. Either there is a global belief space or separate belief spaces for each population. In the former case, execution runtime will be dictated by the slowest node, as only after all generations have finished their evaluation, the best solution can be identified to update the belief space. In the latter case, the merging of different belief spaces to contain the best solution is rather complex and application specific.

This paper deals with suggesting an entire new class of algorithms that can be used to solve dynamic problems in a parallel fashion, using multiple populations and belief spaces, without the need to merge these belief spaces. The proposed revolutionary algorithms extend the concept of cultural algorithms by evolving culture into politics. In the real world, belief systems are relatively constant over time. However, when a belief system fails to inspire its followers to achieve greater goals, when it fails to make good for what it promises, a revolution might eradicate the belief system itself and replace it with new content.

Revolutionary algorithms seek to emulate this human behavior in their goals to optimize a function. While being comparable to cultural algorithms at first glance, there are important differences. Revolutionary algorithms allow for some solutions to exist and evolve beyond the influence of the belief system. This subculture either prospers or withers and dies. If the alternative path of the subculture becomes more successful than the main line inspired by the belief system, the subculture takes over the belief system and installs its own values. Thereby the former

subculture establishes hegemonic reign over all other solutions, spreading its influence. Until a new subculture evolves and eventually takes over.

This approach promises to successfully evade local minima by always also trying out unorthodox solutions and to rapidly adapt to dynamic problems. Revolutionary algorithms thereby provide a class of algorithms that can be applied to solve problems previously intractable.

2. Overview of Cultural Algorithms

Research in genetic algorithms has produced a vast number of subtypes [8]. In the following, the focus will be put on cultural algorithms. First, the basic concepts of cultural algorithms will be presented. Then, more sophisticated extensions into the realms of multi-population approaches and dynamic optimization are surveyed.

A cultural algorithm as initially envisioned by [15] extends the biologist model of a genetic algorithm with a linked concept of culture. Here, a so called belief system stores important information regarding the search. What precisely entails important information is problem dependent.

For cultural algorithms to function, a communication protocol between the population component and the belief system is required. For once, this protocol must control write access to the belief system and also manages how the belief system influences the individuals. These functions are called update and influence functions, respectively. To extend the evolutionary notion of survival of the fittest, only the best individuals of any generation are permitted to update the belief system.

Herein lies also a potential pitfall: Since the best individuals are exerting a large influence over the entire population via the belief system, local optima can become deadly traps, especially in dynamic scenarios. In a problem where the data in the search space is updated frequently, the belief system must fear to stay behind and actually prevent its population from exploring more promising areas of the search space.

Cultural algorithms are a vital field of research. For instance, they have been daisy chained so that a second cultural algorithm optimizes the belief system obtained from an earlier run. This process, akin to two-pass encoding of media data was demonstrated by [12]. For an overview of genetic algorithms and their suitability for cultural algorithms, see [1].

The extension of cultural algorithms to multi-population approaches is still in its early phase. For a general overview on multi-population evolutionary computing, see [2, 8, 10]. As a quasi-cultural algorithm

with a rudimentary belief system, [14] consider the employment of the mechanics of societal leadership for the purposes of a search. There, solutions cluster together and vote the best of their respective ranks to be a leader. These leaders further influence their constituents. The leaders themselves, after becoming leader are only influenced by other leaders. This approach is marked with a notion of cooperation between competing societies, where information is exchanged for a greater good.

Ref. [5] employ a number of parallel populations, each with a separate belief system. The best solutions of each belief system are exchanged. Ref. [4] extend that approach with having multiple belief systems control the mutation rates of their attached populations. In both of these solutions, there is some form of migration of individuals between belief systems, but the information exchange between belief systems themselves is restricted. Ref. [7] give an excellent account on their approach that only allows for belief system communication; migration of individuals is prohibited. Noteworthy here is, that they envision a complex protocol of merging the contents of different belief systems to form an optimal belief system content. This merged, or enriched content is then copied over to other belief systems.

3. Evolving Evolution: Politics

Culture is a basic treat of humanity. Among other things, culture includes norms of human behavior, limits of human behavior. By this, culture describes behavior that is acceptable and sets forth ways of punishing unacceptable behavior. The processes of negotiating acceptable behavior is also know as politics. If we can settle for politics to equate with that process, then the institutions where this process takes place should be named polity. And finally, the contents themselves are named policies in political science. With these terms, we can start conceptualizing culture as something extremely volatile, by elevating it to the levels of ideology. An ideology promises a greater good to its followers, as long as they adhere to the prescribed norms. By this definition, an ideology will have many followers as long as the encoded norms and values will provide the followers with an advantage over individuals subscribed to another ideology.

Obviously, humankind has produced a great many number of cultures over the course of history. At a naive glance, all of them are equal. Yet, at times, some cultures are more persuasive than others. Let this dominant culture, or ideology be known as hegemon. A hegemonic culture seeks to eradicate opposition and instill its values, its content into every individual as universal. The methods of this instilling vary from one

culture to the next, but the pattern appears to be constant. However, this only works as long as an ideology's promises are being kept.

When any ideology amasses enough power, i.e. enough followers to achieve the status of hegemon, that ideology provides usable solutions to real problems experienced by its followers by definition. Subjectively, the individuals judge the cost of subscribing to another ideology as being higher than to remain with the hegemonic ideology. Unfortunately, ideologies are exceptionally resistant to change. So since the world keeps on turning, and the answers an ideology provides by means of its encoded norms and values fail to provide its subscribers with a definite advantage, this ideology is waning. Its followers will start to look for alternatives to subscribe to, that provide better solutions for their needs.

These dissidents form subcultures of their own, shielded from the hegemon by the use of code and clandestinity. There they develop alternative norms and values (through the means of politics). Eventually, once a dissident group becomes strong enough, they will abandon their hiding and challenge the hegemon. In the real world, this means revolution. Not always these revolutions are bound to be violent, and not always they are predetermined to be successful. But the message is clear: the hegemon has lost support in the population and its belief system is bound to be replaced by something new.

For example, consider the revolutions reaping throughout Europe in and around the key year of 1848. The old ideology of absolute monarchical power had started to fail its followers. In clandestine circles at first, individuals were searching for alternative solutions that were more apt to the problems they faced. Eventually, the masses were taking to the streets, and at great loss of life, challenged the hegemon. What happened next, depended on the country. In some locations, the effective powers of the monarch were limited, in others the hegemon survived the altercation more or less intact.

There have always been multiple, concurrent cultures. These cultures can be conceived as ideologies, and at times, some of them may claim hegemonic and dominant status over others. Whenever a dominant ideology, a hegemon fails to deliver its followers from evil, so to speak, these followers will start to reorient themselves and follow alternative, dissident subcultures. Until, eventually, popular subcultures take over, and it all starts over again. Much of human development is found in this evolution of ideas. From the god-emperors of Rome, to absolute monarchs in medieval Europe to democracies and her challengers, ideas and revolutionary changes to them marked the path. In the next section, this human behavior is mapped onto an algorithm, ripe for applications in the optimization of dynamic problems.

4. Revolutionary Algorithms

The proposed revolutionary algorithms seek to emulate the struggle of ideas for hegemonic position. A revolutionary algorithm is an extension to cultural algorithms with multiple sub-populations, each equipped with their own belief space. The novelty in this approach lies within the communication protocols between populations and belief spaces. To the best of our knowledge, we are the first to suggest this.

At the foundation of a revolutionary algorithm lies a genetic algorithm that steers the evolution of individuals. There is a large number of individuals that concurrently seek out the best solution. The genetics of their reproduction is guided by a belief system, to which a group of individuals adhere. Initially, the link between belief system and individual is random.

During the course of the optimization, individuals are more likely to subscribe to the belief system that is most successful. Once a belief system has lost all of its followers, it will be deleted. Since it is the belief system that an individual subscribes to has definite effect on the genetics of that individual's children, the effect of the belief system is directly visible in the population after a single generation.

So far, this approach is similar to a cultural algorithm with multiple sub-populations and individual migration. The distinct feature lies in the definition of success. In terms of genetic optimization, success is the distance between an individual's fitness and the true minimum. Since the true minimum is not known, a different metric is needed. We suggest to use the average rate of improvement over a certain time frame. So a population/belief system combination that achieves lower values at a greater rate than another one in a given time, the former will be considered more successful. Since the success of a belief system directly translates into the number of followers is has, the notion of choosing a suitable ideology is aptly covered.

A key notion of genetic algorithms is that in the beginning of the optimization process, improvements to the initially random solutions are provided rapidly. At later stages, improvements are only produced over the course of many generations. So in the proposed revolutionary algorithm, initially a hegemon – dominant belief space – will become evident. As the generations progress, the rate of improvement per generation of that belief system will wane, and individuals of the population will start to spawn alternative belief systems and start subscribing to them.

The proposed algorithm starts out with multiple population and belief system combinations. These subcultures all compete for finding the best solution. Eventually, one culture will improve faster and produce the

globally best individual and thus become the global hegemon. However, as this hegemonic subculture comes closer to the perfect solution, its rate of improvement will decrease. As the population being influenced by the hegemon's belief system, fails to witness further improvement, the probability of them being influenced by other subcultural belief systems increases.

The longer the hegemon fails to produce significant improvement, the more subcultures will spawn, most of them with a higher rate of improvement (and thus attractiveness) than the hegemon. These dissident subcultures will lure population away from the hegemon until one of them takes over the role of hegemon.

Since revolutionary algorithms are primarily intended to solve dynamic optimization problems, the number of dissident subcultures spawning is not only dependent on the lack of hegemonial improvement, but also from the detection of a change in data. The more data points change from one time to another, the more subcultures are spawned.

To operationalize human behavior as identified above, some form of communication protocol needs to be established. A way of determining the hegemon needs to be found, as well as a means of clarifying the allegiance of individuals to a belief system. The hegemon shall be the belief system/population combination that is largest, i.e. that has the most followers or whose population component is largest. The allegiance function is more complicated. It is a stochastic function that depends on the number of followers of any belief system, the rate of improvement of that belief system and gives the probability of an individual j appertaining (A) to a belief system i:

$$P(A_{ij}) = \frac{n_i r_i}{N r_j}, \tag{1}$$

with n_i being the proportion of followers of that belief system out of all individuals, r being the success of that belief system or that of the jth individual's belief system and N the number of all belief systems in existence. The outcome of this function is the probability of an individual j subscribing to a belief space i. This allegiance function is at the core of the proposed revolutionary algorithms. It contributes directly to the vitality of a belief system and the chance of revolution against a hegemon.

As the hegemon fails to produce a sufficient rate of improvement, the probability of occurrence of dissidents increases. These dissidents are elements of the hegemon's belief system's population. However, they forsake the hegemons influence and start to form their own, independent belief systems. The spawning of dissidents is a side product of evaluating

the allegiance function described above: The smaller the attractiveness of the hegemon, the larger the probability of dissidents spawning.

$$P(S) = dp \times div \tag{2}$$

This formula gives the probability of dissident sub-populations spawning. It is based on some measure dp quantifying the severeness of a data change in the search space, for instance the proportion of data points that have changed during a certain time frame. The factor div is some measure of diversity: The more diverse the different sub-populations are, the more active they are in different regions of the search space, the less probable is the spawning of new dissidents. A possible way of arriving at this quantity is

$$div = \frac{\max(d_{mn})}{0.5 \sum_{m \neq n} d_{mn}},$$

with d_{mn} being the entries of a Euclidean distance matrix between any two solutions m and n.

Algorithm 1 Pseudocode of a revolutionary algorithm

```
Initialize() r random populations and these individuals' allegiences
Estimate fitness of i individuals in r populations
Update b belief systems according to Update() and Allegience() functions
while not stopCriterion do
    Apply Influence() function with respect to belief system and individual
    Procreate offspring
    Evaluate fitness of offspring
    Update belief systems according to Update() and Allegience() functions
    Spawn new dissidents
    Update individuals allegience
end while
Report best individual
```

The proposed algorithm's pseudocode is presented in Fig. 1. In words, as a first step the r different populations and their – initially random – allegiances are initialized. At the beginning, the belief spaces are still empty. As a next step, every individual i's fitness is being determined and their respective belief spaces are being updated with that knowledge. After the initialization, the optimization starts, until some stop criterion

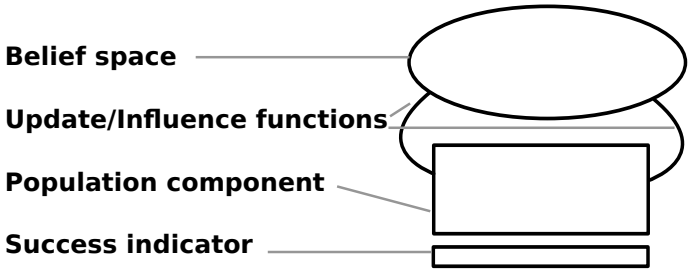


Figure 1. Building block of revolutionary algorithms: the components of a population/belief system combination.

(in dynamic optimization problems, usually the end of data input), is met.

In the optimization phase, first, the cultural algorithm's influence function is applied to directly alter the individuals' behavior with respect to their belief space. Then offspring is procreate as informed by the belief space and this offspring's fitness evaluated. The best individuals are of any generation are then able to update the belief system according to their respective allegiance functions and the global update function. After the belief system has been updated, the allegiance system is re-evaluated if there are any shifts to be recorded. According to the hegemon's performance the availability of new data points, dissidents are spawned. Finally, an individual's allegiance to a given belief system is being updated. This optimization process is continued until some stop criterion is met. Typically, in a dynamic optimization problem, this criterion would be the end of data input.

Figure 2 depicts the process of how a revolutionary algorithm works. The six panels depict six typical stages during an optimization run. The pictogram used is explained in Fig. 1. At first different sub-populations are initialized together with their belief systems. Here, all the population/belief system combinations have the same size. After a few generations, the forming of a hegemon begins: a single combination of population and belief system will outclass the others, as depicted with the success indicator. By being more successful than the competing subcultures, population is drawn to this hegemon. In panel 3, all population has subscribed to the most successful belief system. Now the algorithm either stops, as some stopping criterion is met, or the optimization continues, potentially with new data being introduced. As the hegemon will fail to produce sufficient increase in fitness to keep its followers at bay, dissident sub-cultures will form. Eventually, one of this sub-cultures will

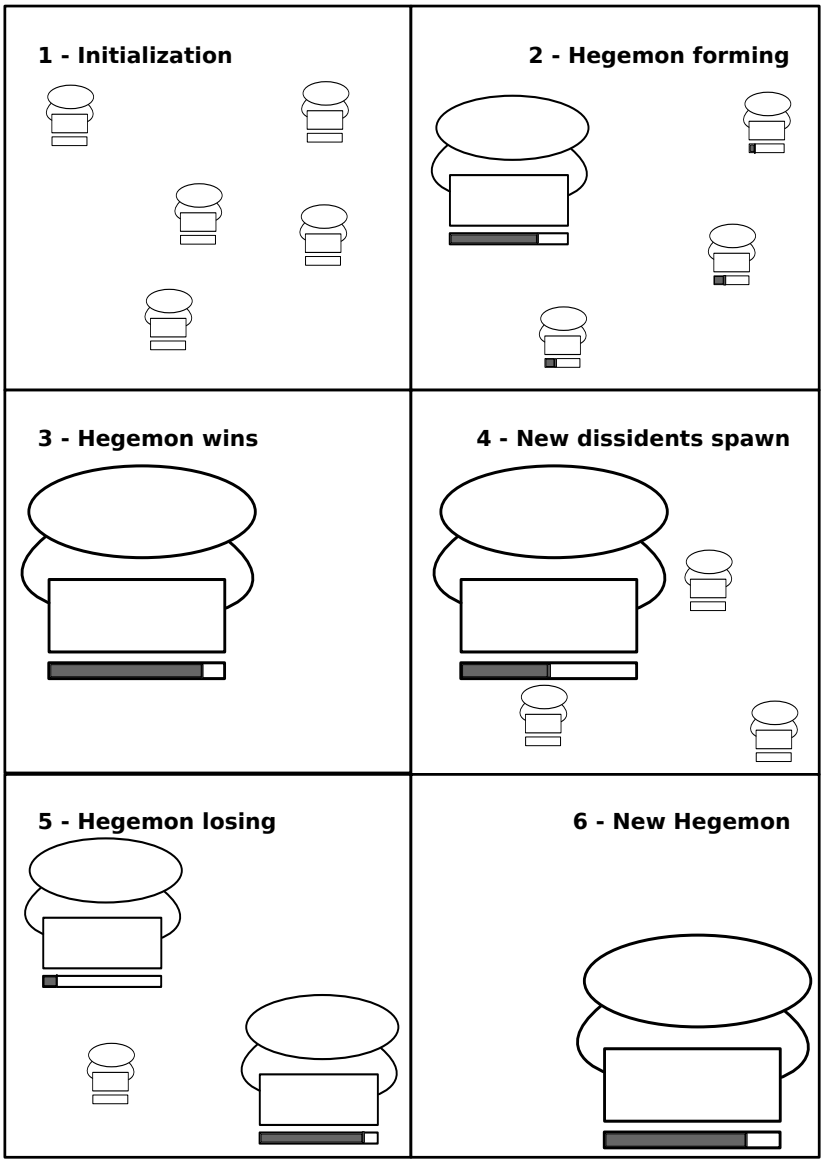


Figure 2. Six typical stages during an optimization run using a revolutionary algorithm.

become popular among the population and draw individuals away from the (former) hegemon, and become a hegemon itself (panels 5 and 6).

5. Conclusion

The proposed evolutionary algorithm, Revolutionary algorithm, seeks to emulate human political behavior. Conceptionally, it is thus an extension of cultural algorithms mapping the competition of different human cultures onto the heuristical determination of an optimum. By doing so, the threat of any cooperation – to peacefully reside at a place, just because a better one is not dared to be imagined, is avoided. Thus, the search can quickly and confidently adopt to updated search spaces and is less prone to get trapped in local optima. As opposed to a simple restart of the search, a revolutionary algorithm keeps track of the previous search history. As suggestions for future work we would like to point to researching runtime behavior of the algorithm in different applications and on combining the idea of multiple, competing belief systems with different genetic algorithms for the population space.

References

[1] R. L. Becerra and C. A. Coello Coello. A cultural algorithm with differential evolution to solve constrained optimization problems. *Lect. Notes Comput. Sc.*, 3315:881–890, 2004.

[2] E. Cantú-Paz. A survey of parallel genetic algorithms. *Calculateurs parallèles, réseaux et systèms répartis*, 10(2):141–171, 1998.

[3] E. Cantú-Paz. *Efficient and accurate parallel genetic algorithms*, volume 1 of *Genetic Algorithms and Evolutionary Computation*. Springer, 2000.

[4] L. S. Coelho, R. C. T. Souza, and V. C. Mariani. Improved differential evolution approach based on cultural algorithm and diversity measure applied to solve economic load dispatch problems. *Math. Comput. Simulat.*, 79(10):3136–3147, 2009.

[5] J. G. Digalakis and K. G. Margaritis. A multipopulation cultural algorithm for the electrical generator scheduling problem. *Math. Comput. Simulat.*, 60(3-5):293–301, 2002.

[6] M. Dorigo and G. Di Caro. Ant colony optimization: a new meta-heuristic. In *Proc. IEEE Congress on Evolutionary Computation*, 1999.

[7] Y. Guo, J. Cheng, Y. Cao, and Y. Lin. A novel multi-population cultural algorithm adopting knowledge migration. *Soft Comput.*, 15(5):897–905, 2011.

[8] R. L. Haupt, S. E. Haupt, and S. E. Haupt. *Practical genetic algorithms*. Wiley-Interscience, 1998.

[9] R. Hochreiter and C. Waldhauser. Evolved election forecasts: using genetic algorithms in improving election forecast results. In *Proc. 13th Annual Conference Companion on Genetic and Evolutionary Computation*, pages 229–230, 2011.

[10] H. Homayounfar, S. Areibi, and F. Wang. An advanced island based GA for optimization problems. In *Proc. International DCDIS Conference on Engineering Applications and Computations*, pages 46–51, 2003.

[11] A. B. Kurzhanski and P. Varaiya. Dynamic optimization for reachability problems. *J. Optimiz. Theory App.*, 108(2):227–251, 2001.

[12] D. A. Ostrowski, T. Tassier, M. Everson, and R. G. Reynolds. Using cultural algorithms to evolve strategies in agent-based models. In *Proc. IEEE Congress on Evolutionary Computation*, pages 741–746, 2002.

[13] S. R. Pliska. A stochastic calculus model of continuous trading: optimal portfolios. *Math. Oper. Res.*, 11(2):371–382, 1986.

[14] T. Ray and K. M. Liew. Society and civilization: An optimization algorithm based on the simulation of social behavior. *IEEE T. Evolut. Comput.*, 7(4):386–396, 2003.

[15] R. G. Reynolds. An introduction to cultural algorithms. In *Proc. 3rd Annual Conference on Evolutionary Programming*, pages 131–139, 1994.

TOWARDS SOCIAL NETWORKS MODEL

Vida Vukašinović, Jurij Šilc
Computer Systems Department, Jožef Stefan Institute, Ljubljana, Slovenia
{vida.vukasinovic; jurij.silc}@ijs.si

Riste Škrekovski
Faculty of Mathematics and Physics, University of Ljubljana, Slovenia
riste.skrekovski@fmf.uni-lj.si

Abstract If we want to imitate a complex human social behavior, we first need to understand the communication patterns inside human societies. We present an interaction-based network model, where each interaction between two individuals has an influence on the future interactions in a similar way as the pheromone trails influence the ant future decision of a path. In the paper, we analyze how this model obeys the features of social networks.

Keywords: Balance theory, Pheromone infrastructures, Small world, Social network, Weighted network.

1. Introduction

When we want to understand the world around us we can use different techniques to describe the observed principles of the nature. One way of exploring associations between objects is to use graphs. One of the most interesting real-world large graph is the graph where we are the vertices and the social ties between us are the edges.

Many large graphs are natural artifacts that have grown through some evolutionary process. They are neither entirely random nor entirely regular. In order to understand them we study the balance of order and chaos inside them. Understanding them bring us new knowledge about the world. Understanding the world bring us a good opportunity to make a progress in technology and science which could make our lives more comfortable.

There are many tries of imitating the nature laws in order to find good optimization methods. Some of them have been more successful

than others, but all of them base on the knowledge extracted from the nature. There is high tendency to use the knowledge of communication abilities of individuals inside social structures found in nature to solve complex optimization problems. Some of socially motivated algorithms are the Particle Swarm Optimization (PSO) [7], the Ant Colony Optimization (ACO) [3], and the Cultural Algorithm (CA) [15]. The PSO is inspired by social behavior of birds or bees inside the swarm, while ACO algorithm bases on the observation of ant colonies. The ants are capable to find the shortest path from a food source to the nest. The ants put chemical substance called pheromone on trail while walking from the food source to the nest. During the food search each ant probabilistically prefers to follow a direction rich in pheromone. In CA Reynolds built a computational model derived from observing the cultural evolution process. Social interactions between people are more complex than social interactions between various animal species and therefore the human population have better chance to find better solutions for their problems. CA uses five basic knowledge types that human social systems support and each of them is supported by various animal species [18].

Nevertheless, we still do not understand human social systems well enough and in this paper we try to come one step closer. In the last decades there grows a tendency to consider human social systems as social networks and then to study the properties of the networks in order to understand them. One way to approach them is to build a model and to observe how well this model fits real-world graphs. This paper represents an interaction-based model, where each interaction between two individuals has an influence on the future interactions in a similar way as pheromone trails influence the ant future decision of a path.

2. Social Network Models

Some ideas of the social network theory could be found very far in the history. However, the major developments in the field have been done since the beginning of twenty century when the graph theory became more statistical and algorithmic. It turned out that the large real-world graphs do not have only large size in common but they also tend to be sparse, clustered and they tend to have a small diameter. That kind of graphs are termed small world graphs. The diameter of a graph is the longest shortest path between any two graph vertices. The clustering coefficient of a vertex measures how close the vertex neighbors are to being a complete graph. The clustering coefficient of a graph is the average clustering coefficient of vertices. It is calculated as

$c = n^{-1} \sum_{i=1}^{n} \binom{|N(i)|}{2}^{-1} |E(N(i))|$, where n is the graph size, $N(i)$ is the neighborhood of the vertex i, and $|E(N(i))|$ is the number of edges in the graph induced by $N(i)$.

Small worlds have $O(n \log n)$ edges, the diameter is $O(\log n)$ and the clustering coefficient is a small constant independent from the network size. Watts and Strogatz [17] found a natural way to build small world graphs by interpolating between lattice and random graph. They start with a regular lattice and then rewire every edge with a given probability p.

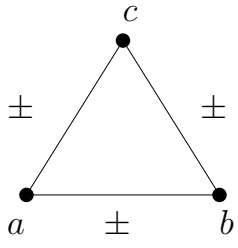
Some of the first studies on distances in social networks were done by Rapoport [14] and later by de Sola Pool and Kochen [16], while at about the same time Milgram [12] in 1960's made the first empirical work. Milgram in his experiment prepared letters addressed to a person in Massachusetts and asked some persons in Nebraska to send a letter to someone in his acquaintances who might be closer to the target person. The average number of intermediate steps of the trials that successfully reached the target person was found to lie between five and six. From this result comes the well known principle called "six degree of separation". Kleinberg [8] criticizes the Watts and Strogatz model [17] since it does not consider Milgrams findings [12]: individuals using local information are collectively very effective at actually constructing short paths between two vertices in a social network. He built a model for which decentralized algorithms are effective [8].

Sparseness, clustering and small diameter are not the only properties of large real-world graphs. Barabási and Réka [1] found out that in many large networks the vertex degree follow a scale-free power-law distribution. However, it has been shown that some real networks do not obey the power-law distribution [13], despite that the degree distributions are skewed.

3. Locality Principle, Balance Theory and Weighted Ties

We believe that most of noticeable interactions that happen to an average person during a day are the interactions between the person and his acquaintances or the acquaintances of his acquaintance. We can more generally say that in the social networks majority of the meaningful interactions among individuals are based on the local information of the network. Based on this principle Davidsen et al. [2] simulate the evolution of social networks. The dynamics of their model is defined by randomly linking up neighboring vertices: in case the vertex does not have enough neighbors, one other randomly chosen vertex is linked.

Interactions between people could cause positive or negative sentiments. These sentiments define how people perceive their relations with other people. Heider's balance theory [6] says that two persons equivalently perceive their relations as positive or negative. Three persons perceive their triadic relations as balanced or imbalanced, depending on the number of positive and negative relations between them. If there are three positive or one positive and two negative relations, they perceive it as balanced. Otherwise, if there are one or three negative relations, they perceive it as imbalanced, as presented in Fig. 1. From the perspective of a person the balanced triadic relations can be understood with one of the following statements: "friend of my friend is my friend", "friend of my enemy is my enemy", "enemy of my friend is my enemy", and "enemy of my enemy is my friend". According to social balance theory imbalanced triads are uncomfortable for the persons involved. Moreover, they could cause reorganization of the entire network. Persons in the imbalanced triads tend to make them balanced or some connections may broke.



Triadic relation	*ab*	*ac*	*bc*
Balanced	+	+	+
	−	+	−
	+	−	−
	−	−	+
Imbalanced	+	−	+
	−	+	+
	+	+	−
	−	−	−

Figure 1. Balanced and imbalanced triadic relations where every two individuals perceive their relation as being positive or negative.

Based on these insights Ludwig and Abell [11] built an evolutionary social network model. They build a model by sequentially attaching randomly selected positive and negative edges to a given set of vertices. At each time step they take care that each vertex keeps low number of imbalanced triads.

Differences in relations are not just in positive and negative perceptions but also in the way how close (i.e. strong) are the relations between different persons. Therefore, weighted networks have been introduced: to the edges of these networks are assigned appropriate weights, where the weight quantifies the strength of the given relation. It is natural to expect that weights have an influence on the formation of the network. In the weak link hypothesis Granovetter [5] says that weak ties between

people connect two or more communities keep the network connected whereas strong ties are mostly inside communities.

Kumpula et al. [10] present a model where the weights are generated dynamically. They distinguish two mechanisms of tie formation. The first one refers to forming ties with one's network neighbor, while other refers to forming ties between people who share the same activities independently of the distance inside the network. In their model these two mechanisms are local attachment and global attachment. Local attachment process is local two-step weighted self avoiding random walk. In local attachment the visited edges and the edge between the first and last vertex in weighted random walk are reinforced by a small amount. In global attachment a random edge is established with some probability p. Such model proves weak link hypothesis.

4. Interaction-Based Model

What keeps ties between us are everyday interactions. More interactions happen between two persons, more is this relation important for them on daily basis and it has higher weight. The interactions could be positive or negative and they appear as described in previous section. These interactions have a strong influence on how individuals will perceive their relations. More often the interaction between two persons is positive, more friendly their relation is, and therefore their tie is stronger. Negative interactions between friends make their friendship weaker until the connection breaks or becomes negative.

According to balance theory [6], interactions between individuals in triads tend to be balanced and the relations as they are, are the results of balanced interactions between them. It seems that the interactions between people who are connected with stronger ties are more often, being them positive or negative. There are some connections between persons, which are not strong, but does not seem to be affected by anything. These are usually weak ties between persons from different communities [5]. We model the network growth by simulating interactions between individuals in triads, where each interaction has an influence on the quality of the relation.

We consider a fixed size network of N vertices. At each time step, each vertex having at least one neighbor starts a weighted local search for new acquaintances, see Fig. 2. The vertex i chooses its neighbor j with probability $|w_{ij}|/s_i$, where w_{ij} is the weight of the edge between vertices i and j, and $s_i = \sum_{j \in N(i)} |w_{ij}|$ is the strength of i. The weight w_{ij} is increased for δ with probability p_+ and it is decreased for δ with probability $1 - p_+$. If the chosen vertex j has other neighbors besides

i, it chooses out of them a vertex k with probability $|w_{jk}|/(s_j - |w_{ij}|)$. Similarly as before, w_{jk} is increased for δ with probability p_+ and it is decreased for δ with probability $1 - p_+$. According to the balance theory the link between i and k is established or reinforced by positive or negative amount of δ. If vertices i or j have no neighbors, they create a connection to a random chosen vertex, where the connection is positive with probability p_+ and negative otherwise.

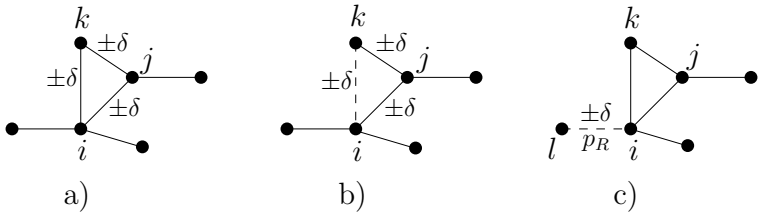


Figure 2. Weighted local search and preferential random attachment rules. A weighted local search starts at vertex i, proceeds to vertex j and then to k. Weights on ij and jk are increased by δ with probability p_+ and otherwise they are decreased by δ. Weight on ik is established or refreshed by $\pm\delta$ considering the balance theory. a) vertices j and k are i's neighbors. b) if vertex k is not i's neighbor, new edge is established. c) with probability p_R an edge incident to a vertex l chosen by the random proportional rule is established or refreshed.

In addition, at every time step each vertex establishes a new connection with probability p_R using the random proportional rule. The probability of choosing a particular vertex increases with the vertex degree value. The weight of connection is increased by δ with probability p_+ and decreased otherwise.

After each time step the absolute value of the weights on edges are decreased by some amount ε. Thus, if the weight is positive, then it is reduced by ε, and if the weight is negative, then the weight is increased by ε. The weights of edges loose their strength in a similar way as the pheromone amount on ant's trails evaporate.

5. Results

The proposed model was studied by simulations that start from a graph with N vertices and no edges. We tested models where at each time step weighted local search and random attachment procedures are executed on every vertex of the network ($p_R = 1$). The value of δ was set to 1.0, while the probability that positive interaction happens p_+ was set to 0.6 in weighted local search and to 1.0 in random attachment procedure. After each time step weights on edges evaporate by $\varepsilon =$

$\frac{\delta}{2}$. We tested models on $N \in \{10, 100, 1000, 5000, 10000\}$ vertices and evaluated them at five different time steps.

In Fig. 3 are presented degree distributions for the models of different size and time steps. For each model we did 15 runs and calculated the average of their properties. For each number of vertices the models were evaluated at different time steps as larger networks need more time to come into the mature phase. This phase is the stable state where the number of edges does not change significantly.

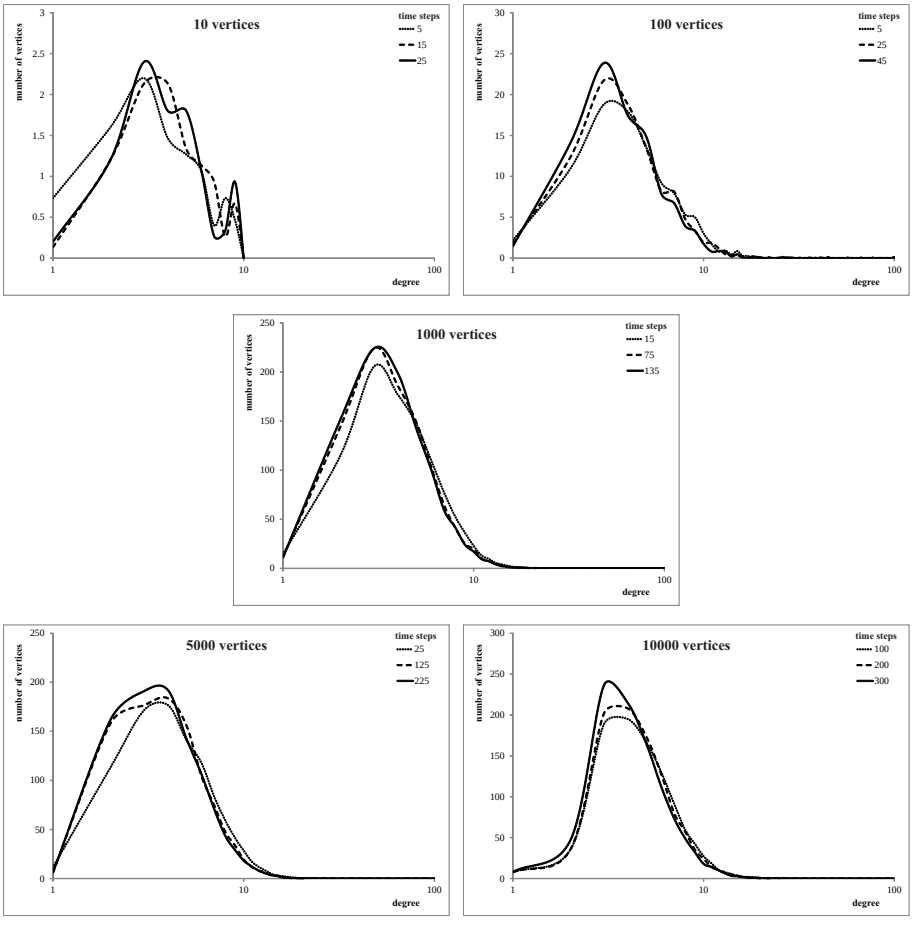


Figure 3. Degree distributions for different network size at different time steps in already stable state of the network. In order to capture the differences between the simulated models corresponding to different time steps, the degree values are plotted on logarithmic scale.

Table 1. Main properties of the interaction-based network model at different network size.

# vertices	# edges	diameter	clustering coefficient	# time steps
10	22.47	2.4	0.646	30
100	254.80	5.0	0.350	45
1000	2449.93	7.4	0.296	135
5000	12101.07	8.6	0.312	225
10000	24884.45	9.1	0.319	300

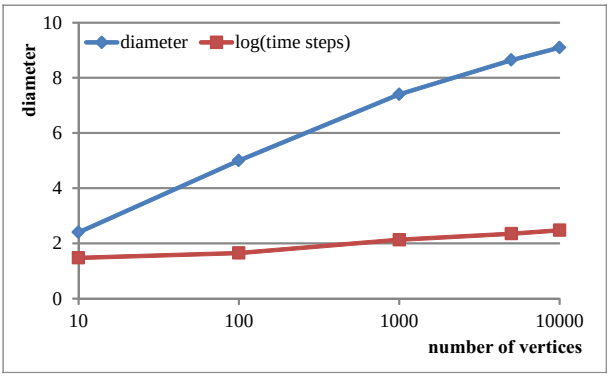


Figure 4. Diameter of a network in a mature phase changes as $\log N$.

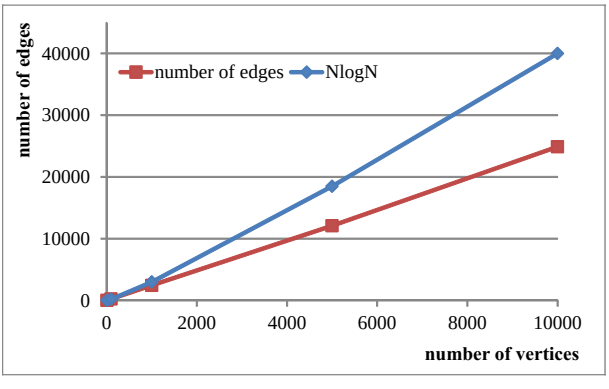


Figure 5. Number of edges grows linear with the network size.

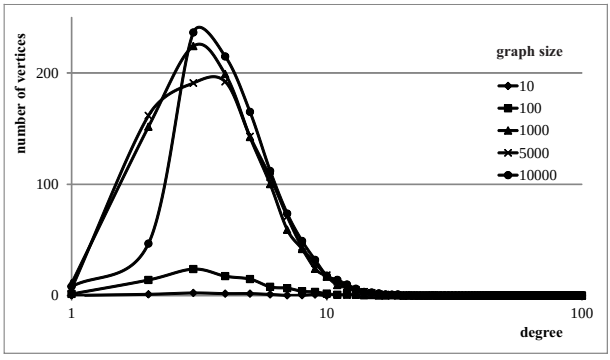


Figure 6. Degree distributions are skewed and the position of the peak of the distribution does not change as the number of vertices grows. In order to capture the differences between the simulated models corresponding to different graph sizes, the degree values are plotted on logarithmic scale.

Table 1 shows how the diameter and the number of edges are changing as the number of vertices grows. From Table 1 it is evident that the clustering coefficient stays independent from the number of vertices, which is also the small world property. Figure 4 shows that the diameter grows as $\log N$. The number of edges grows linear with the number of vertices N (see Fig. 5), while the number of edges in small worlds usually grows as $N \log N$. These results (Table 1 and Fig. 4) suit the small world properties, except in the case of two sparse networks (Fig. 5), where the number of edges changes linearly with the number of vertices.

The degree distributions (Fig. 6) are skewed and the position of the peak of the degree distribution does not change as number of vertices grows.

In addition, we studied how models change while the probability that positive interacted happens p_+ changes. Table 2 shows how the number of positive edges and the clustering coefficient are increasing and the diameter is decreasing, while the probability p_+ in weighted local search is increasing. Figure 7 shows an example network with 1000 vertices at 135 time steps and $p_+ = 0.6$.

6. Conclusion

We have introduced an interaction-based model based on the balance theory and sociological locality principle, where every interaction between individuals has an influence on the connection between them; being positive or negative. Such models turned out to be sparse, they

Table 2. Main properties of Interaction-based network model at size 1000 and different p_+ parameter in weighted local search procedure.

p_+	# edges	# pos. edges	# neg. edges	diameter	clustering coefficient
0.4	2491.33	1563.00	928.33	7.7	0.248
0.6	2449.93	1773.20	676.73	7.4	0.296
0.8	2400.60	2114.53	286.07	6.3	0.438
1	2397.93	2397.93	0	5.3	0.568

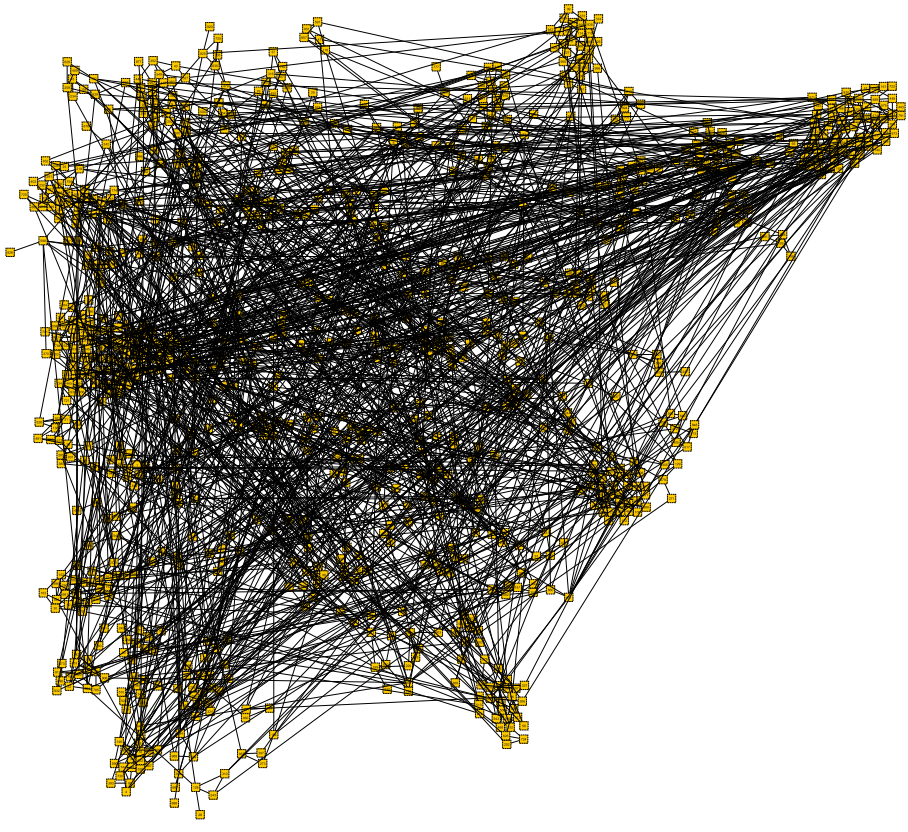


Figure 7. An example network with 1000 vertices after 135 time steps and $p_+ = 0.6$.

have a small diameter and the clustering coefficient is independent from the number of vertices and rather high.

In the future we plan to analyze some other properties of these models, such as number of balanced and imbalanced triangles in the network. Moreover, we are interested how the structure of the networks with large number of negative weights compares with networks that have mostly positive weights.

At first glance, in our models, most of the weak ties break and the new weak ties are established in time. Therefore, we also plan to study what are untouchable weak ties according to Granovetter [5] that connect different communities.

References

[1] A.-L. Barabási and A. Réka. Emergence of scaling in random networks. *Science*, 286(5439):509–512, 1999.

[2] J. Davidsen, H. Ebel, and S. Bornholdt. Emergence of a small world from local interactions: Modeling acquaintance networks. *Phys. Rev. Lett.*, 88(12):128701, 2002.

[3] M. Dorigo, V. Maniezzo, and A. Colorni. Ant system: Optimization by a colony of cooperating agents. *IEEE T. Syst. Man Cy. B*, 26(1):29–41, 1996.

[4] A. Fabrikant, E. Koutsoupias, and C. H. Papadimitriou. Heuristically optimized trade-offs: a new paradigm for power laws in the Internet. In *Proc. 29th International Colloquium on Automata, Languages and Programming*, pages 110–122, 2002.

[5] M. S. Granovetter. The strength of weak ties. *Am. J. Sociol.*, 78(6):1360–1380, 1973.

[6] F. Heider. Attitudes and cognitive organization. *J. Psychol.*, 21:107–112, 1946.

[7] J. Kennedy and R. C. Eberhart. Particle swarm optimization. In *Proc. IEEE International Conference on Neural Networks*, volume 4, pages 1942–1948, 1995.

[8] J. Kleinberg. The small-world phenomenon: an algorithmic perspective. Cornell Computer Science Technical Report 99-1776, 1999.

[9] G. Kossinets and D. J. Watts. Empirical analysis of an evolving social network. *Science*, 311(5757):88–90, 2006.

[10] J. M. Kumpula, J. Onnela, J. Saramäki, K. Kaski, and J. Kertesz. Emergence of communities in weighted networks. *Phys. Rev. Lett.*, 99(22):228701, 2007.

[11] M. Ludwig and P. Abell. An evolutionary model of social networks. *Eur. Phys. J. B*, 58(1):97–105, 2007.

[12] S. Milgram. The small world problem. *Psychol. Today*, 1(1):61–67, 1967.

[13] M. E. J. Newman. The structure of scientific collaboration networks. *P. Natl Acad. Sci. USA*, 98(2):404–409, 2001.

[14] A. Rapoport. Contribution to the theory of random and biased nets. *B. Math. Biol.*, 19(4):257–277, 1957.

[15] G. R. Reynolds. An introduction to cultural algorithms. In *Proc. 3rd Annual Conference on Evolutionary Programming*, pages 131–139, 1994.

[16] I. de Sola Pool and M. Kochen. Contacts and influence. *Soc. Networks*, 1(1):5–58, 1978/79.

[17] D. J. Wattsand and S. H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393(6684):440–442, 1998.

[18] C. D. L. Wynne. *Animal Cognition: The Mental Lives of Animals*. Palgrave Macmillan, 2001.

SHRINKING THREE STAGE OPTIMAL MEMETIC EXPLORATION

Ilpo Poikolainen
Department of Mathematical Information Technology, University of Jyväskylä, Finland
ilpo.poikolainen@jyu.fi

Giovanni Iacca
INCAS^3, Assen, The Netherlands
giovanniiacca@incas3.eu

Ferrante Neri, Ernesto Mininno, Matthieu Weber
Department of Mathematical Information Technology, University of Jyväskylä, Finland
{ferrante.neri; ernesto.mininno; matthieu.weber}@jyu.fi

Abstract The Ockham's razor in Memetic Computing states that optimization algorithms composed of few, simple, and tailored components can be very efficient, if properly designed. If the designer is aware of the role and effect of each algorithmic component an high performance can be easily obtained. Following this principle, this paper proposes a novel algorithm for numerical optimization. The proposed algorithm, namely Shrinking Three Stage Optimal Memetic Exploration (S-3SOME), performs the progressive perturbation of a candidate solution by alternating three search operators, the first is a stochastic global search, the second is random sampling within progressive narrowing hypervolume, the third is a deterministic local search. The proposed S-3SOME is an efficient scheme which outperforms, for the considered problems, a similar scheme proposed in literature and, despite its simplicity, is competitive with complex population-based algorithms which require massive overhead and memory employment.

Keywords: Algorithm for resource-constrained hardware, Computational intelligence optimization, Memetic computing.

1. Introduction

During the latest years, modern research papers in numerical optimization claim to propose high performance general purpose algorithms. For practical reasons, these algorithms are usually tested on standardized sets of artificial test problems proposed in conferences and special issues and are compared with other recent algorithms. Commonly, new algorithms are not really based on novel ideas, on the contrary, they are designed by following one of the design criteria mentioned below.

1) Starting from an existing optimization algorithm, its structure is "perturbed" by slightly modifying the structure and adding on extra components. Obviously, this approach attempts to obtain a certain performance improvement in correspondence to the proposed modifications. Successful examples of this research approaches are given in [2], where a controlled randomization on Differential Evolution (DE) parameters offers a promising alternative to the standard DE framework, and [5], where the variation operator combining the solutions of a population is modified in the context of Particle Swarm Optimization (PSO).

2) Starting from a set of algorithms, they are combined in a hybrid fashion with the trust that their combination and coordination leads to a flexible structure displaying a better performance than the various algorithms considered separately. Two examples of recently proposed algorithms which are basically the combination, by means of activation probabilities, of various meta-heuristics are given in [9] and [13].

Recently, algorithmic design has been tackled by means of the so-called Ockham Razor's principle in Memetic Computing (MC) or more generally in Computational Intelligence Optimization, see [4]. According to the Ockham's Razor, the simplest explanation of natural phenomena is likely to be the closest to the truth. In an analogous way, an optimization problem can be seen as a natural phenomenon and the optimization algorithm should be based on its understanding. This means that the optimization algorithm contains the countermeasures to handle the features of the fitness landscape. In order, on one hand, to avoid a waste of computational resources, and, on the other hand, to allow a proper algorithmic development when the fitness landscape changes (adding and removing memes/components/modules) only the strictly necessary components must be employed.

It must be remarked that we are referring to the concept of MC as reported in [7], MC is intended as "a broad subject which studies complex and dynamic computing structures composed of interacting modules (memes) whose evolution dynamics is inspired by the diffusion of ideas. Memes are simple strategies whose harmonic coordination allows the so-

lution of various problems". In a metaphorical way, the proposed view on the subject considers the various operators as cooking ingredients. The ingredients require then to be properly selected and combined in order to address the specific taste and requirements of the guests. Thus, the algorithmic design should be performed by implementing a bottom-up procedure where the role and function of each operator should be clear. On the basis of this idea, in [4], it is shown how a very simple algorithm, namely Three Stage Optimal Memetic Exploration (3SOME), which combines three perturbation mechanisms over a single solution, is competitive with complex algorithms representing the-state-of-the-art in Computational Intelligence Optimization, while requiring only little memory and computational resources (see [4] for a detailed analysis of the computational overhead and memory usage of 3SOME).

By following the *lex parsimoniae* principle of Ockham's Razor and the example of 3SOME algorithm, this paper proposes a simple algorithmic solution obtained by the sequential application of three perturbation procedures, one is a random global search which attempts to detect promising search directions, one is a random local search and the last is a deterministic local search. Two perturbation mechanisms are taken from the 3SOME algorithm while one of them has been replaced with a novel one. More specifically, while 3SOME is built-up on the idea that the search of the optimum should be reached by alternating the radius of the search and thus using long, middle, and short distance perturbation, this paper combines the long distance exploration with two short distance operators, with different roles, the first is stochastic and progressively focuses the search towards the most interesting areas of the decision space, the second is deterministic and perturbs each variable separately. The combination of these two different operators has been done on purpose to flexibly handle landscapes with diverse features. Since the stochastic operator performs the search by means of a progressive shrinking of the hypervolume, the proposed algorithm is indicated as Shrinking 3SOME (S-3SOME).

The remainder of this paper is organized in the following way. Section 2 gives a detailed description of the three operators and the coordination scheme composing the proposed algorithm. Section 3 shows the performance comparison between the proposed algorithm, 3SOME algorithm, and a set of modern algorithms. Finally, Section 4 gives the conclusive remarks of this study.

2. Shrinking Three Stage Optimal Memetic Exploration

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

In the beginning of the optimization procedure one candidate solution is randomly sampled within the decision space D. In analogy with compact optimization, see [8], we will refer to this candidate solution as elite and indicate it with the symbol x_e. In addition to x_e, the algorithm makes use of another memory slot for attempting to detect other solutions. The latter solution, namely trial, is indicated with x_t. In the following subsections the three exploratory stages and their coordination are described.

2.1 Long Distance Exploration

This exploration move attempts to detect a new promising solution within the entire decision space. While the elite x_e is retained, at first, a trial solution x_t is generated by randomly sampling a new set of n genes. Subsequently, the exponential crossover in the fashion of DE is applied between x_e and x_t, see [10]. More specifically, one gene from x_e is randomly selected. This gene replaces the corresponding gene within the trial solution x_t. Then, a set of random numbers between 0 and 1 are generated. As long as $rand(0,1) \leq Cr$, where the crossover rate Cr is a predetermined parameter, the design variables from the elite x_e are copied into the corresponding positions of the trial solution x_t. The first time that $rand(0,1) > Cr$, the copy process is interrupted. Thus, all the remaining design variables of the offspring are those initially sampled (belonging to the original x_t). This exploration stage performs the global stochastic search and thus attempts to detect unexplored promising basins of attraction. On the other hand, while this search mechanism extensively explores the decision space, it also promotes retention of a small section of the elite within the trial solution. This kind of inheritance of some genes appears to be extremely beneficial in terms of performance with respect to a stochastic blind search (which would generate a completely new solution at each step). If the trial solution outperforms the elite, a replacement occurs. A replacement has been set also if the newly generated solution has the same performance of the elite. This is to prevent the search getting trapped in some plateaus of the decision space (regions of the decision space characterized by a null gradient). The pseudo-code of this component is shown in Algorithm 1.

It can easily be observed that, for a given value of Cr, the meaning of the long distance exploration would change with the dimensionality of the problem. In order to avoid this problem and make the crossover action independent on the dimensionality of the problem, the following quantity, namely inheritance factor, is fixed: $\alpha_e \approx \frac{n_e}{n}$, where n_e is the number of genes we expect to copy from x_e into x_t in addition to the gene deterministically copied. The probability that n_e genes are copied is $Cr^{n_e} = Cr^{n\alpha_e}$. In order to control the approximate amount of copied genes and to achieve that about n_e genes are copied into the offspring we imposed that $Cr^{n\alpha_e} = 0.5$. It can easily be seen that, for a chosen α_e, the crossover rate can be set on the basis of the dimensionality as follows: $Cr = \frac{1}{n\alpha_e\sqrt{2}}$. The long distance exploration is repeated until it does not detect a solution that outperforms the original elite. When a new promising solution is detected, and thus the elite is updated, the stochastic short distance exploration is activated.

Algorithm 1 Long distance exploration

```
generate a random solution x_t within D
generate i = round(n · rand(0,1))
x_t[i] = x_e[i]
while rand(0,1) ≤ Cr do
   x_t[i] = x_e[i]
   i = i + 1
   if i == n then
      i = 1
   end if
end while
if f(x_t) ≤ f(x_e) then
   x_e = x_t
end if
```

2.2 Stochastic Short Distance Exploration

This exploration move attempts to detect promising areas of the decision space by making use of a stochastic logic. In a nutshell, when the long distance exploration detects a new promising solution, the stochastic short distance exploration generates a hypercube centered in the newly detected solution x_e and having a hypervolume which is 20% of the decision space D. This exploration samples a point for n times and simply attempts to outperform solution x_e where n is dimensionality of the problem. In other words, for n times, a trial solution x_t is generated. If $f(x_t) \leq f(x_e)$, the elite solution is updated and the hypervolume is centered around the new elite solution. If, after n comparisons, at least one replacement occurred, n new samples of trial solutions and the re-

spective comparisons are scheduled. On the contrary, if all the n comparisons led to no improvements, the hypervolume is halved and the search is repeated by sampling n solutions around the elite x_e. This shrinking mechanism is repeated until the hypervolume is smaller than 0.0001% of the total hypervolume. The pseudo-code displaying the working principles of the stochastic short distance exploration is given in Algorithm 2. The new elite x_e is then passed to the following operator for further improvements. The size of the initial hypervolume and the number of times the hypercube volume is halved were empirically fixed to achieve good performance over the various problems considered in this paper.

Algorithm 2 Stochastic short distance exploration

```
generate a hypercube around x_e with a hypervolume 20% of that of D;
while the hypervolume is bigger than 0.0001% of D do
  for i = 1 : n do
    generate randomly a trial solution x_t within the hypercube;
    if f(x_t) ≤ f(x_e) then
      x_e = x_t;
      centre the hypercube around x_e;
    end if
  end for
  if no elite update occurred then
    halve the hypervolume;
  end if
end while
```

2.3 Deterministic Short Distance Exploration

This exploration move attempts to fully exploit promising search directions. The meaning of this exploration stage is to perform the descent of promising basins of attraction and possibly finalize the search if the basin of attraction is globally optimal. De facto, the short distance exploration is a simple steepest descent deterministic local search algorithm, with an exploratory move similar to that of Hooke-Jeeves algorithm, see [3], or the first local search algorithm of the multiple trajectory search, see [12]. The short distance exploration stage requires an additional memory slot, which will be referred to as x_s (s stands for short). Starting from the elite x_e, this local search, explores each coordinate i (each gene) and samples $x_s[i] = x_e[i] - \rho$, where ρ is the exploratory radius. Subsequently, if x_s outperforms x_e, the trial solution x_t is updated (it takes the value of x_s), otherwise a half step in the opposite direction $x_s[i] = x_e[i] + \frac{\rho}{2}$ is performed. Again, x_s replaces x_t if it outperforms x_e. If there is no update, i.e. the exploration was unsuccessful, the radius ρ is halved. This exploration is repeated for all the design variables and

Algorithm 3 Deterministic short distance exploration

```
while local budget condition do
  x_t = x_e
  x_s = x_e
  for i = 1 : n do
    x_s[i] = x_e[i] − ρ
    if f(x_s) ≤ f(x_t) then
      x_t = x_s
    else
      x_s[i] = x_e[i] + ρ/2
      if f(x_s) ≤ f(x_t) then
        x_t = x_s
      end if
    end if
  end for
  if f(x_t) ≤ f(x_e) then
    x_e = x_t
  else
    ρ = ρ/2
  end if
end while
```

stopped when a prefixed budget (equal to 150 iterations as suggested in [12]) is exceeded. The pseudo-code displaying the working principles of the deterministic short distance exploration is given in Algorithm 3.

After the application of the deterministic short distance exploration, if there is an improvement in the quality of the solution, the stochastic short distance exploration is repeated subsequently. Otherwise, if no improvement in solution quality is found, the long distance search is activated to attempt to find new basins of attractions.

As a remark, a toroidal management of the bounds has been implemented for the three operators above. This means that if, along the dimension i, the design variable $x[i]$ exceeds the bounds by a value of ζ, it is reinserted from the other end of the interval at a distance of ζ from the edge, i.e. given an interval $[a, b]$, if $x[i] = b + \zeta$ it takes the value of $a + \zeta$.

Finally, if we consider that each operator (meme) processes an elite x_e and returns, as an output, a fitness-wise improved elite solution, the operator can be said to "succeed" if the output is different from the input (and obviously better than it) and can be said to "fail" otherwise. In this light, S-3SOME functioning is represented by the scheme composed of interacting memes reported in Fig. 1. With the words, "Long", "Stochastic short", and "Deterministic short" the three above mentioned operators are represented. The arrows represent the interaction amongst

the memes. The "S" and "F", represent success and failure, respectively, of the meme application.

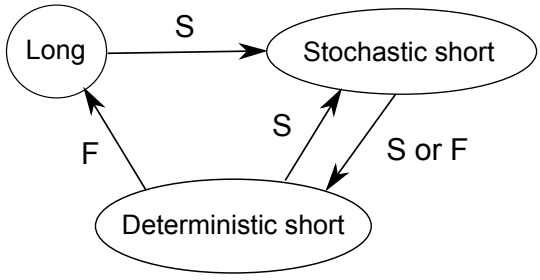


Figure 1. Functioning scheme of S-3SOME.

3. Numerical Results

In order to test the potential of S-3SOME, numerical experiments have been performed on the testbed in [6] (24 problems) in 10, 40, and 100 dimensions. The proposed S-3SOME has been compared with the 3SOME algorithm proposed in [4]. Both 3SOME and S-3SOME have the same parameter values for the two common components, i.e. $\alpha_e = 0.05$ and ρ equal to 40% of the total decision space width. With reference to 3SOME, it can be mentioned that middle distance exploration samples $4n$ points at each activation, as suggested in the original paper. In addition, S-3SOME has been compared with the following modern algorithms. 1) Covariance Matrix Adaptive Evolution Strategy with increasing population size (G-CMA-ES) proposed in [1] with initial population $\lambda_{start} = 10$ and factor for increasing the population size equal to 2. All the other parameters are set to standard values. 2) Self-Adaptive Differential Evolution (SADE) proposed in [11]. SADE has been run with Learning Period $LP = 20$ and population size $N_p = 50$. The other cconstant values are the same reported in the formulas of the original paper. For each algorithm and test problem, 30 runs have been performed. Each run has been continued for $5000n$ fitness evaluations. Numerical results containing average final value and standard deviations are reported in Tables 1, 2, and 3, respectively. The best results are highlighted in bold face. In order to strengthen the statistical significance of the results, the Wilcoxon Rank-Sum test has also been applied according to the description given in [14], where the confidence level has been fixed at 0.95. The null-hypothesis being that the results of both algorithms come from the same statistical distribution, the rejection of the null-hypothesis is indicated with the symbol "+"("-") –

the reference algorithm thus performs better (worse) than the algorithm labeled on the top of the column – while the symbol "=" indicates that the null-hypothesis cannot be rejected and that both algorithms have a statistically equivalent performance.

Numerical results show that the S-3SOME scheme appears to be, on a regular basis, more promising than the 3SOME scheme. According to our interpretation the shrinking mechanism is leading to an efficient algorithmic behavior. More specifically, S-3SOME, in a memetic fashion, contains two local search components that compete and cooperate tackling the problems from different perspectives. While the deterministic search is very efficient for separable problems or for non highly multivariate landscapes, the stochastic search attempts to improve upon the solutions regardless of the separability or multi-modality of problem. The combination of these two simple mechanism with a stochastic global search leads to the generation of a very efficient algorithm capable to compete with modern complex optimization algorithms. For example, it can be observed that S-3SOME tends to have, for all the dimensionality levels under examination, a similar performance with respect to the G-CMA-ES. Regarding the comparison with SADE, an interesting observation can be done: while for low dimensionality levels SADE seems to be extremely competitive, in 100 dimensions its performance tends to deteriorate and is outperformed by S-3SOME. This fact can be explained that while SADE is a complex and efficient structure designed for addressing problems with a low dimensionality (tuned for up to 30 dimensions), the proposed approach lead to simple (which employ limited resources and can be encoded in a few lines) and flexible algorithms having a respectable performance for a various set of optimization problems.

4. Conclusion

This paper proposes a simple scheme for numerical optimization. The design of the algorithm has been performed in a bottom-up fashion by following the *lex parsimoniae*. The proposed algorithm, namely Shrinking Three Stage Optimal Memetic Exploration (S-3SOME) is composed of a random optimizer, a stochastic local search which progressively shrinks the search radius, and a deterministic local search. Numerical results show that this simple scheme is flexible and performs well on a set of diverse problems and for various dimensionality levels. The comparison with the inspiring algorithm, i.e. 3SOME, shows that S-3SOME enhances upon its performance by converging to similar or better solution. The comparison with modern complex algorithms demonstrates

Table 1. Average Fitness ± Standard Deviation and Wilcoxon rank-sum test on the Fitness (reference = S-3SOME) for 10D case.

	S-3SOME	*GCMAES*		SADE		*3SOME*	
f1	$\mathbf{7.95e+01 \pm 0.00e+00}$	$\mathbf{7.95e+01 \pm 0.00e+00}$	=	$\mathbf{7.95e+01 \pm 0.00e+00}$	=	$\mathbf{7.95e+01 \pm 0.00e+00}$	=
f2	$\mathbf{-2.10e+02 \pm 0.00e+00}$	$1.37e+03 \pm 1.16e+03$	+	$\mathbf{-2.10e+02 \pm 0.00e+00}$	=	$\mathbf{-2.10e+02 \pm 0.00e+00}$	=
f3	$-4.60e+02 \pm 1.28e+00$	$-5.32e+01 \pm 8.06e+01$	+	$-4.60e+02 \pm 2.66e+00$	=	$\mathbf{-4.61e+02 \pm 1.05e+00}$	=
f4	$-4.59e+02 \pm 1.70e+00$	$-1.59e+01 \pm 8.88e+01$	+	$-4.59e+02 \pm 2.61e+00$	=	$\mathbf{-4.60e+02 \pm 9.64e-01}$	-
f5	$5.14e+00 \pm 2.87e+01$	$1.11e+01 \pm 8.59e+00$	+	$\mathbf{-9.21e+00 \pm 0.00e+00}$	=	$2.52e+01 \pm 3.69e+01$	+
f6	$\mathbf{3.59e+01 \pm 0.00e+00}$	$\mathbf{3.59e+01 \pm 0.00e+00}$	=	$\mathbf{3.59e+01 \pm 0.00e+00}$	=	$\mathbf{3.59e+01 \pm 0.00e+00}$	=
f7	$1.06e+02 \pm 1.03e+01$	$\mathbf{9.29e+01 \pm 0.00e+00}$	-	$9.32e+01 \pm 3.52e-01$	-	$1.10e+02 \pm 1.54e+01$	=
f8	$1.49e+02 \pm 1.65e-01$	$\mathbf{1.49e+02 \pm 0.00e+00}$	-	$\mathbf{1.49e+02 \pm 0.00e+00}$	-	$1.49e+02 \pm 1.85e-01$	=
f9	$1.24e+02 \pm 1.20e+00$	$\mathbf{1.24e+02 \pm 0.00e+00}$	-	$\mathbf{1.24e+02 \pm 0.00e+00}$	-	$1.27e+02 \pm 1.31e+01$	=
f10	$5.68e+03 \pm 2.92e+04$	$5.11e+02 \pm 5.64e+02$	=	$\mathbf{-1.51e+01 \pm 4.10e+01}$	-	$2.48e+02 \pm 1.08e+02$	=
f11	$1.65e+02 \pm 2.63e+01$	$\mathbf{7.63e+01 \pm 0.00e+00}$	-	$7.74e+01 \pm 1.40e+00$	-	$1.60e+02 \pm 2.68e+01$	=
f12	$-6.13e+02 \pm 1.78e+01$	$8.90e+03 \pm 1.82e+04$	+	$\mathbf{-6.21e+02 \pm 7.60e-01}$	=	$-6.13e+02 \pm 1.43e+01$	=
f13	$3.88e+01 \pm 1.10e+01$	$\mathbf{3.00e+01 \pm 0.00e+00}$	-	$3.00e+01 \pm 0.00e+00$	-	$4.26e+01 \pm 1.20e+01$	=
f14	$\mathbf{-5.23e+01 \pm 0.00e+00}$	$5.07e+00 \pm 4.97e+01$	+	$\mathbf{-5.23e+01 \pm 0.00e+00}$	=	$\mathbf{-5.23e+01 \pm 0.00e+00}$	=
f15	$1.07e+03 \pm 3.32e+01$	$\mathbf{1.00e+03 \pm 9.74e-01}$	-	$1.01e+03 \pm 4.18e+00$	-	$1.11e+03 \pm 7.04e+01$	+
f16	$7.80e+01 \pm 3.91e+00$	$\mathbf{7.14e+01 \pm 0.00e+00}$	-	$7.23e+01 \pm 7.78e-01$	-	$7.85e+01 \pm 4.42e+00$	=
f17	$-1.41e+01 \pm 1.01e+00$	$1.31e+01 \pm 2.36e+01$	+	$\mathbf{-1.69e+01 \pm 0.00e+00}$	-	$-1.08e+01 \pm 3.74e+00$	+
f18	$-8.25e+00 \pm 4.78e+00$	$6.55e+01 \pm 6.98e+01$	+	$\mathbf{-1.69e+01 \pm 0.00e+00}$	-	$8.59e-01 \pm 1.75e+01$	+
f19	$-1.00e+02 \pm 1.35e+00$	$-5.50e+01 \pm 2.54e+01$	+	$\mathbf{-1.02e+02 \pm 6.26e-01}$	-	$-9.81e+01 \pm 2.03e+00$	+
f20	$-5.46e+02 \pm 2.98e-01$	$2.54e+04 \pm 3.78e+04$	+	$\mathbf{-5.46e+02 \pm 3.03e-01}$	=	$-5.46e+02 \pm 3.40e-01$	=
f21	$4.98e+01 \pm 6.78e+00$	$4.22e+01 \pm 1.52e+00$	-	$\mathbf{4.19e+01 \pm 9.47e-01}$	-	$5.36e+01 \pm 1.15e+01$	=
f22	$-9.89e+02 \pm 1.35e+01$	$-9.17e+02 \pm 3.24e+00$	+	$\mathbf{-9.98e+02 \pm 3.51e-01}$	-	$-9.87e+02 \pm 1.46e+01$	=
f23	$7.98e+00 \pm 4.53e-01$	$\mathbf{6.96e+00 \pm 0.00e+00}$	-	$7.23e+00 \pm 2.91e-01$	-	$7.85e+00 \pm 5.05e-01$	=
f24	$1.68e+02 \pm 1.98e+01$	$\mathbf{1.12e+02 \pm 4.33e+00}$	-	$1.21e+02 \pm 4.43e+00$	-	$1.89e+02 \pm 3.98e+01$	+

Table 2. Average Fitness ± Standard Deviation and Wilcoxon rank-sum test on the Fitness (reference = S-3SOME) for 40D case.

	S-3SOME	*GCMAES*		SADE		*3SOME*	
f1	$\mathbf{7.95e+01 \pm 0.00e+00}$	$\mathbf{7.95e+01 \pm 0.00e+00}$	-	$\mathbf{7.95e+01 \pm 0.00e+00}$	-	$\mathbf{7.95e+01 \pm 0.00e+00}$	=
f2	$\mathbf{-2.10e+02 \pm 0.00e+00}$	$3.57e+02 \pm 2.30e+02$	+	$\mathbf{-2.10e+02 \pm 0.00e+00}$	-	$\mathbf{-2.10e+02 \pm 0.00e+00}$	=
f3	$-4.43e+02 \pm 5.15e+00$	$1.08e+02 \pm 8.31e+01$	+	$-4.24e+02 \pm 1.35e+01$	+	$\mathbf{-4.56e+02 \pm 2.97e+00}$	-
f4	$-4.38e+02 \pm 6.99e+00$	$1.29e+02 \pm 7.11e+01$	+	$-4.04e+02 \pm 2.86e+01$	+	$\mathbf{-4.51e+02 \pm 3.42e+00}$	-
f5	$\mathbf{-9.21e+00 \pm 0.00e+00}$	$2.16e+01 \pm 9.77e+00$	+	$\mathbf{-9.21e+00 \pm 0.00e+00}$	+	$\mathbf{-9.21e+00 \pm 0.00e+00}$	+
f6	$\mathbf{3.59e+01 \pm 0.00e+00}$	$\mathbf{3.59e+01 \pm 0.00e+00}$	-	$3.79e+01 \pm 2.92e+00$	+	$\mathbf{3.59e+01 \pm 0.00e+00}$	=
f7	$1.78e+02 \pm 3.02e+01$	$\mathbf{9.48e+01 \pm 3.13e+00}$	-	$1.30e+02 \pm 1.02e+01$	-	$2.15e+02 \pm 7.26e+01$	=
f8	$1.49e+02 \pm 3.59e-01$	$\mathbf{1.49e+02 \pm 0.00e+00}$	-	$1.82e+02 \pm 2.23e+01$	+	$1.50e+02 \pm 7.35e-01$	+
f9	$1.25e+02 \pm 1.60e+00$	$\mathbf{1.24e+02 \pm 0.00e+00}$	-	$1.58e+02 \pm 1.67e+00$	+	$1.25e+02 \pm 2.07e+00$	=
f10	$3.78e+05 \pm 2.03e+06$	$\mathbf{1.91e+02 \pm 1.12e+02}$	-	$6.09e+03 \pm 2.16e+03$	-	$9.73e+02 \pm 4.00e+02$	=
f11	$3.38e+02 \pm 4.95e+01$	$\mathbf{7.63e+01 \pm 0.00e+00}$	-	$1.16e+02 \pm 1.20e+01$	-	$3.67e+02 \pm 5.95e+01$	+
f12	$\mathbf{-6.14e+02 \pm 7.57e+00}$	$2.69e+03 \pm 2.68e+03$	+	$-6.13e+02 \pm 6.60e+00$	=	$-6.10e+02 \pm 9.48e+00$	=
f13	$4.00e+01 \pm 9.14e+00$	$\mathbf{3.00e+01 \pm 0.00e+00}$	-	$3.75e+01 \pm 6.50e+00$	=	$4.37e+01 \pm 1.28e+01$	=
f14	$\mathbf{-5.23e+01 \pm 0.00e+00}$	$5.01e+01 \pm 4.14e+01$	+	$\mathbf{-5.23e+01 \pm 0.00e+00}$	+	$\mathbf{-5.23e+01 \pm 0.00e+00}$	=
f15	$1.41e+03 \pm 1.04e+02$	$\mathbf{1.01e+03 \pm 2.40e+00}$	-	$1.06e+03 \pm 1.52e+01$	-	$1.99e+03 \pm 4.48e+02$	+
f16	$8.36e+01 \pm 5.33e+00$	$\mathbf{7.14e+01 \pm 0.00e+00}$	-	$8.31e+01 \pm 3.72e+00$	=	$8.94e+01 \pm 6.30e+00$	+
f17	$-1.02e+01 \pm 1.31e+00$	$1.47e+01 \pm 1.42e+01$	+	$\mathbf{-1.60e+01 \pm 4.13e-01}$	-	$-5.39e+00 \pm 3.51e+00$	+
f18	$7.02e+00 \pm 4.89e+00$	$2.18e+01 \pm 1.18e+01$	+	$\mathbf{-1.30e+01 \pm 1.25e+00}$	-	$2.67e+01 \pm 1.31e+01$	+
f19	$-9.67e+01 \pm 1.67e+00$	$-5.03e+01 \pm 1.29e+01$	+	$\mathbf{-1.01e+02 \pm 1.27e+00}$	-	$-9.48e+01 \pm 2.97e+00$	+
f20	$-5.46e+02 \pm 1.64e-01$	$3.66e+03 \pm 2.87e+03$	+	$-5.45e+02 \pm 1.98e-01$	+	$\mathbf{-5.46e+02 \pm 1.67e-01}$	=
f21	$5.08e+01 \pm 1.36e+01$	$\mathbf{4.16e+01 \pm 1.04e+00}$	-	$4.25e+01 \pm 2.30e+00$	-	$5.20e+01 \pm 1.71e+01$	=
f22	$-9.86e+02 \pm 1.00e+01$	$-9.14e+02 \pm 6.51e-01$	+	$-9.91e+02 \pm 8.09e+00$	-	$\mathbf{-9.91e+02 \pm 7.57e+00}$	-
f23	$8.19e+00 \pm 5.09e-01$	$\mathbf{7.62e+00 \pm 1.29e+00}$	-	$8.21e+00 \pm 7.58e-01$	=	$8.11e+00 \pm 7.06e-01$	=
f24	$5.45e+02 \pm 8.96e+01$	$\mathbf{1.50e+02 \pm 1.68e+01}$	-	$1.90e+02 \pm 1.21e+01$	-	$8.98e+02 \pm 2.53e+02$	+

Table 3. Average Fitness ± Standard Deviation and Wilcoxon rank-sum test on the Fitness (reference = S-3SOME) for 100D case.

	S-3SOME	*GCMAES*		SADE		*3SOME*	
f1	$\mathbf{7.95e+01 \pm 0.00e+00}$	$\mathbf{7.95e+01 \pm 0.00e+00}$	=	$\mathbf{7.95e+01 \pm 0.00e+00}$	=	$\mathbf{7.95e+01 \pm 0.00e+00}$	=
f2	$\mathbf{-2.10e+02 \pm 0.00e+00}$	$2.56e+02 \pm 1.01e+02$	+	$\mathbf{-2.10e+02 \pm 0.00e+00}$	=	$\mathbf{-2.10e+02 \pm 0.00e+00}$	=
f3	$-4.03e+02 \pm 9.30e+00$	$3.77e+02 \pm 1.57e+02$	+	$-2.92e+02 \pm 4.41e+01$	+	$\mathbf{-4.41e+02 \pm 7.57e+00}$	-
f4	$-3.88e+02 \pm 1.28e+01$	$3.83e+02 \pm 1.56e+02$	+	$-1.59e+02 \pm 7.73e+01$	+	$\mathbf{-4.27e+02 \pm 1.20e+01}$	-
f5	$\mathbf{-9.21e+00 \pm 0.00e+00}$	$1.18e+02 \pm 1.82e+01$	+	$\mathbf{-9.21e+00 \pm 0.00e+00}$	+	$1.04e+02 \pm 4.22e+02$	=
f6	$\mathbf{3.59e+01 \pm 0.00e+00}$	$\mathbf{3.59e+01 \pm 0.00e+00}$	-	$1.18e+02 \pm 3.97e+01$	+	$\mathbf{3.59e+01 \pm 0.00e+00}$	=
f7	$3.75e+02 \pm 9.17e+01$	$\mathbf{1.33e+02 \pm 9.67e+00}$	-	$3.63e+02 \pm 7.58e+01$	=	$5.89e+02 \pm 2.63e+02$	+
f8	$1.71e+02 \pm 3.06e+01$	$\mathbf{1.50e+02 \pm 1.59e+00}$	-	$2.79e+02 \pm 4.17e+01$	+	$1.71e+02 \pm 2.61e+01$	=
f9	$1.75e+02 \pm 2.08e+01$	$\mathbf{1.24e+02 \pm 1.13e+00}$	-	$2.31e+02 \pm 2.31e+01$	+	$1.77e+02 \pm 1.12e+01$	=
f10	$3.04e+03 \pm 8.61e+02$	$\mathbf{6.31e+01 \pm 5.68e+01}$	-	$4.57e+04 \pm 1.70e+04$	+	$3.03e+03 \pm 9.61e+02$	=
f11	$6.30e+02 \pm 8.95e+01$	$\mathbf{7.63e+01 \pm 0.00e+00}$	-	$1.79e+02 \pm 2.23e+01$	-	$3.68e+02 \pm 6.38e+01$	-
f12	$\mathbf{-6.19e+02 \pm 3.74e+00}$	$1.23e+03 \pm 1.04e+03$	+	$-6.15e+02 \pm 7.44e+00$	+	$-6.13e+02 \pm 9.46e+00$	+
f13	$3.66e+01 \pm 4.13e+00$	$\mathbf{3.00e+01 \pm 0.00e+00}$	-	$3.16e+01 \pm 2.64e+00$	-	$3.27e+01 \pm 2.88e+00$	-
f14	$\mathbf{-5.23e+01 \pm 0.00e+00}$	$2.59e+01 \pm 1.58e+01$	+	$\mathbf{-5.23e+01 \pm 0.00e+00}$	+	$\mathbf{-5.23e+01 \pm 0.00e+00}$	=
f15	$2.44e+03 \pm 2.77e+02$	$\mathbf{1.03e+03 \pm 1.16e+01}$	-	$1.33e+03 \pm 4.72e+01$	-	$4.49e+03 \pm 6.17e+02$	+
f16	$8.92e+01 \pm 3.32e+00$	$\mathbf{7.15e+01 \pm 0.00e+00}$	-	$9.75e+01 \pm 4.14e+00$	+	$9.33e+01 \pm 5.37e+00$	+
f17	$-8.00e+00 \pm 1.67e+00$	$5.16e+00 \pm 4.19e+00$	+	$\mathbf{-1.39e+01 \pm 4.73e-01}$	-	$6.39e-02 \pm 3.81e+00$	+
f18	$1.57e+01 \pm 5.86e+00$	$2.43e+01 \pm 1.77e+01$	=	$\mathbf{-5.43e+00 \pm 1.90e+00}$	-	$4.63e+01 \pm 1.53e+01$	+
f19	$-9.32e+01 \pm 2.39e+00$	$6.69e+00 \pm 1.57e+01$	+	$\mathbf{-9.98e+01 \pm 4.45e-01}$	-	$-8.99e+01 \pm 4.96e+00$	+
f20	$-5.46e+02 \pm 0.00e+00$	$2.45e+03 \pm 2.25e+03$	+	$-5.45e+02 \pm 1.94e-01$	+	$\mathbf{-5.46e+02 \pm 0.00e+00}$	-
f21	$5.36e+01 \pm 1.44e+01$	$\mathbf{4.30e+01 \pm 1.06e+00}$	-	$4.68e+01 \pm 6.22e+00$	-	$5.24e+01 \pm 1.02e+01$	=
f22	$-9.83e+02 \pm 1.32e+01$	$-9.14e+02 \pm 0.00e+00$	+	$\mathbf{-9.95e+02 \pm 5.84e+00}$	-	$-9.80e+02 \pm 1.46e+01$	=
f23	$8.31e+00 \pm 6.04e-01$	$8.59e+00 \pm 2.06e+00$	=	$9.33e+00 \pm 7.66e-01$	+	$\mathbf{8.27e+00 \pm 4.79e-01}$	=
f24	$1.66e+03 \pm 3.07e+02$	$\mathbf{3.66e+02 \pm 9.61e+01}$	-	$3.74e+02 \pm 3.33e+01$	-	$2.79e+03 \pm 4.45e+02$	+

that S-3SOME is highly competitive, especially in high dimensions, despite a much smaller requirement in terms of memory (as it perturbs only one solution) and overhead (as it does not make use of learning components or sophisticated structures).

Acknowledgments

This research is supported by the Academy of Finland, Akatemiatutkija 130600 and Tutkijatohtori 140487.

References

[1] A. Auger and N. Hansen. A restart CMA evolution strategy with increasing population size. In *Proc. IEEE Congress on Evolutionary Computation*, pages 1769–1776, 2005.

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

[3] R. Hooke and T. A. Jeeves. Direct search solution of numerical and statistical problems. *J. ACM*, 8(2):212–229, March 1961.

[4] G. Iacca, F. Neri, E. Mininno, Y. S. Ong, and M. H. Lim. Ockham's razor in memetic computing: Three stage optimal memetic exploration. *Inform. Sciences*, 188(1):17-43, 2012.

[5] J. J. Liang, A. K. Qin, P. N. Suganthan, and S. Baskar. Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. *IEEE T. Evolut. Comput.*, 10(3):281–295, 2006.

[6] N. Hansen, S. Finck, R. Ros, and A. Auger. Real-parameter black-box optimization benchmarking 2010: Noiseless functions definitions. Technical Report RR-6829, INRIA, 2010.

[7] F. Neri, C. Cotta, and P. Moscato. *Handbook of Memetic Algorithms*, volume 379 of *Studies in Computational Intelligence*. Springer, 2012.

[8] F. Neri, G. Iacca, and E. Mininno. Disturbed exploitation compact differential evolution for limited memory optimization problems. *Inform. Sciences*, 181(12):2469–2487, 2011.

[9] F. Peng, K. Tang, G. Chen, and X. Yao. Population-based algorithm portfolios for numerical optimization. *IEEE T. Evolut. Comput.*, 14(5):782–800, 2010.

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

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

[12] L. Y. Tseng and C. Chen. Multiple trajectory search for large scale global optimization. In *Proc. IEEE Congress on Evolutionary Computation*, pages 3052–3059, 2008.

[13] J. A. Vrugt, B. A. Robinson, and J. M. Hyman. Self-adaptive multimethod search for global optimization in real-parameter spaces. *IEEE T. Evolut. Comput.*, 13(2):243–259, 2009.

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

MEMETIC FIREFLY ALGORITHM FOR COMBINATORIAL OPTIMIZATION

Iztok Fister Jr., Iztok Fister, Janez Brest
Faculty of Electrical Engineering and Computer Science, University of Maribor, Slovenia
iztok.fister@guest.arnes.si; {iztok.fister; janez.brest}@uni-mb.si

Xin-She Yang
National Physical Lab, Teddington, London, UK
xin-she.yang@npl.co.uk

Abstract Firefly algorithms belong to modern meta-heuristic algorithms inspired by nature that can be successfully applied to continuous optimization problems. In this paper, we have been applied the firefly algorithm, hybridized with local search heuristic, to combinatorial optimization problems, where we use graph 3-coloring problems as test benchmarks. The results of the proposed memetic firefly algorithm (MFFA) were compared with the results of the Hybrid Evolutionary Algorithm (HEA), Tabucol, and the evolutionary algorithm with SAW method (EA-SAW) by coloring the suite of medium-scaled random graphs (graphs with 500 vertices) generated using the Culberson random graph generator. The results of firefly algorithm were very promising and showed a potential that this algorithm could successfully be applied in near future to the other combinatorial optimization problems as well.

Keywords: Firefly algorithm, Graph 3-coloring, Nature inspired algorithms, NP-complete problem, Swarm intelligence.

1. Introduction

Nature, especially biological systems, has always been an inspiration for those scientists who would like to transform some successful features of a biological system into computer algorithms for efficient problem solving. Birds, insects, ants, and fish may display some so-called, collective or swarm intelligence in foraging, defending, path finding, etc. This collective intelligence of self-organizing systems or agents has served as a basis for many good and efficient algorithms developed in the past,

e.g.: ant-colony optimization [7], particle swarm optimization [22], artificial bee colony [11, 12, 21], bacterial foraging [23]. Today, all these algorithms are referred to as *swarm intelligence*.

The firefly algorithm (FFA) belongs to swarm intelligence as well. It was developed by X. S. Yang [28]. This algorithm is based on the behavior of fireflies. Each firefly flashes its lights with some brightness. This light attracts other fireflies within the neighborhood. On the other hand, this attractiveness depends on the distance between the two fireflies. The closer the two fireflies are, the more attractive they will seem to other. In FFA, each firefly represents a point in a search space. When the attractiveness is proportional to the objective function the search space is explored by moving the fireflies towards more attractive neighbors.

FFA has displayed promising results when applied to continuous optimization problems [29, 30]. Conversely, within the area of combinatorial optimization problems, only a few papers have been published to date. Therefore, aim of this paper is to show that FFA can be applied to this kind of optimization problems as well. In this context, FFA for graph 3-coloring (3-GCP) has been developed. 3-GCP can informally be defined as follows: How to color a graph G with three colors so that none of the vertices connected with an edge is colored with the same color. The problem is NP-complete as proved by Garey and Johnson [16].

The most natural way to solve this problem is in a greedy fashion. Here, the vertices are ordered into a permutation and colored sequentially one after the other. However, the quality of coloring depends on the order in which the vertices will be colored. For example, the *naive* method orders the vertices of graph randomly. One of the best traditional heuristics for graph coloring today is DSatur by Brelaz [2], which orders the vertices v according to *saturation degree* $\rho(v)$. The saturation degree denotes the number of distinctly colored vertices to the vertex v.

This problem cannot be solved by an exact algorithm for graph instances of more than 100 vertices. Therefore, many heuristic methods have been developed for larger instances of graphs. These methods can be divided into local search [15] and hybrid algorithms [26]. One of the more successful local search heuristic was Tabucol developed by Herz and De Werra [19], who employed the tabu search proposed by Glover [17]. The most effective local search algorithms today are based on reactive partial local search [1, 25], adaptive memory [20], and variable search space [1]. On the other hand, various evolutionary algorithms have been hybridized using these local search heuristics. Let us refer to three such algorithms only: the hybrid genetic algorithm by Fleurent and Ferland [13], the hybrid evolutionary algorithm by Galinier and Hao [14], and the memetic algorithm for graph coloring by Lü and Hao [24].

Some modifications of the original algorithm need to be performed in order to apply FFA to 3-GCP. The original FFA algorithm operates on real-valued vectors. On the other hand, the most traditional heuristics act on the permutation of vertices. In order to incorporate the benefits of both, solutions of the proposed memetic FFA (MFFA) algorithm are represented as real-valued vectors. The elements of these vectors represent *weigths* that determine how hard the vertices are to color. The higher the weight is, the sooner the vertex should be colored. The permutation of vertices is obtained by sorting the vertices according to their weights. The DSatur traditional heuristic is used for construction of 3-coloring from this permutation. A similar approach was used in the evolutionary algorithm with the SAW method (EA-SAW) of Eiben et al. [8], and by the hybrid self-adaptive differential evolution and hybrid artificial bee colony algorithm of Fister et al. [10, 12]. Additionally, the *heuristical swap* local search is incorporated into the proposed MFFA. In order to preserve the current best solution in the population, the elitism is considered by this algorithm.

The results of the proposed MFFA algorithm for 3-GCP were compared with the results obtained with EA-SAW, Tabucol, and HEA by solving an extensive set of random medium-scale graphs generated by the Culberson graph generator [6]. The comparison between these algorithms shows that the results of the proposed MFFA algorithm are comparable, if not better, than the results of other algorithms used in the experiments.

The structure of this paper is as follows: In Section 2, the 3-GCP is discussed, in detail. The MFFA is described in Section 3, while the experiments and results are presented in Section 4. The paper is concluded with a discussion about the quality of the results, and the directions for further work are outlined.

2. Graph 3-Coloring

Let us suppose, an undirect graph $G = (V, E)$ is given, where V is a set of vertices $v \in V$ for $i = 1, \ldots, n$, and E denotes a set of edges that associate each edge $e \in E$ for $i = 1, \ldots, m$ to the unordered pair $e = \{v_i, v_j\}$ for $i = 1, \ldots, n$ and $j = 1, \ldots, n$. Then, the vertex 3-coloring (3-GCP) is defined as a mapping $c : V \rightarrow S$, where $S = \{1, 2, 3\}$ is a set of three colors and c a function that assigns one of the three colors to each vertex of G. A coloring $\mathbf{s}$ is *proper* if each of the two vertices connected with an edge are colored with a different color.

3-GCP can be formally defined as a constraint satisfaction problem (CSP). It is represented as a pair $\langle S, \phi \rangle$, where S denotes the search

space, in which all solutions $\mathbf{s} \in S$ are *feasible*, and ϕ a Boolean function on S (also a *feasibility condition*) that divides the search space into *feasible* and *unfeasible* regions. To each $e \in E$ the constraint b_e is assigned with $b_e(\langle s_1, \ldots, s_n \rangle) = true$ if and only if $e = \{v_i, v_j\}$ and $s_i \neq s_j$. Suppose that $B^i = \{b_e | e = \{v_i, v_j\} \wedge j = 1 \ldots m\}$ defines the set of constraints belonging to variable v_i. Then, the feasibility condition ϕ is expressed as a conjunction of all the constraints $\phi(\mathbf{s}) = \wedge_{v \in V} B^v(\mathbf{s})$.

As in evolutionary computation, constraints can be handled indirectly in the sense of a *penalty function*, that punishes the unfeasible solutions. The farther the unfeasible solution is from the feasible region, the higher is the penalty function. The penalty function is expressed as:

$$f(\mathbf{s}) = min \sum_{i=0}^{n} \psi(\mathbf{s}, B^i), \tag{1}$$

where the function $\psi(\mathbf{s}, B^i)$ is defined as:

$$\psi(\mathbf{s}, B^i) = \begin{cases} 1 & \text{if } \mathbf{s} \text{ violates at least one } b_e \in B^i, \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

Note that Eq. (1) also represents the objective function. On the other hand, the same equation can be used as a feasibility condition in the sense that $\phi(\mathbf{s}) = true$ if and only if $f(\mathbf{s}) = 0$. If this condition is satisfied a proper graph 3-coloring is found.

3. Memetic Firefly Algorithm for Graph 3-Coloring

The phenomenon of fireflies is that fireflies flash their lights that can be admired on clear summer nights. This light is produced by a complicated set of chemical reactions. Firefly flashes in order to attract mating partners and serve as a protection mechanism for warning off potential predators. Their light intensity I decreases when the distance r from the light source increases according to term $I \propto r^2$. On the other hand, air absorbs the light as the distance from the source increases.

When the flashing light is proportional to the objective function of the problem being optimized (i.e., $I(\mathbf{w}) \propto f(\mathbf{w})$), where $\mathbf{w}$ represents the candidate solution) this behavior of fireflies can represent the base for an optimization algorithm. However, artificial fireflies obey the following rules: all fireflies are unisex, their attractiveness is proportional to their brightness, and the brightness of a firefly is affected or determined by the landscape of the objective function.

These rules represent the basis on which the firefly algorithm acts [28]. The FFA algorithm is population-based, where each solution denotes a

point in the search space. The proposed MFFA algorithm is hybridized with a local search heuristic. In this algorithm, the solution is represented as a real-valued vector $\mathbf{w_i} = (w_{i,1}, \ldots, w_{i,n})$ for $i = 1 \ldots NP$, where NP denotes the size of population P. The vector $\mathbf{w_i}$ determines the weights assigned to the corresponding vertices. The values of the weights are taken from the interval $w_{i,j} \in [lb, ub]$, where lb and ub are the lower and upper bounds, respectively. The weights represent an initial permutation of vertices $\pi(\mathbf{v_i})$. This permutation serves as an input to the DSatur heuristic that obtains the graph 3-coloring. The pseudocode of the MFFA algorithm is illustrated in Algorithm 1.

Algorithm 1 Pseudo code of the MFFA algorithm

```
1: t = 0; fe = 0; found = FALSE; s* = ∅;
2: P^(t) = InitializeFFA();
3: while (!TerminateFFA(fe, found)) do
4:     fe += EvaluateFFA(P^(t));
5:     P' = OrderFFA(P^(t));
6:     found = FindTheBestFFA(P^(t), s*);
7:     P^(t+1) = MoveFFA(P^(t), P');
8:     t = t+1;
9: end while
```

As can be seen from Algorithm 1, the search process of the MFFA algorithm (statements within the **while** loop) that is controlled by generation counter t consists of the following functions:

- EvaluateFFA(): evaluating the solution. This evaluation is divided into two parts: In the first part, the solution $\mathbf{w_i}$ is transformed into a permutation of vertices $\pi(\mathbf{v_i})$ according to the non-decreasing values of the weights. In the second part, the permutation of vertices $\pi(\mathbf{v_i})$ is decoded into a 3-coloring $\mathbf{s}_i$ by the DSatur heuristic. Note that the 3-coloring $\mathbf{s}_i$ represents the solution of 3-GCP in its original problem space, where the quality of the solution is evaluated according to Eq. (1). Conversely, looking for a new solution is performed within the real-valued search space.

- OrderFFA(): forming an intermediate population P' by copying the solutions from the original population $P^{(t)}$ and sorting $P^{(t)}$ according to the non-decreasing values of the objective function.

- FindTheBestFFA(): determining the best solution in the population $P^{(t)}$. If the best solution in $P^{(t)}$ is worse than the $\mathbf{s}^*$ the later replaces the best solution in $P^{(t)}$, otherwise the former becomes the best solution found so far $\mathbf{s}^*$.

- MoveFFA(): moving the fireflies towards the search space according to the attractiveness of their neighbour's solutions.

Two features need to be developed before this search process can take place: the initialization (function InitializeFFA()) and termination (function TerminateFFA()). The population is initialized randomly according to the following equation:

$$w_{i,j} = (ub - lb) \cdot rand(0,1) + lb, \tag{3}$$

where the function rand$(0,1)$ denotes the random value from the interval $[0,1]$. The process is terminated when the first of the following two condition is satisfied: the number of objective function evaluations fe reaches the maximum number of objective function evaluations (MAX_FES) or the proper coloring is found ($found == true$).

The movement of i-th firefly is attracted to another more attractive firefly j, and expressed as follows:

$$\mathbf{w_i} = \mathbf{w_i} + \beta_0 e^{-\gamma r_{i,j}^2}(\mathbf{w_j} - \mathbf{w_i}) + \alpha(rand(0,1) - \frac{1}{2}), \text{ for } j = 1...n. \tag{4}$$

Note that the move of the i-th firefly is influenced by all the j-th fireflies for which $I[j] > I[i]$. As can be seen from Eq. (4), two summands are added to the current firefly position $\mathbf{w_i}$. The former reflects the attractiveness between firefly i and j (determined by $\beta(r) = \beta_0 e^{-\gamma r_{i,j}^2}$), while the latter is the randomized move in the search space (determined by the randomized parameter α). Furthermore, the attractiveness depends on the β_0 that is the attractiveness at $r = 0$, absorbtion coefficient γ, and Euclidian distance between the attracted and attracting firefly $r_{i,j}$.

3.1 The Heuristical Swap Local Search

After the evaluation step of the MFFA algorithm, the heuristical swap local search tries to improve the current solution. This heuristic is executed until the improvements are detected. The operation of this operator is illustrated in Fig. 1, which deals with a solution on G with 10 vertices. This solution is composed of a permutation of vertices $\mathbf{v}$, 3-coloring $\mathbf{s}$, weights $\mathbf{x}$, and saturation degrees ρ. The heuristical swap unary operator takes the first uncolored vertex in a solution and orders the predecessors according to the saturation degree descending. The uncolored vertex is swapped with the vertex that has the highest saturation degree. In the case of a tie, the operator randomly selects a vertex among the vertices with higher saturation degrees (1-opt neighborhood).

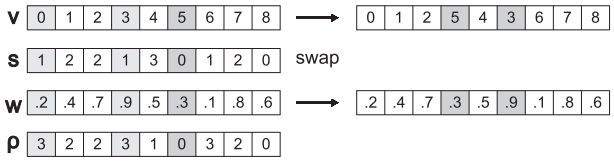


Figure 1. The heuristical swap unary operator.

In Fig. 1, an element of the solution corresponding to the first uncolored vertex 5 is in dark gray and the vertices 0 and 3 with the highest saturation degree are in light gray. From vertices 0 and 3, heuristical swap randomly selects vertex 3 and swaps it with vertex 5 (the right-hand side of Fig. 1).

4. Results

In the experimental work, the results of the proposed MFFA algorithm were compared with the results of: EA-SAW, HEA, and Tabucol. The algorithms used in the experiments were not selected coincidentally. In order to help the developers of new graph coloring algorithms, the authors Chiarandini and Stützle [4] made the code of Tabucol and HEA available within an online compendium [5]. On the other hand, this study is based on the paper of Eiben and al. [8], in which the authors proposed the evolutionary algorithm with SAW method for 3-GCP. The source code of this algorithm can also be obtained from Internet [18]. The goal of this experimental work was see whether the MFFA could also be applied to combinatorial optimization problems like 3-GCP.

The characteristics of the MFFA in the experiments were as follows. The population size was set at 500. The MAX_FES was fixed at 300,000 by all algorithms to make this comparison as fair as possible. All algorithms executed each graph instance 25 times. The algorithm parameters of MFFA were set as follows: $\alpha = 0.1$, $\beta_0 = 0.1$, and $\gamma = 0.8$. Note that these values of algorithm parameters optimize the performance of MFFA and were obtained during parameter tuning within the extensive experimental work. This parameter tuning satisfies the first perspective of parameter tuning, as proposed by Eiben and Smit [9], i.e., choosing parameter values that optimize the algorithm's performance. In order to satisfy the second perspective of the parameter tuning, i.e., how the MFFA performance depends on its parameter values, only the influence of the edge density was examined because of limited paper length.

The algorithms were compared according to the measures *success rate* (SR) and *average evaluations of objective function to solution* (AES). While the first measure represents the ratio between the number of suc-

cessfully runs and all runs, the second determines the efficiency of a particular algorithm. The aim of these preliminary experiments was to show that MFFA could be applied for 3-GCP. Therefore, any comparison of algorithms according to the time complexity was omitted.

4.1 Test-Suite

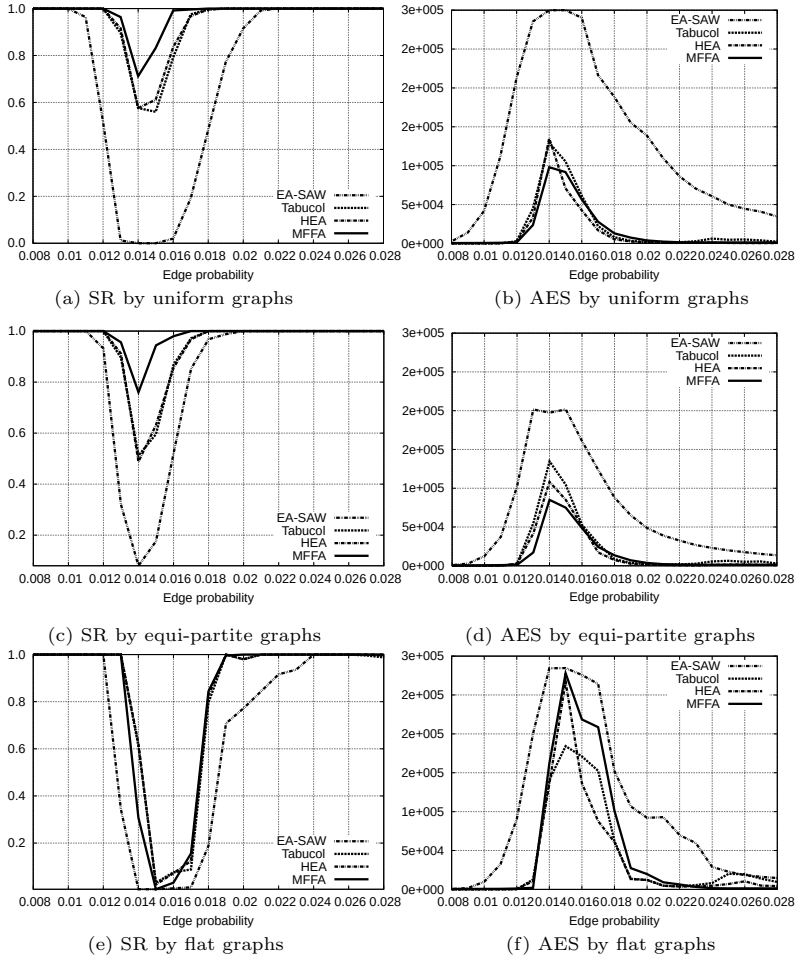
The test-suite, considered in the experiments, consisted of graphs generated with the Culberson random graph generator [6]. The graphs generated by this generator are distinguished according to type, number of vertices n, edge probability p, and seeds of random number generator q. Three types of graphs were employed in the experiments: *uniform* (random graphs without variability in sizes of color sets), *equi-partite*, and *flat*. The edge probabilities were varied in the interval $p \in 0.008 \ldots 0.028$ with a step of 0.001. Finally, the seeds were varied in interval $q \in 1 \ldots 10$ with a step of one. As a result, $3 \times 21 \times 10 = 630$ different graphs were obtained. That is, each algorithm was executed $15,750$ times to complete this experimental setup.

The experimental setup was selected so that a *phase transition* was captured. The phase transition is a phenomenon that accompanies almost all NP-hard problems and determines the region where the NP-hard problem passes over the state of "solvability" to a state of "unsolvability" and vice versa [31]. Typically, this region is characterized by ascertain problem parameter. This parameter is edge probability for 3-GCP. Many authors have determined this region differently. For example, Petford and Welsh [27] stated that this phenomenon occurs when $2pn/3 \approx 16/3$, Cheeseman et al. [3] when $2m/n \approx 5.4$, and Eiben et al. [8] when $7/n \leq p \leq 8/n$. In the presented case, the phase transition needed to be by $p = 0.016$ over Petford and Welsh, by $p \approx 0.016$ over Cheeseman, and between $0.014 \leq p \leq 0.016$ over Eiben et al.

4.2 Influence of the Edge Density

During this experiment, the influence of the edge density on the performance of the tested algorithms were investigated. The results are illustrated in Fig. 2. This figure consists of six diagrams corresponding to graphs of different types (uniform, equi-partite, and flat), and according to the measures SR and AES. In these diagrams, the average of those values accumulated after 25 runs are presented. Especially, we focus on the behavior of tested algorithms within the phase transition.

As can be seen in Fig. 2.a and Fig. 2.c, the results of MFFA outperformed the results of the other algorithms according to the measure SR on uniform and equi-partite graphs. HEA was slightly better than



(a) SR by uniform graphs (b) AES by uniform graphs

(c) SR by equi-partite graphs (d) AES by equi-partite graphs

(e) SR by flat graphs (f) AES by flat graphs

Figure 2. Results of algorithms for 3-GCP solving different types of random graphs.

Tabocol during the phase transition ($p \in [0.014, 0.016]$), whilst EA-SAW exposed the worst results within this region. As can be seen in Fig. 2.e, flat graphs were the hardest nuts to crack for all algorithms. Here, the results of MFFA according to measure SR were slightly worse but comparable to the results of HEA and Tabucol, whilst EA-SAW reported the worst results.

According to the measure AES (Fig. 2.b and Fig. 2.d), the best results were reported for MFFA by coloring the uniform and equi-partite graphs. On average, MFFA found solutions using a minimum number of evaluations. Note that the highest peak by $p = 0.014$ denotes the

hardest instance of uniform and equi-partite graphs to color for almost all the observed algorithms. Conversely, the graph with $p = 0.015$ was the hardest instance when coloring the flat graphs.

In summary, the proposed MFFA outperformed the results of HEA and Tabucol when coloring the uniform and equi-partite graphs, while by coloring the flat graphs it behaved slightly worse. The results of EA-SAW fell behind the results of the other tested algorithms for coloring all other types of graphs.

4.3 Discussion

The results of other parameter tuning experiments have been omitted because of limited paper length. Notwithstanding, almost four characteristics of MFFA could be exposed from the results of the last experiment. First, it is very important whether the movement of i-th firefly according to Eq. (4) is calculated from the position of the j-th firefly taken from the population $P^{(t)}$ or the intermediate population P', because the former incorporates an additional randomness within the search process. Thus, the results were significantly improved. Second, an exploration of the search space depends on the best solution in the population that directs the search process to more promising regions of the search space. As a result, this elitist solution needs to be preserved. Third, the local search serves as a search mechanism for detailed exploration of the basins of attraction. Consequently, those solutions that would normally be ignored by the regular FFA search process can be discovered. Fourth, the α parameter determines the size of the randomness move within the search space. Unfortunately, all conducted tests to self-adapt this parameter did not bear any improvement, at this moment. In summary, these preliminary results of MFFA encourage us to continue developing this algorithm in the future.

5. Conclusion

The results of MFFA showed that this algorithm could in future be a very promising tool for solving 3-GCP and, consequently, the other combinatorial optimization problems as well. In fact, it produced better results than HEA and Tabucol, when coloring the medium-scale uniform and equi-partite graphs. Unfortunately, this algorithm is slightly worse on flat graphs that remains the hardest to color for all the tested algorithms.

However, these good results could be misleading until further experiments on large-scale graphs (graphs with 1,000 vertices) are performed.

Fortunately, in the sense of preserving the obtained results, we have several ways for improving this MFFA in the future.

References

[1] I. Blöchliger and N. Zufferey. A graph coloring heuristic using partial solutions and a reactive tabu scheme. *Comput. Oper. Res.*, 35(3):960–975, 2008.

[2] D. Brelaz. New methods to color vertices of a graph. *Commun. ACM*, 22(4):251–256, 1979.

[3] P. Cheeseman, B. Kanefsky, and W. M. Taylor. Where the really hard problems are. In *Proc. International Joint Conference on Artificial Intelligence*, volume 1, pages 331–337, 1991.

[4] M. Chiarandini and T. Stützle. An aAnalysis of heuristics for vertex colouring. *Lect. Notes Comput. Sc.*, 6049:326–337, 2010.

[5] M. Chiandini and T. Stützle. Online compendium to the article: An Analysis of Heuristics for Vertex Colouring. `http://www.imada.sdu.dk/~marco/gcp-study/`. Accessed at 20 March 2012.

[6] J. Culberson. Graph Coloring Page. `http://www.ncbi.nlm.nih.gov`. Accessed at 20 March 2012.

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

[8] Á. E. Eiben, K. Hauw, and J. I. Hemert. Graph coloring with adaptive evolutionary algorithms. *J. Heuristics*, 4(1):25–46, 1998.

[9] Á. E. Eiben and S. K. Smit. Parameter tuning for configuring and analyzing evolutionary algorithms. *Swarm and Evolutionary Computation*, 1(1):19–31, 2011.

[10] I. Fister and J. Brest. Using differential evolution for the graph coloring. In *Proc. IEEE Symposium Series on Computational Intelligence*, pages 150–156, 2011.

[11] I. Fister, I. Fister Jr., J. Brest, and V. Žumer. Memetic Artificial bee colony algorithm for large-scale global optimization. In *Proc. IEEE Congress on Evolutionary Computation*, 2012.

[12] I. Fister Jr., I. Fister, and J. Brest. A hybrid artificial bee colony algorithm for graph 3-coloring. In *Proc. 11th International Conference on Artificial Intelligence and Soft Computing* , 2012.

[13] C. Fleurent and J. Ferland. Genetic and hybrid algorithms for graph coloring. *Ann. Oper. Res.*, 63(3):437–464, 1996.

[14] P. Galinier and J. Hao. Hybrid evolutionary algorithms for graph coloring. *J. Comb. Optim.*, 3(4):379–397, 1999.

[15] P. Galinier and A. Hertz. A survey of local search methods for graph coloring. *Comput. Oper. Res.*, 33(9):2547–2562, 2006.

[16] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman & Co., 1979.

[17] F. Glover. Future paths for integer programming and links to artificial intelligence. *Comput. Oper. Res.*, 13(5):533–549, 1986.

[18] J. I. Van Hemert. Jano's Homepage. `http://www.vanhemert.co.uk/csp-ea.html`. Accessed at 20 March 2012.

[19] A. Hertz and D. de Werra. Using tabu search techniques for graph coloring. *Computing*, 39(4):345–351, 1987.

[20] A. Hertz, M. Plumettaz, and N. Zufferey. Variable Space Search for Graph Coloring. *Discrete Appl. Math.*, 156(13):2551–2560, 2008.

[21] D. Karaboga and B. Basturk. A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. *J. Global Optim.*, 39(3):459–471, 2007.

[22] J. Kennedy and R. C. Eberhart. Particle swarm optimization. In *Proc. IEEE International Conference on Neural Networks*, volume 4, pages 1942–1948, 1995.

[23] Y. Liu and K.M. Passino. Biomimicry of social foraging bacteria for distributed optimization: Models, principles, and emergent behaviors. *J. Optimiz. Theory App.*, 115(3):603–628, 2002.

[24] Z. Lü and J. K. Hao. A memetic algorithm for graph coloring. *Eur. J. Oper. Res.*, 203(1):241–250, 2010.

[25] E. Malaguti, M. Monaci and P. Toth. A metaheuristic approach for the vertex coloring problem. *INFORMS J. Comput.*, 20(2):302–316, 2008.

[26] E. Malaguti and P. Toth. A survey on vertex coloring problems. *Intl. Trans. Oper. Res.*, 17(1):1–34, 2009.

[27] A. D. Petford and D. J. A. Welsh. A randomized 3-coloring algorithms. *Discrete Math.*, 74(1-2):253–261, 1989.

[28] X.-S. Yang. *Nature-Inspired Metaheuristic Algorithms*. Luniver Press, 2008.

[29] X.-S. Yang. Firefly Algorithm, Lévy Flights and Global Optimization. In M. Bramer, R. Ellis, and M. Petridis, editors, Springer London, *Research and Development in Intelligent Systems*, volume 26, pages 209–218, 2010.

[30] X.-S. Yang. Firefly algorithm, stochastic test functions and design optimization. *Int. J. Bio-Inspired Computation*, 2(2):78–84, 2010.

[31] J. S. Turner. Almost all k-colorable graphs are easy to color. *J. Algorithm.*, 9(1):63–82, 1988.

A DIVERSITY-GUIDED DISTRIBUTED EVOLUTIONARY ALGORITHM

Muhannad Hijaze, David W. Corne
Department of Computer Science, Heriot-Watt University, Edinburgh, UK
{mh132; d.w.corne}@hw.ac.uk

Abstract Evolutionary algorithms are consistently troubled by early convergence to suboptimal solutions, and this problem is just as prevalent with distributed evolutionary algorithms (dEAs). One of several approaches used to address this issue (but still not much explored, especially in dEAs) is the use of diversity-guided control, where population diversity is monitored and steps are taken to boost diversity when necessary. Here, we explore such a mechanism in a dEA context. We monitor the diversity of subpopulations, and migrate suitable chromosomes into them when their diversity is low. This builds on previous work where we have explored a number of topology and migration mechanisms; in the current work, we investigate whether diversity-guidance can provide additional benefits. Experimental results are presented on twelve difficult benchmark function optimization problems. The results suggest that diversity guidance provides additional benefits, especially in the higher dimensional cases, although the method does not outperform a recent alternative strategy, in which the migration of new chromosomes was based on monitoring progress, rather than diversity.

Keywords: Distributed evolutionary algorithm, Function optimisation, Parallel evolutionary algorithm.

1. Introduction

Evolutionary algorithms (EAs) continue to be successfully applied to a very wide range of problems. Their main advantage over classical optimization methods, especially in the context of hard combinatorial optimization or function optimization problems, lies in their ability to explore the search space effectively without committing rapidly to a single region of the search space. Nevertheless, most EAs do eventually converge, and in many cases this convergence happens too early for the problem at hand and leads to suboptimal solutions and waste of search effort. This drawback is usually called premature convergence.

Increasingly, however, researchers and practitioners are exploring distributed EAs (dEAs), which utilize parallel hardware, and tend to naturally involve aspects that delay the onset of premature convergence. Distributed EAs involve several independent subpopulations of chromosomes (candidate solutions), distributed over several processors, with occasional interactions between them. These interactions are usually *migrations*, which means that selected chromosomes move (or are copied) between subpopulations. In a typical dEA, the separate populations will evolve independently for some amount of time, or for a fixed number of generations, and then migration occurs. Between migrations, the populations continue to evolve independently. This type of distributed EA design has several beneficial properties, and is well known to be capable of outperforming standard EAs operating on a single processor without migration, and otherwise the same (e.g. in terms of genetic operators, total population size, and for the same number of total fitness evaluations) [5, 6, 8, 11, 20].

The improved performance of dEAs is commonly attributed to the fact that they are naturally better at maintaining diversity in the population. Nevertheless the standard approaches still tend to delay premature convergence, rather than eliminate it. In this paper, we explore the use of a diversity-guided adaptive mechanism in a distributed evolutionary algorithm context. In particular we explore whether this mechanism provides additional benefits when used together with a topology/migration scheme that was itself found to improve a basic dEA in our previous work [9, 10].

In the remainder, Section 2 briefly provides more background and detail on dEAs and on diversity guided control mechanisms. The algorithms we compare in this paper are described in Section 3, and experimental results on high-dimensional variants of twelve benchmark function optimization problems are then provided in Section 4, along with analysis of the results, and we conclude in Section 5.

2. Further Background and Details

2.1 Distributed Evolutionary Algorithms

The simplest kind of dEA is one that is distributed over several processors in a straightforward way, but otherwise a standard EA. Changes to the underlying EA in such cases are those necessary to enable exploitation of the parallel hardware. The main line of research in dEAs involves algorithms that establish semi-independent populations, called subpopulations or demes, which share information at regular intervals [14]. This sharing is usually *migration* of chromosomes between sub-

populations. There are very many potential alternatives for migration schemes. The common migration scheme is typically to send copies of good chromosomes from some populations directly to other populations. This amounts to balancing useful exploration in the search (since there are independent subpopulations, mostly non-interacting) with occasional exploitation (promoted by migration), in a way that is not achieved by standard single-population EA designs. This typically leads to improved performance, beyond the benefits obtainable simply by utilizing parallel hardware.

Distributed EAs are usually described as either fine-grained or coarse-grained, depending on the sizes of the subpopulations. A common approach, given prevailing hardware configurations, is the coarse grained model (e.g. [1, 2, 7, 11, 15, 16, 20]). Genitor [20] is a well-known example, in which the subpopulations are linked in a ring topology, and migrations occur between neighbouring nodes. A recent alternative is Alba et al's GD-RCGA [2], in which the populations are linked in a cube-based topology, in which subpopulations are at vertices, and each is linked to three others along the edges of the cube. GD-RCGA reported strong performance, and we have adopted elements of it in the current work.

In our previous work [9, 10] we respectively tested a simple dEA, which we call here dEAT2, with a specific population topology, and an adaptive migration version which we here call dEAA. The dEAT2 algorithm uses the following topology: 16 subpopulations are partitioned into 8 islands each containing two subpopulations. Each subpopulation regularly sends the master process updates of its best chromosome so far, and at specified intervals the master process migrates a copy of the globally best chromosome into the island whose current best chromosome was the worst among the islands. The worst island was that whose subpopulation's best chromosome so far was the worst of the 16. In [9], this approach was compared with other topologies, including that studied in [2], and "topology 2" (as in dEAT2) is the one which gave best performance, and is the one we use here in each of dEAT2, DGdEA and dEAA. In [10] we explored adaptive migration schemes, in which the simple migration approach was replaced by an approach that maintained a probability of migrating the current best chromosome to each island. This was based on the level of progress in each island, as measured by the number of updates of best chromosome received by the master process. The fewer updates in a recent period, the more this seems to indicate stagnation, and hence the higher the probability that the master chromosome will send the current best overall chromosome to this island, in attempt to rejuvenate its performance. The best performing approach

from [10] again used the topology of dEAT2, and used a window of size 100 - i.e. used the most recent 100 updates to determine migration probabilities. This is the algorithm we later refer to as dEAA.

2.2 Diversity-Guided Control

The basic idea of diversity-guided control in an EA is to monitor the diversity of the population, and take action when the diversity level has moved below a threshold. This action would be designed to lead to a boost in diversity – e.g. either certain parameters of the algorithm will be changed, or new candidate solutions are introduced into the population, or both. Mauldin [13] was among the first to describe the concept of controlling the diversity level of the population in such ways, to maintain a balance between exploration and exploitation. Prominent and successful examples of such approaches include the work of Ursem et al. [19, 18]. In [18], excellent results were achieved by using signals from population diversity monitoring, based on which the EA would switch between two quite distinct modes of operation, favouring exploitation and exploration respectively. Meanwhile, an interesting and recent approach is by [4]. In this approach, based on signals from diversity monitoring, "diversified artificial chromosomes" (DACs) are generated and introduced to boost diversity at particular times.

Meanwhile, in the context of dEAs, diversity-guided adaptation is less explored. This is probably because premature convergence is less often seen as an issue for dEAs, although, as we have argued, the problem still exists and is revealed when addressing particularly difficult or high-dimensional problems.

Methods for diversity guided control of search behaviour, such as those described briefly above, have shown significant benefit in single-processor EAs. However diversity-guided control in dEAs is an open issue that is little explored as yet. In this paper we therefore explore the use of a diversity-guided control mechanism in a dEA. We adopt Chang et al's approach [4], using "DACs" to inject new chromosomes into the population when diversity is low. Next, we therefore give more details of the approach taken from [4], and how we adapt it to our setting.

3. Diversity-Guided Control in a Distributed Evolutionary Algorithm

The approach to diversity-guided control in [4] centred on the use of "diversified artificial chromosomes" (DAC). These chromosomes are injected into the population when diversity is too low, and the injection stops when diversity has reached a suitable level. We need to define how

this process could operate within a dEA, which has several subpopulations interconnected by a certain topology. We also need to define a diversity metric, and we need to set thresholds of this metric for determining when to start and stop injecting DACs, and we need to define how we create the DACs. We deal with these issues in the following subsections.

3.1 Monitoring Diversity and Generating DACs

The basic element of diversity measurement is a distance metric that gives the distance between any two chromosomes. In common with [4], we use Hamming distance for this purpose. The Hamming distance between chromosomes c_1 and c_2 is the proportion of genes for which the values in c_1 and c_2 are different. We define population diversity as the mean Hamming distance over all distinct pairs of chromosomes in the population. This is updated continually in each subpopulation. Note that Hamming distance is highly preferable to Euclidean distance in situations such as ours in which many distance calculations need to be done repeatedly.

We use the terminology of [4] to describe Diversified Artificial Chromosomes (DACs). DACs emerge from an "archive pool" maintained by the master process. The master process, as we will see, continually receives updates from each island indicating a new (locally) best chromosome, and maintains an archive containing the most recent 100 such chromosomes (covering all islands). When the archive is full, the next updated chromosome received will overwrite the current worst in the archive, if it has better (or equal) fitness to this current worst. When it is determined that an island needs an injection of a DAC (when the diversity level of that island falls between the lower and upper thresholds), a single chromosome is selected from the archive pool, and injected into the island, where it will overwrite the current worst fit chromosome in each of the two subpopulations in that island. When injecting a DAC into island d, the choice of DAC to select from the archive is as follows: we draw a sample of 10 from the archive pool that came from islands other than d, and choose the one most distant in Hamming distance from the current best in d, breaking ties randomly.

Finally, when the diversity of an island falls below the lower threshold, the subpopulations in that island are both reinitialised by using the archive pool. This is done by using the same population regeneration mechanism used in the underlying EA, filling the new subpopulations with children of crossover and mutation bred from the archive pool.

3.2 Integrating DAC-Injection Based Diversity Control in the dEA

The overall flow of control in DGdEA is characterised follows.

- While each island evolves independently, every time the local best-so-far improves on one of the island's subpopulations, a copy is migrated to its neighbour, and to the master process.
- Every chromosome received by the master process is added by the master process into an archive pool, which will be used to generate diversified artificial chromosomes (DAC).
- Each island regularly calculates its diversity level and communicates this to the master process (an island's diversity is the minimum of the diversity levels for its two subpopulations)
- The master process selects DACs and sends them to all sub-populations with diversity level below d_{high}.
- When an island receives a DAC, it injects it into both of its subpopulations, and each overwrites the chromosome in that subpopulation with worst fitness.
- If a sub-population's diversity level drops below the minimum threshold level d_{low}, the subpopulation is reinitialised.

Effectively, each subpopulation in the DGdEA switches between three modes of operation. When its diversity level is above the predefined threshold d_{high}, it operates a standard evolutionary algorithm. In this standard mode, it communicates its diversity level and new best chromosome to the master process, but never receives DACs to inject. When its diversity level is below this threshold, but above d_{low}, it will continually receive fresh injections of DACs from the master process. If the diversity level falls below d_{low}, it will be reinitialised by breeding from the master's archive pool.

4. Comparative Algorithms, Test Problems and Experiments

In this section we compare three algorithms, DGdEA, dEAT2 and dEAA, as described in Section 2, with further specifics described in this section. The underlying evolutionary algorithm in all cases is as follows, largely following that used in [2]. We use a truncation selection strategy [15], in which, in each generation, the best 33% of the population

are retained, and, selecting only from this best 33%, crossover and mutation are employed to generate the remainder of the new population. We use the crossover and mutation operators used in [2]; these are the fuzzy connective-based crossover operators: F-Crossover, S-Crossover, L-Crossover, and M-Crossover, along with one-point, two-point, and uniform crossover. Each time a crossover operator is applied (crossover probability is 0.6), a choice between these seven operators is made uniformly at random. Mutation (Gaussian mutation of a single randomly chosen parameter) is performed on a child of crossover with probability 0.25.

All our algorithms use 15 individuals per subpopulation and, in dEAT2, a migration is performed every 25 generations. There is a predefined maximum number of generations (10000), but a trial run will terminate if it has reached the target fitness values. The physical hardware used is a cluster of eight personal computers running Microsoft widows XP Professional SP3, each one having an Intel Pentium IV 2.99 GHz processor and 2 GB of memory. The machines are interconnected by a Fast-Ethernet (100 Mbps) network. Comparisons are done against 12 standard benchmark optimisation problems; these are the standard and shifted versions of each of the Sphere, Rosenbrock, Rastrigin, Schwefel, Griewangk and Ackley functions, as described in detail in [17].

4.1 Experiments and Results

Initially, a variety of experiments were done to investigate the settings of the diversity thresholds, d_{low} and d_{high}. These tests were all done on the 50-dimensional versions of the test functions. The results of these experiments are summarized in Table 1.

Table 1. Summary of diversity threshold investigation on 50-D problems using DGdEA. Each entry represents the number of functions for the DGdEA outperformed dEAT2, when comparing means or totals over 20 trials.

d_{low}	0.1	0.1	0.1	0.1	0.2	0.2	0.2	0.2
d_{high}	0.5	0.6	0.7	0.8	0.5	0.6	0.7	0.8
Execution time	1	4	6	3	5	4	9	6
Success rate	0	3	6	2	2	2	8	5
d_{low}	0.3	0.3	0.3	0.3	0.4	0.4	0.4	0.4
d_{high}	0.5	0.6	0.7	0.8	0.5	0.6	0.7	0.8
Execution time	0	4	7	3	0	0	2	3
Success rate	0	0	7	4	0	0	3	2

For example, Table 1 shows that, when d_{low} and d_{high} were respectively 0.2 and 0.7, the DGdEA outperformed the dEAT2 on 9 of the 12 test functions in terms of execution time in successful trials, and in 8 out of the 12 test functions in terms of the success rate (i.e number of times in reach a trial reached the optimum fitness) over the 20 trials. The results clearly show that DGdEA is sensitive to the settings of d_{low} and d_{high}, while also showing that there are some clearly good settings that perform well over a variety of functions. The settings of $d_{low} = 0.2$ and $d_{high} = 0.7$ were chosen for the experiments reported next.

Table 2. Summary of results comparing dEAT2, dEAA and DGdEA on the 12 test functions. Entries give performance ranks (1 best, 3 worst).

dimension	30		50		100	
performance factor	ET	SR	ET	SR	ET	SR
dDEAT2	3	1	3	3	3	3
dDEAA	1	2	1	1	1	1
DGdEA	1	3	2	1	2	1

We compared dEAT2, dEAA, and DGdEA on each of 30-D, 50-D and 100-D versions of the 12 test functions. The results are summarised in Table 2, which (for space-saving purposes) provides a highly digested and simplified summary of many experiments. Each column indicates the ranking of dEAT2,dEAA and DGdEA in terms of their performance on either the 30D, 50D or 100D functions, and in terms of either mean execution time over 20 trials, or success rate over 20 trials.

It is interesting at first to see that the simplest approach, dEAT2, was most successful on the 30D problems in terms of success rate. This can be explained in two ways: first, dEAT2 is nevertheless a capable dEA variant with an effective migration strategy; second, each of dEAA and DGdEA use more sophisticated strategies which, as is typical of such augmentations, achieve greater success on more complex and difficult problems often at the expense of reduced performance on simpler problems. Next, we notice that dEAA and DGdEA are comparable on the 50D and 100D cases, with dEAA seeming to have the advantage, especially on the 100D functions. So, although the diversity guidance mechanism described here seems to have benefits over less sophisticated methods in a dEA context, it does not seem to outperform our previous approach in which migrations were done on the basis of a subpopulation's search progress. Speculating on the underlying reasons, we consider the case of a subpopulation that is continually finding new best fitnesses but is otherwise not very diverse (e.g. the population may be clustered around a steep local optimum in the landscape). In dEAA, this subpop-

ulation would be left alone while it continued to advance, however in DGdEA, it would be continually injected with DACs, which may dilute and slow down its progress.

5. Summary and Conclusions

Distributed evolutionary algorithms (dEAs) are of increasing interest and practical importance. We therefore need to understand how to design dEAs well, and address a large number of open questions and unexplored areas in dEA design. Here, we investigated a new dEA variant that uses a diversity-guidance scheme, inspired by one described for serial EAs in [4]. This diversity guidance scheme involves monitoring the diversity of subpopulations, and injecting chromosomes into them when diversity is too low. This was tested on 30, 50, and 100-dimensional versions of standard benchmark test functions. Results showed that, although the new diversity-guided dEA (DGdEA) was clearly more successful than a simpler dEA from earlier work, its performance was similar to but not better than an alternative approach which instead monitors the level of progress in subpopulations, and adapts the frequency of migration of best-so-far chromosomes into those subpopulations according to their level of progress.

In conclusion, we have shown that diversity-guidance can be beneficial in dEAs, but conclude that alternative mechanisms need to be explored. Meanwhile we have shown further evidence for the promise of adaptive migration based on progress in subpopulations, which was explored in earlier work.

References

[1] U. Aickelin and L. Bull. Partnering strategies for fitness evaluation in a pyramidal evolutionary algorithm. In *Proc. Genetic and Evolutionary Computation Conference*, pages 263–270, 2002.

[2] E. Alba, F. Luna, A. J. Nebro, and J. M. Troya. Parallel heterogeneous genetic algorithms for continuous optimization. *Parallel Comput.*, 30(5-6):699-719, 2004

[3] S. Baluja. Structure and performance of fine-grain parallelism in genetic search. In *Proc. 5th International Conference on Genetic Algorithms*, pages 155–162, 1993.

[4] P.-C. Chang, W.-H. Huang, and C.-J. Ting. Dynamic diversity control in genetic algorithm for mining unsearched solution space in TSP problems. *Expert Syst. Appl.*, 37(3):1863–1878, 2010.

[5] R. Collins and D. Jefferson. Selection in Massively Parallel Genetic Algorithms. In *Proc. 4th International Conference on Genetic Algorithms*, pages 249–256, 1991.

[6] Y. Davidor. A Naturally Occurring Niche & Species Phenomenon: The Model and First Results. In *Proc. 4th International Conference on Genetic Algorithms*, pages 257–263, 1991.

[7] V. S. Gordon and L. D. Whitley. Serial and parallel genetic algorithms as function optimizers. In *Proc. 5th International Conference on Genetic Algorithms*, pages 177–183, 1993.

[8] M. Gorges-Schleuter. Explicit parallelism of genetic algorithms through population structures. *Lect. Notes Comput. Sc.*, 496:150–159, 1991

[9] M. Hijaze and D. Corne. An investigation of topologies and migration schemes for asynchronous distributed evolutionary algorithms. In *Proc. World Congress on Nature and Biologically Inspired Computing*, 2009.

[10] M. Hijaze and D. Corne. Distributed evolutionary algorithm topologies with adaptive migration schemes. In *Proc. IEEE Congress on Evolutionary Computation*, pages 608–615, 2011.

[11] B. Manderick and P. Spiessens. Fine-grained parallel genetic algorithms. In *Proc. 3rd International Conference on Genetic Algorithms*, pages 428–433, 1989.

[12] T. Maruyama, T. Hirose, and A. Konagaya. A fine-grained parallel genetic algorithm for distributed parallel systems. In *Proc. 5th International Conference on Genetic Algorithms*, pages 184–190, 1993.

[13] M. L. Mauldin. Maintaining diversity in genetic search. In *Proc. National Conference on Artificial Intelligence*, pages 247–250, 1984.

[14] H. Muhlenbein. Parallel genetic algorithms, population genetics and combinatorial optimization. In *Proc. 3rd International Conference on Genetic Algorithms*, pages 416–421, 1989.

[15] H. Muhlenbein and D. Schlierkamp-Voosen. Predictive models for the breeder genetic algorithm i. continuous parameter optimization. *Evol. Comput.*, 1(1):25–49, 1993.

[16] T. Starkweather, L. D. Whitley, and K. E. Mathias. Optimization using distributed genetic algorithms. *Lect. Notes Comput. Sc.*, 496:176–185, 1991.

[17] P. N. Suganthan, N. Hansen, J. J. Liang, K. Deb, Y.-P. Chen, A. Auger, and S. Tiwari. Problem definitions and evaluation criteria for the CEC 2005 Special Session on Real Parameter Optimization. Technical Report, Nanyang Technological University, Singapore and KanGAL Report Number 2005005, Kanpur Genetic Algorithms Laboratory, IIT Kanpur, 2005. Available as `http://www.ntu.edu.sg/home/epnsugan/index_les/CEC-05/TechReport-May-30-05.pdf`.

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

[19] R. K. Ursem. Multinational evolutionary algorithms. In *Proc. Congress on Evolutionary Computation*, volume 3, pages 1633–1640, 1999.

[20] L. D. Whitley. The GENITOR algorithm and selection pressure: Why rank-based allocation of reproductive trials is best, In *Proc. 3rd International Conference on Genetic Algorithms*, pages 116–121, 1989.

ANALYSIS OF VEGA AND SPEA2 USING EXPLORATION AND EXPLOITATION MEASURES

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

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

Abstract This paper presents the analysis and comparison between Vector Evaluated Genetic Algorithm (VEGA) and Strength Pareto Evolutionary Algorithm 2 (SPEA2) using the recently introduced exploration and exploitation measures. The experimental results show that the inner workings of both multiobjective algorithms can be observed and explained in a refined way in terms of exploration and exploitation.

Keywords: Exploitation, Exploration, Multi-objective 0/1 knapsack problem.

1. Introduction

Exploration and exploitation are fundamental concepts of any search algorithm [4]. In this respect Evolutionary Algorithms (EAs) show remarkable balance between exploration and exploitation of the search space [7]. On analyzing the rationales of experimental results of EAs, exploration and exploitation have been often employed to show their leverage towards better performance. Although the terms exploration and exploitation have been widely used in the EA community, there has not been really a common agreement on the definitions of exploration and exploitation [4]. To our best knowledge exploration and exploitation have not been really quantified[1], and hence assessing the performance of EAs with regard to exploration and exploitation is mostly done in qualitative manners. Without proper definitions and precise quantitative

measurement, it is less objective to justify the impacts that exploration and exploitation bring to EAs.

To tackle these problems, we have recently introduced an ancestry tree approach to measure exploration and exploitation based on genealogy [2]. Our previous work focused on introducing the concepts and definitions of the approach. Also, some preliminary results that applied the approach to multiobjective 0/1 knapsack problem using Strength Pareto Evolutionary Algorithm 2 (SPEA2) [15] were covered. Our approach had not been applied to any other single- or multi-objective evolutionary algorithms. The main objective of this paper is to revisit two well-known multiobjective algorithms, Vector Evaluated Genetic Algorithm (VEGA) [10] and SPEA2, and analyze these algorithms using the newly introduced exploration and exploitation measures. Namely, the paper will utilize exploration and exploitation's perspectives to show why SPEA2 outperformed VEGA and encore the explanations of existing work (e.g., [13]).

The paper is organized as follows. Section 2 summarizes the ancestor-tree approach. Section 3 presents the experimental results of VEGA and SPEA2, followed by the conclusion in Section 4.

2. The Ancestry Tree Approach

Fitness landscaping [5, 11] is a popular topic for observing and analyzing EAs with a number of visual tooling supports available [6]. It can be defined by three mathematical objects, including the landscape underlying the hypergraph whose vertices are the elements from configurations with values evaluated by fitness function and whose edges are specified by move operators of EAs [11]. Different from fitness landscaping, a primary objective of the ancestry tree approach [2] is, from the perspectives of exploration and exploitation [3] rather than fitness, to quantitatively define exploration and exploitation associated metrics derived from the evolution history represented in a tree structure. Namely, the ancestry tree approach stores important historical information including an individual's parent, its representation (genome), how (e.g., by mutation or crossover) and when (e.g., generation or fitness evaluation) the individual was created, and how far an individual is from its parent. For example, in Fig. 1, the root of the tree shows that the 8^{th} individual (the 2^{nd} digit in the left box of the root) is introduced at generation 0 (the 1^{st} digit in the left box of the root). After reproduction and selection, three offsprings are produced from this individual at different generations, where c means reproduction by crossover and the number within the parenthesis is the hamming distance from its parent. An important and novel concept re-

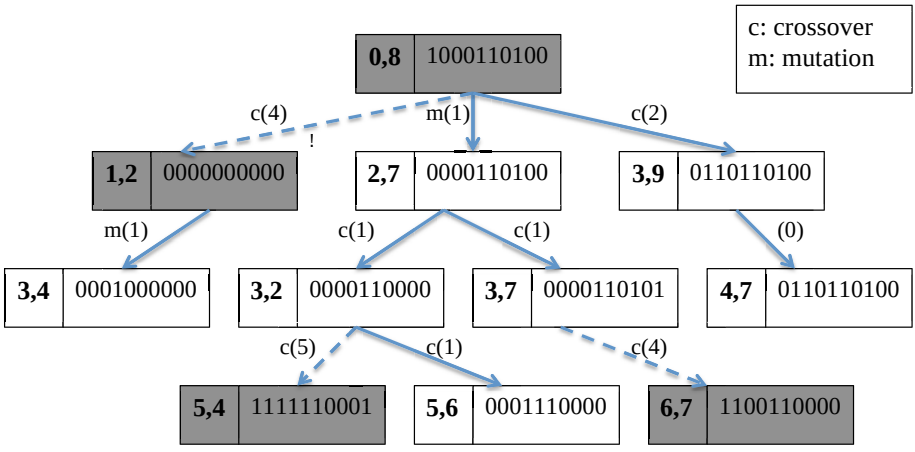


Figure 1. An ancestry tree example.

lated to exploration and exploitation, called splitting process, is shown as dash lines. A splitting process that splits the ancestry tree into several subtrees is triggered when the distance between parent and offspring is farther than a predefined threshold, which determines the boundaries of exploitation and exploration regions from the perspective of an individual. Namely, when a newly generated offspring is farther than the threshold value from its parent, it is exploring an unvisited region and the tree is split. Conversely, when an offspring is generated within the threshold distance, it is exploiting a previously visited region and no split will occur. With splitting processes, nodes in an ancestry tree can be categorized into exploration and exploitation nodes: Exploration nodes are those generated by the splitting processes, shown as gray nodes in Fig. 1; Exploitation nodes, on the other hand, are generated without splitting processes and are shown as white nodes. Exploration trees can be identified by connecting all the gray nodes from the same ancestry tree together, whilst exploitation trees are the subtrees with white nodes except that all their roots are exploration nodes. For example, there exists an exploration tree (i.e., {(0,8),(1,2),(5,4),(6,7)}) and four exploitation trees (i.e., {{(0,8),(2,7),(3,2),(3,7),(3,9),(4,7),(5,6)}, {(1,2),(3,4)}, {(5,4)}, {(6,7)}}) in Fig. 1 given threshold = 4. With such, different exploration and exploitation measures can be defined and computed by simply identifying various tree characteristics. For example, *exploreRatio* computes the number of root nodes of exploitation trees (i.e., exploration nodes) over all the nodes generated during the evolution process, whilst *exploitRatio* computes the number of intermediate and leaf nodes of ancestry trees (i.e., exploitation nodes) over all the

nodes. Due to space limitations, Section 3 will only define the measures appeared in experimental results. Please refer to [2] for more details about the approach and associated measures. Note that both distance and threshold are problem-dependent. This paper chooses hamming distance because of the nature of the the multiobjective 0/1 knapsack problem to be investigated in the next section. The threshold value is chosen from the best value among a large number of empirical results.

3. Experimental Results

The aim of this section is to apply the exploration and exploitation measures on two existing algorithms and to show the usefulness of such measures in understanding how the search space has been explored and exploited. We choose to present the proposed measures on the multiobjective 0/1 knapsack problem, which is defined as follows: Given a set of m items and a set of n knapsacks, where c_i is capacity of knapsack i, $p_{i,j}$ is a profit and $w_{i,j}$ is weight of item j according to knapsack i. The task is to find a vector $x = (x_1, ..., x_m) \in \{0,1\}^m$, such that $\forall i \in 1, ..., n : \sum_{j=1}^{m} w_{i,j} \cdot x_j \leq c_i$ and for which $f(x) = (f_1(x), ..., f_n(x))$ is maximum, where $f_i(x) = \sum_{j=1}^{m} p_{i,j} \cdot x_j$.

A comparison of several multiobjective EAs (e.g., VEGA and SPEA) on the multiobjective 0/1 knapsack problem has been also presented and discussed in [13], where the importance of elitism in multiobjective evolutionary optimization was shown. Our aim is not to repeat these studies, but to analyze explorative and exploitative power of two arbitrary EAs employing the newly proposed exploration and exploitation measures [2]. Since both VEGA and SPEA have been extensively studied before, we choose them for our study as well, although we include an improved version of SPEA called SPEA2 [15]. Our upcoming plan is to compare a number of more recent multi-objective algorithms. VEGA [10] uses the criterion selection strategy, where the search is guided in multiple directions by switching among the objectives, while SPEA2 [15] is an elitist Pareto-based strategy that uses an external secondary population to store the non-dominated solutions found so far. It is also a niching technique where niche is the portion of search space covered by a non-dominated solution.

First, the results of exploration and exploitation measures are presented on the multiobjective 0/1 knapsack problem, where $m = 100$ (number of items) and $n = 2$ (number of knapsacks). The VEGA and SPEA2 control parameters were, according to the recommendations in [14], set to: $runs = 30$, $G = 250$, $pop\_size = 100$, and $p_c = 0.8$, while probability of mutation was varied $p_m = \{0.000625, 0.00125, 0.0025, 0.005,$

Table 1. Comparison with the C metrics.

		SPEA2			VEGA			
	p_m	0.01	0.005	0.02	0.0025	0.005	0.00125	#
SPEA2	0.01	0	0.4010 ±0.18	0.4387 ±0.20	0.7610 ±0.27	0.8326 ±0.17	0.8255 ±0.19	5
	0.005	0.2880 ±0.16	0	0.3500 ±0.19	0.7559 ±0.24	0.8074 ±0.18	0.8284 ±0.19	4
	0.02	0.2658 ±0.15	0.3416 ±0.18	0	0.7010 ±0.24	0.7485 ±0.22	0.7725 ±0.23	3
VEGA	0.0025	0.0245 ±0.06	0.0217 ±0.04	0.0516 ±0.08	0	0.3619 ±0.26	0.3659 ±0.31	2
	0.005	0.0195 ±0.03	0.0257 ±0.04	0.0338 ±0.04	0.2549 ±0.25	0	0.3741 ±0.30	1
	0.00125	0.0216 ±0.03	0.0289 ±0.05	0.0457 ±0.07	0.2547 ±0.25	0.3231 ±0.29	0	0
	#	5	4	3	2	1	0	

$0.01, 0.02, 0.04, 0.08, 0.16\}$. The best results for VEGA on this particular problem were obtained with $p_m = 0.0025$ followed by $p_m = 0.005$ and $p_m = 0.00125$, while SPEA2 produces the best results when $p_m = 0.01$ followed by $p_m = 0.005$ and $p_m = 0.02$. For comparing the performance of VEGA and SPEA2, we used C measure, coverage of two sets, defined in [13] as:

$$C(X', X'') := \frac{|\{a'' \in X''; \exists a' \in X' : a' \succeq a''\}|}{|X''|}, \tag{1}$$

where $\succeq$ denotes well-known cover relation [14].

If $C(X', X'') = 1$, the set of solutions in X'' is completely covered by set of solutions in X'. Since $C(X', X'')$ is not necessarily equal to $1-C(X'', X')$, both $C(X', X'')$ and $C(X'', X')$ need to be compared. Table 1 shows the comparison of VEGA and SPEA2 on C measure, where SPEA2 clearly outperforms VEGA. The results do not surprise us since Zitzler et al. [13] already showed superiority of SPEA over VEGA on the same problem. SPEA2 ($p_m = 0.01$) found more solutions which are better distributed over Pareto front, while VEGA ($p_m = 0.0025$) found only a few solutions, which are mostly dominated by SPEA2 solutions. We were interested in seeing if the newly developed exploration and exploitation measures can provide satisfactory explanations for such results.

In Tables 2 and 3, exploration and exploitation measures for VEGA and SPEA2 are presented, where the threshold value was set to $x = 4$. Whenever the difference between an individual and its dominant parent is smaller than four bits, the new solution is still in the neighborhood of the dominant parent and credit is given to exploitation. Due to space limitation we only present results for $p_m = \{0.00125, 0.0025, 0.005, 0.01, 0.02\}$. Note that those results are the most important and interesting anyway.

Table 2. VEGA exploration and exploitation measures (*Prog.* = Progressiveness, *Sel.P.* = Selection Pressure, *nRRatio* = non-Revisited Ratio).

m \ p_m	0.00125	**0.0025**	0.005	0.01	0.02
Exploration					
Ratio	0.1835 ± 0.02	$\mathbf{0.2520 \pm 0.02}$	0.3845 ± 0.01	0.5795 ± 0.02	0.7936 ± 0.01
Type(c)	0.6798 ± 0.01	$\mathbf{0.6130 \pm 0.01}$	0.5459 ± 0.00	0.4819 ± 0.00	0.4253 ± 0.00
Type(m)	0.1716 ± 0.01	$\mathbf{0.2522 \pm 0.01}$	0.3287 ± 0.00	0.3999 ± 0.00	0.4540 ± 0.00
Type(r)	0.1333 ± 0.01	$\mathbf{0.1248 \pm 0.01}$	0.1194 ± 0.00	0.1146 ± 0.00	0.1183 ± 0.00
Gap	13.7317 ± 2.74	$\mathbf{11.5149 \pm 1.57}$	7.8763 ± 1.33	4.9137 ± 0.95	2.9350 ± 0.38
Prog.	14.5140 ± 2.17	$\mathbf{17.9963 \pm 3.27}$	24.3086 ± 3.62	29.9970 ± 4.82	32.9172 ± 6.27
Sel.P.	0.5653 ± 0.01	$\mathbf{0.5531 \pm 0.01}$	0.5124 ± 0.01	0.4590 ± 0.01	0.4139 ± 0.01
Exploitation					
Ratio	0.8165 ± 0.02	$\mathbf{0.7480 \pm 0.02}$	0.6155 ± 0.01	0.4205 ± 0.02	0.2064 ± 0.01
Type(c)	0.6125 ± 0.01	$\mathbf{0.5634 \pm 0.01}$	0.4900 ± 0.01	0.3917 ± 0.01	0.2712 ± 0.01
Type(m)	0.0701 ± 0.00	$\mathbf{0.1205 \pm 0.00}$	0.1922 ± 0.00	0.2801 ± 0.01	0.3560 ± 0.02
Type(r)	0.0591 ± 0.00	$\mathbf{0.0663 \pm 0.00}$	0.0747 ± 0.00	0.0814 ± 0.00	0.0827 ± 0.00
Prog.	18.0207 ± 3.92	$\mathbf{12.0061 \pm 2.11}$	8.3876 ± 1.58	6.4425 ± 1.32	6.3498 ± 1.15
Sel.P.	0.2967 ± 0.01	$\mathbf{0.2697 \pm 0.01}$	0.2176 ± 0.01	0.1418 ± 0.01	0.0610 ± 0.00
nRRatio	0.3774 ± 0.02	$\mathbf{0.5016 \pm 0.02}$	0.6548 ± 0.01	0.7973 ± 0.01	0.9032 ± 0.01

From Table 2 we notice that by changing probability of mutation p_m the ratio of exploration/exploitation in VEGA is increased/decreased very fast. When $p_m = 0.00125$, VEGA is more like a hill climbing algorithm, since the exploitation ratio prevails ($ExploitRatio = 0.8165$) and exploration ratio is minimal ($ExploreRatio = 0.1835$). When $p_m = 0.02$, VEGA acts more like a random search ($ExploreRatio = 0.7936$, $ExploitRatio = 0.2064$). Such drastic changes might indicate that parameter tuning can be a tricky task, and good parameter settings might be overlooked. Less striking changes are exhibited in SPEA2 (Table 3). When $p_m = 0.00125$, $ExploreRatio = 0.2303$ and $ExploitRatio = 0.7697$, while $p_m = 0.02$ $ExploreRatio = 0.5977$ and $ExploitRatio = 0.4023$. The best results for VEGA are obtained when $p_m = 0.0025$ demonstrating $ExploreRatio = 0.2520$ and $ExploitRatio = 0.7480$, while the best results for SPEA2 are obtained when $p_m = 0.01$ demon-

Table 3. SPEA2 exploration and exploitation measures (*Prog.* = Progressiveness, *Sel.P.* = Selection Pressure, *nRRatio* = non-Revisited Ratio).

m \ p_m	0.00125	0.0025	0.005	**0.01**	0.02
Exploration					
Ratio	0.2303 ± 0.03	0.2566 ± 0.03	0.3263 ± 0.03	$\mathbf{0.4213 \pm 0.03}$	0.5977 ± 0.02
Type(c)	0.5586 ± 0.01	0.5138 ± 0.01	0.4641 ± 0.01	$\mathbf{0.4120 \pm 0.01}$	0.3705 ± 0.00
Type(m)	0.1009 ± 0.01	0.1665 ± 0.01	0.2441 ± 0.01	$\mathbf{0.3232 \pm 0.00}$	0.3818 ± 0.00
Type(r)	0.3303 ± 0.01	0.3114 ± 0.01	0.2859 ± 0.00	$\mathbf{0.2607 \pm 0.00}$	0.2449 ± 0.00
Gap	63.5962 ± 9.75	66.4281 ± 7.12	64.8193 ± 6.74	$\mathbf{61.5735 \pm 7.05}$	54.8933 ± 5.88
Prog.	10.2888 ± 1.49	10.0457 ± 1.32	10.6611 ± 1.55	$\mathbf{11.3856 \pm 1.46}$	12.6322 ± 1.91
Sel.P.	0.8946 ± 0.01	0.9044 ± 0.01	0.9176 ± 0.01	$\mathbf{0.9281 \pm 0.00}$	0.9396 ± 0.00
Exploitation					
Ratio	0.7697 ± 0.03	0.7434 ± 0.03	0.6737 ± 0.03	$\mathbf{0.5787 \pm 0.03}$	0.4023 ± 0.02
Type(c)	0.5961 ± 0.01	0.5522 ± 0.01	0.4887 ± 0.01	$\mathbf{0.4213 \pm 0.01}$	0.3627 ± 0.01
Type(m)	0.0800 ± 0.00	0.1438 ± 0.00	0.2335 ± 0.00	$\mathbf{0.3420 \pm 0.00}$	0.4438 ± 0.00
Type(r)	0.1335 ± 0.01	0.1409 ± 0.01	0.1494 ± 0.01	$\mathbf{0.1557 \pm 0.00}$	0.1548 ± 0.00
Prog.	2.6963 ± 0.50	2.8138 ± 0.43	2.4192 ± 0.28	$\mathbf{2.1366 \pm 0.25}$	1.6657 ± 0.10
Sel.P.	0.7432 ± 0.03	0.7157 ± 0.03	0.6470 ± 0.03	$\mathbf{0.5539 \pm 0.03}$	0.3837 ± 0.02
nRRatio	0.2698 ± 0.03	0.3489 ± 0.03	0.4692 ± 0.02	$\mathbf{0.6393 \pm 0.02}$	0.8312 ± 0.02

strating $ExploreRatio = 0.4213$ and $ExploitRatio = 0.5787$. We can notice that the achieved balance between exploration and exploitation in SPEA2 is much better than in VEGA resulting in better found Pareto front. By increasing p_m to 0.005 in VEGA, balance between exploration and exploitation is better ($ExploreRatio = 0.3845$ and $ExploitRatio = 0.6155$). Unfortunately, the overall solutions are worst (Table 1). Hence, simply increasing mutation rate to increase exploration power does not work in VEGA. It seems that VEGA searches the space by small improvements (other characteristics of exploitation trees also support this view), and hence it needs high exploitation power. A designer of VEGA algorithm faces a problem of how to increase the explorative power of the algorithm, while not losing too much in exploitation. It is only our speculation that multi-process driven approaches may be favorable, where balance between exploration and exploitation would be synergistically coordinated by selection, mutation and crossover. Indeed, according to Zitzler et al.'s findings [13], introduction of elitism together with increasing probability of mutation can be helpful.

From Tables 2 and 3 we notice that various operators (c - crossover, m - mutation, r - repair) contribute equally to exploration and exploitation ($Type(T)$ represents the ratio that operator T participates during the evolution process). For example, the contribution of crossover in VEGA ($p_m = 0.0025$) to exploration is 0.6130 and to exploitation is

0.5634. Similarly, the contribution of crossover in SPEA2 ($p_m = 0.01$) to exploration is 0.4120 and to exploitation is 0.4213. Obviously, we can not proclaim crossover as being exclusively an exploration operator. We also notice that by increasing probability of mutation p_m, mutation contributes more to exploration, while its influence to exploitation is also increasing.

Regarding progressiveness, measuring the average depth of an exploration/exploitation tree to represent the progress of exploration/exploitation, we notice that VEGA exhibits higher progressiveness of exploitation, indicating that VEGA improves the results in many small steps, staying most of the time in the neighborhood of dominant parents. Such a feature may not be beneficial for multiobjective optimization. Progressiveness of exploitation in SPEA2 is surprisingly small and stable (mostly independent of p_m variations) indicating that better solutions are quickly found outside of the current neighborhood. While progressiveness of exploration in VEGA is bigger than in SPEA2, the latter has a much higher exploration gap (the average number of generations to identify new and unexplored parts of the search space) than the former. In VEGA, moving to new exploration regions happened in average after 11.5149 generations when $p_m = 0.0025$, while SPEA2 can discover new regions in average even after 61.5735 generations ($p_m = 0.01$). This can be contributed to the use of the archive in SPEA2.

The influence of selection pressure (calculated from trees' wideness – number of leafs) to exploration and exploitation is very different in both algorithms. The width of exploration and exploitation trees in VEGA is much smaller than in SPEA2. The explanation is that SPEA2 is much better than VEGA at exploring and exploiting the region around dominant parents, which is selected several times, by searching in different directions. Here, the influence of SPEA2 elitism is also clearly visible. Lastly, SPEA2 ($p_m = 0.01$) revisits fewer solutions in average than VEGA ($p_m = 0.0025$).

To visualize how VEGA and SPEA2 search the space, we repeat the experiment on a smaller multiobjective 0/1 knapsack problem ($m = 18$, $n = 2$). Figure 2 represents the search space after running VEGA and SPEA2 100 generations, where a blue point represents a solution obtained by exploitation, and a red point is obtained by exploration. Figure 2 clearly indicates that SPEA2 searched the space much better than VEGA, and more solutions are derived by exploration in SPEA2. From Fig. 2 we notice that some Pareto solutions were not discovered even though some points obtained by exploitation and exploration were close. The search in VEGA around dominant parents were not exhaustive enough, or dominant parents were not selected frequently enough,

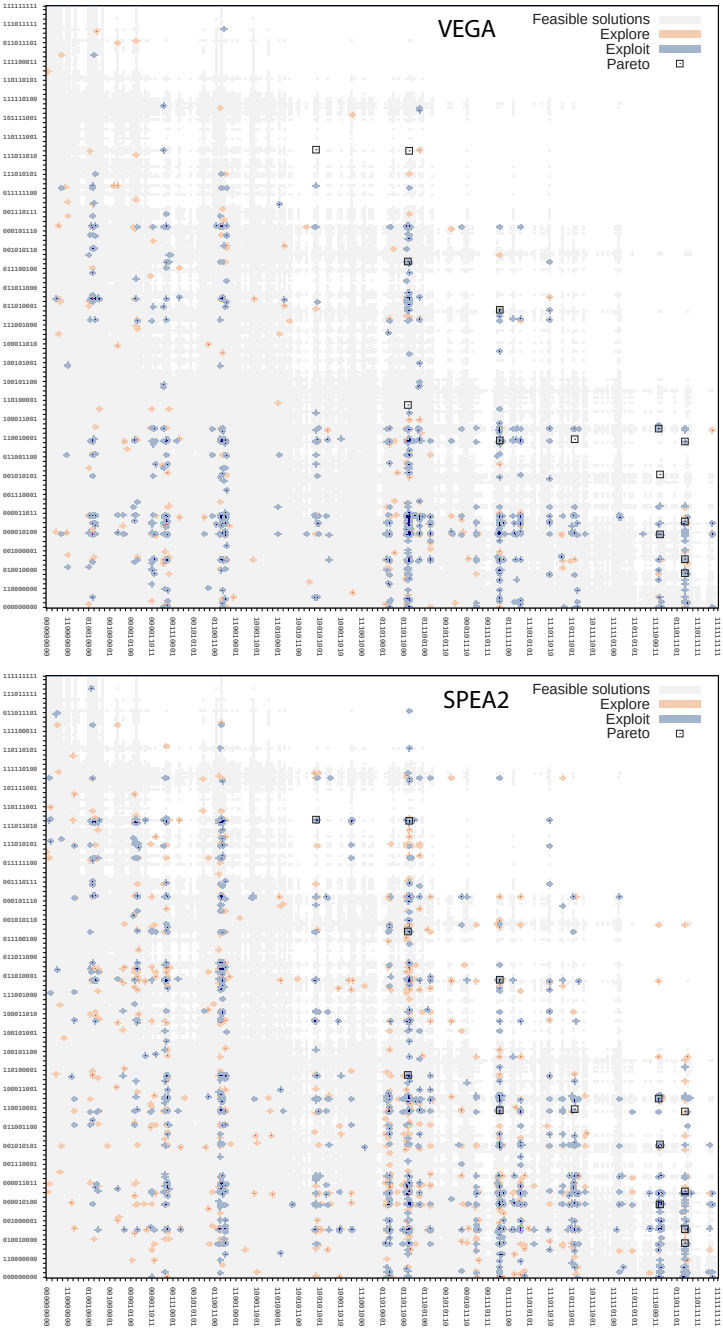


Figure 2. Search space VEGA/SPEA2.

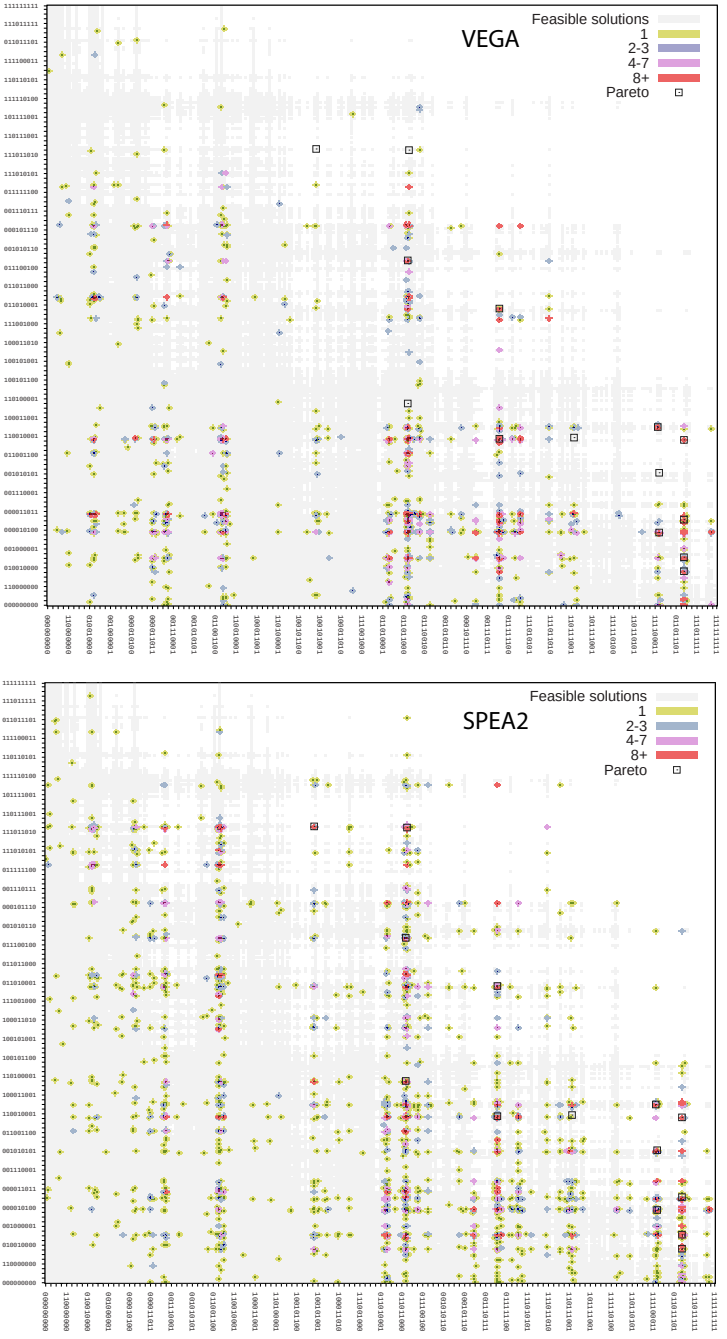


Figure 3. Revisited solutions in VEGA/SPEA2.

to find Pareto solutions. This can be rectified by the introduction of a local search or an improved selection mechanism. On the other hand, SPEA2 does not exhibit such shortcomings.

Even though the number of fitness evaluations were the same in both cases (VEGA and SPEA2), the space was not searched equally well. SPEA2 covered the search space much better than VEGA. Intuitively this means that more solutions were revisited in VEGA than in SPEA2. Figure 3 clearly supports this assumption. All non-green points represent solutions which were visited more than once during the search. It can be seen that VEGA exploration power is weak and several dominated solutions are revisited multiple times, while SPEA2 revisited mostly non-dominated solutions. Overall, we are convinced that the proposed exploration and exploitation measures provide new insights of particular algorithms' inner workings which might lead toward elimination of their weaknesses and to the design of better EAs.

4. Conclusions

Exploration and exploitation measures [2] are relatively new metrics in the evolutionary computation community. This paper utilizes such measures to analyze and compare VEGA and SPEA2. By using the measures readers may find finer-grained explanations on why SPEA2 outperforms VEGA. For example, higher values of $Progressiveness$ and $exploitRatio$ indicate that VEGA utilizes higher exploitation power, whilst the results of exploration and exploitation Gaps shows that SPEA2 is better at exploring and exploiting the region around dominant parents. The figures of search space and revisited solutions also reflect the aforementioned findings. Our future direction is to further apply the measures to evolutionary algorithms in different categories (e.g., hypervolume-based [1], parallel VEGA [8], MOEA/D [12], and OMOPSO [9]) and utilizes them as feedback to control the algorithms.

Notes

1. In [3] various diversity measures are considered as requisites and drives toward the balance between exploration and exploitation, and hence they are not considered as quantitative measures of exploration and exploitation.

References

[1] J. Bader and E. Zitzler. Hype: An algorithm for fast hypervolume-based many-objective optimization. *Evol. Comput.*, 19(1):45–76, 2011.

[2] M. Črepinšek, M. Mernik, and S.-H. Liu. Analysis of exploration and exploitation in evolutionary algorithms by ancestry trees. *International Journal of Innovative Computing and Applications*, 3(1):11–19, 2011.

[3] M. Črepinšek, S.-H. Liu, and M. Mernik. Exploration and exploitation in evolutionary algorithms: A survey. *ACM Comput. Surv.*, in press.

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

[5] J. Knowles and D. Corne. Towards Landscape Analyses to Inform the Design of Hybrid Local Search for the Multiobjective Quadratic Assignment Problem. In *Soft Computing Systems: Design Management, and Applications*, pages 271–279, 2002.

[6] E. Lutton and J.-D. Fekete. Visual Analytics and Experimental Analysis of Evolutionary Algorithms. Technical Report. INRIA, 2011 (`http://hal.archives-ouvertes.fr/docs/00/58/71/70/PDF/RR-7605.pdf`).

[7] Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs.* Springer-Verlag, 1996.

[8] K. Parsopoulos, D. K. Tasoulis, and M. N. Vrahatis. Multiobjective Optimization Using Parallel Vector Evaluated Particle Swarm Optimization. In *Proce. IASTED International Conference on Artificial Intelligence and Applications*, pages 823–828, 2004.

[9] M. Reyes and C. Coello. Improving PSO-based Multiobjective Optimization Using Crowding, Mutation and e-dominance, In *Evolutionary Multi-Criterion Optimization*, pages 505–519, 2005.

[10] J. D. Schaffer. Multiple Objective Optimization with Vector Evaluated Genetic Algorithms. In *Proc. 1st International Conference on Genetic Algorithms*, pages 93–100, 1985.

[11] V. K. Vassilev, T. C. Fogarty, and J. F. Miller. Smoothness, Ruggedness and Neutrality of Fitness Landscapes: from Theory to Application. In A. Ghosh and S. Tsutsui, editors. *Advances in Evolutionary Computing*, pages 3–44. Springer, 2003.

[12] Q. Zhang and H. Li. MOEA/D: A multi-objective evolutionary algorithm based on decomposition. *IEEE T. Evolut. Comput.*, 11(6):712–731, 2007.

[13] E. Zitzler and L. Thiele. Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach. *IEEE T. Evolut. Comput.*, 3(4):257–271, 1999.

[14] E. Zitzler, K. Deb, and L. Thiele. Comparison of Multiobjective Evolutionary Algorithms: Empirical Results. *Evol. Comput.*, 8(2):173–195, 2000.

[15] E. Zitzler, M. Laumanns, and L. Thiele. SPEA2: Improving the Strength Pareto Evolutionary Algorithm for Multiobjective Optimization. In *Evolutionary Methods for Design: Optimisation and Control*, pages 95–100, 2002.

MULTI-AGENT COLLABORATIVE SEARCH WITH TCHEBYCHEFF DECOMPOSITION AND MONOTONIC BASIN HOPPING STEPS

Federico Zuiani, Massimiliano Vasile
Department of Mechanical & Aerospace Engineering, University of Strathclyde, Glasgow, UK
{Federico.Zuiani; Massimiliano.Vasile}@strath.ac.uk

Abstract This paper presents a novel formulation of Multi Agent Collaborative Search for multiobjective optimization. A population of agents combines global exploration heuristics and moves to explore the neighborhood of each agent. In this novel formulation the selection process is based on the Tchebycheff decomposition of the multiobjective optimization problem into single objective optimization problems in combination with the use of the dominance index. The decomposition allows the implementation of Monotonic Basin Hopping steps that improve convergence on single funnel structures. The novel agent-based algorithm is tested on a standard benchmark and on a real space trajectory design problem. Its performance is compared against a number of state-of-the-art multi-objective optimization algorithms.

Keywords: Monotonic basin hopping, Multi-agent systems, Multi-objective optimization, Space trajectory design, Tchebycheff decomposition.

1. Introduction

Multi-Agent Collaborative Search (MACS) has been proposed as a framework for the implementation of hybrid, population-based approaches for multi-objective optimization [9, 11]. In particular, the search for Pareto optimal solutions is carried out by a population of agents implementing a combination of social and individualistic actions. An external archive is then used to reconstruct the optimal Pareto set.

The individualistic actions are devised to allow each agent to independently converge to the Pareto optimal set, thus creating its own partial representation of the Pareto front. Therefore, they can be regarded as

memetic mechanisms associated to a single individual. The effectiveness of the use of local moves was recently demonstrated by [3] who proposed innovative local search mechanisms based on mathematical programming.

Other examples of memetic algorithms for multiobjective optimization use local sampling or gradient-based methods, generally building a scalar function to be minimized or hybridazing an evolutionary algorithm with a Normal Boundary Intersection (NBI) technique [2, 6, 7]. The schedule with which the local search is run is critical and defines the efficiency of the algorithm.

In this paper Pareto dominance is used to rank and select the outcomes of each action in combination with the use of Tchebycheff decomposition to solve a number of single objective optimization problems in parallel. Furthermore, opposite to previous implementations of MACS, here all agents perform individualistic actions while social actions are performed only by selected sub-populations of agents.

Recent work by Zhang et al. [12] has demonstrated that Tchebycheff decomposition can be effectively used to solve difficult multi-objective optimization problems. In this paper, it will be demonstrated how MACS based on Tchebycheff decomposition can achieve very good results on a number of cases, improving over previous implementations and state-of-the-art Multi-Objective Optimisation (MOO) algorithms.

The new algorithm is here applied to a set of known standard test cases and a real space mission design problem [11]. In this latter case, it is shown that the introduction of Monotonic Basin Hopping (MBH) steps improves convergence.

2. Problem Formulation

The approach in this paper aims at finding the feasible set of solutions solving the following problem:

$$\min_{\mathbf{x}\in D} \mathbf{f}(\mathbf{x}), \tag{1}$$

where D is a hyperrectangle, defined as $\mathbf{f}$ is the vector function $\mathbf{f} : D \to \mathbb{R}^m, \mathbf{f}(\mathbf{x}) = [f_1(\mathbf{x}), f_2(\mathbf{x}), ..., f_m(\mathbf{x})]^T$. The optimality of a particular solution is defined through the concept of dominance: with reference to problem (1), a vector $\mathbf{y} \in D$ is dominated by a vector $\mathbf{x} \in D$ if $f_l(\mathbf{x}) \leq f_l(\mathbf{y})$ for all $l = 1, ..., m$ and there exists k so that $f_k(\mathbf{x}) \neq f_k(\mathbf{y})$. The relation $\mathbf{x} \prec \mathbf{y}$ states that $\mathbf{x}$ dominates $\mathbf{y}$. A decision vector in D that is not dominated by any other vector in D is said to be Pareto optimal. It is therefore possible to associate, to each solution in a set,

the scalar dominance index:

$$I_d(\mathbf{x}_i) = |\{i^* \mid i, i^* \in N_p \wedge \mathbf{x}_{i^*} \prec \mathbf{x}_i\}|, \tag{2}$$

where the symbol $|.|$ is used to denote the cardinality of a set and N_p is the set of the indices of all the solutions. The dominance index is one way to rank and select optimal moves generated by a search algorithm. Another way is to transform the vector function into a scalar. In Tchebycheff' approach to the solution of problem (1), a number of scalar optimization problems are solved in the form:

$$\min_{\mathbf{x}\in D} g(\mathbf{f}(\mathbf{x}), \lambda, \mathbf{z}) = \max_{l=1,...,m} \{\lambda_l | f_l(\mathbf{x}) - z_l|\}, \tag{3}$$

where $\mathbf{z} = \{z_1, ..., z_m\}$ is the vector of reference points $z_l = \min_{\mathbf{x}\in D} f_l(\mathbf{x})$ and λ_l is the l-th component of the weight vector λ. By solving a number of problems (3), with different weight vectors, one can obtain different Pareto optimal solutions. In the remainder of the paper the two techniques will be combined in a single framework.

3. MACS with Tchebycheff Decomposition

The key idea underneath multi-agent collaborative search is to combine local and global search in a coordinated way such that local convergence is improved while retaining global exploration [9]. This combination of local and global search is achieved by endowing a set of agents with a repertoire of actions producing either the sampling of the whole search space (social actions) or the exploration of a neighborhood of each agent (individualistic actions). A population P_0 of n_{pop} virtual agents, one for each solution vector $\mathbf{x}_i$, with $i = 1, ..., n_{pop}$, is deployed in the problem domain D. A fraction $n_{social} = round(\rho_{pop} n_{pop})$ of agents will perform *social* actions. The population P_h at iteration $h = 0$ is initialized using a Latin Hypercube distribution. Each agent then evaluates the associated objective vector $\mathbf{f}_i = \mathbf{f}(\mathbf{x}_i)$ and all non-dominated agents are cloned and inserted in the global archive A_g (lines 4-5 of Algorithm 1). The q-th element of the archive is the vector $\mathbf{a}_q = [\xi_q \;\; \phi_q]^T$ where ξ_q is a vector in the parameter space and ϕ_q is a vector in the criteria space. Each agent is associated to a neighborhood D_{ρ_i} with size ρ_i. The size ρ_i is initially set to 1, i.e. representing the entire domain D (line 6 of Algorithm 1).

A set of $n_\lambda = 100m$, m-dimensional unit vectors λ_k is initialized such that the first m vectors are mutually orthogonal. The remaining $n_\lambda - m$ have random components instead (line 7 of Algorithm 1). In two dimensions the vectors are initialized with a uniform sampling on a unit circle and in three dimensions with a uniform sampling on a

unit sphere, while in n-dimensions with a Latin Hypercube sampling plus normalization, such that the length of each vector is 1. Note that, although the latter strategy does not guarantee a uniform distribution in n dimension, it will still provide an adequately diversified set of weight vectors. For each vector λ_k, the value of an associated utility function U_k is set to 1 (line 8 of Algorithm 1). The utility function is the one defined in [12] and its value is updated every $f_{sel} = 30$ as described in Section 3.3 (lines 27–29 of Algorithm 1).

Each λ_k represents a subproblem in Eq. (3), i.e. it is used to compute the scalar function g_k. A total of n_{social} λ vectors are inserted in the index set I_a (line 9 of Algorithm 1). The first m indexes in I_a correspond to the m orthogonal λ vectors, the other $n_{social} - m$ are initially chosen randomly. Each λ_k for $k = 1, ..., n_\lambda$ is associated to the element in A_g that minimizes g_k such that:

$$\underline{\phi_k} = \underset{\phi_q}{\operatorname{argmin}}\, g(\phi_q, \lambda_k, \mathbf{z}), \tag{4}$$

where $\mathbf{z}$ is the vector containing the minimum values of each of the objective functions. Then, for each λ_k, with $k \in I_a$ and associated vector $\underline{\phi_k}$, a *social* agent is selected from the current population P_h such that it minimizes the Euclidean distance, in criteria space, from $\underline{\phi_k}$. The indexes of all the selected social agents are inserted in the index set I_λ (lines 11-14 of Algorithm 1). The indexes in I_a and I_λ are updated every f_{sel} iterations (lines 27-29 of Algorithm 1).

At the h-th iteration, the population P_h is evolved through two sets of heuristics: first, every agent $\mathbf{x}_i$ performs a set of *individualistic actions* which aims at exploring a neighborhood D_{ρ_i} of $\mathbf{x}_i$, as described in Section 3.1 (line 17 of Algorithm 1). All the samples collected during the execution of individualistic actions are stored in the local archive A_l. The elements of A_l and the outcome of the social actions are then inserted in the global archive A_g if they are not dominated by any element of A_g (line 19 of Algorithm 1).

Then, a sub-population I_λ of n_{social} selected social agents performs a set of *social actions* (line 20 of Algorithm 1). Social actions aim at sharing information among agents and are described in Section 3.2. Every 20 iterations the following actions are also performed: first, if two or more agents have come too close, a *local restart* is implemented (line 22 of Algorithm 1); secondly, if the size of the global archive A_g has grown larger than $n_{A,max}$, some of them are discarded to reduce it below that limit (lines 23-25 of Algorithm 1). When two or more agents come too close, the *local restart* preserves one agent among the non-dominated agents and among those that belong to the *social* subset while the others are randomly generated in the domain D.

Algorithm 1 MACS2

1: Set $n_{feval,max}$, n_{pop}, n_{social}, F, tol_{conv}, $tol_{locconv}$, $n_{A,out}$
2: Set $n_\lambda = 100m$, $n_{A,max} = round(1.5 \max([n_\lambda, n_{A,out}]))$ $f_{sel} = 30$
3: Set $n_{feval} = 0$
4: Initialize population P_h, $h = 0$
5: Insert the non-dominated elements of P_0 in the global archive A_g
6: $\rho_i = 1$, $\forall i \in \{1, ..., n_{pop}\}$
7: Initialize λ_k for $k \in \{1, ..., n_\lambda\}$ such that $\|\lambda_k\| = 1$
8: Initialize utility function vector $U_k = 1, \forall k \in \{1, ..., n_\lambda\}$
9: Select the n_{social} active subproblems λ_l, $l \in I_a$
10: Initialize $\delta_l = \max_q \phi_{q,l} - \min_q \phi_{q,l}$, $z_l = \min_q \phi_{q,l}$, $q \in \{1, ..., |A_g|\}$, $l = 1, ..., m$
11: **for all** λ_l, $l \in I_a$ **do**
12: Select the $[\mathbf{x}_q \mathbf{f}_q] \in P_h$ which minimises $g(\mathbf{f}_q, \lambda_l, \mathbf{z})$
13: and save its index in the list of the social agents I_λ
14: **end for**
15: **while** $n_{feval} < n_{feval,max}$ **do**
16: $h = h + 1$
17: $[P_h, n_{feval}, A_l, \rho] = explore(P_{h-1}, n_{feval}, n, \rho, \mathbf{b}^l, \mathbf{b}^u, \mathbf{f}, \lambda, I_\lambda, I_a)$
18: If necessary, update the vector of the best objectives $\mathbf{z}$, with A_l
19: Update archive A_g with non dominated elements of A_l
20: $[P_h, A_g, , n_{feval}] = com(P_h, A_g, \mathbf{b}^l, \mathbf{b}^u, n_{feval}, n, T, F, p_{AvsP}, \mathbf{f}, \lambda, I_\lambda, I_a)$
21: **if** (mod $(h, 20) = 0$) **then**
22: If necessary, perform a *local restart*
23: **if** $|A_g| > n_{A,max}$ **then**
24: Perform *archiving resizing* so that $|A_g| = n_{A,max}$
25: **end if**
26: **end if**
27: **if** (mod $(h, f_{sel}) = 0$) **then**
28: $[I_a, I_\lambda, \mathbf{U}, \lambda_f] = select(\mathbf{U}, \lambda, \lambda_f, P_k, A_g, \mathbf{z}, m, n_{social}, n_\lambda)$
29: **end if**
30: Update n_{feval}
31: **end while**
32: Perform *archiving resizing* before output so that $|A_g| = n_{A,out}$

The value $n_{A,max}$ was selected to be the largest number between $1.5n_\lambda$ and $1.5n_{A,out}$, where $n_{A,out}$ is the desired number of Pareto optimal elements in A_g at the last iteration. At the end of each iteration the algorithm checks if the maximum number of function evaluations $n_{feval,max}$, defined by the user, has been reached and if so, the algorithm terminates. At termination, the archive A_g is resized to $n_{A,out}$ if its cardinality is bigger than $n_{A,out}$ (line 32 of Algorithm 1).

3.1 Individualistic Actions

Individualistic actions (line 17 of Algorithm 1) perform an independent exploration of the neighborhood D_{ρ_i} of each agent. As in the origi-

nal version of MACS [9] the neighborhood is progressively resized so that the exploration is over the entire D when the size ρ_i is equal to 1 and becomes progressively more and more local as the neighborhood shrinks down. In this new implementation of MACS each agent performs only a simple sampling along the coordinates similar to the one described in [8]. The neighborhood D_{ρ_i} is a hypercube centered in $\mathbf{x}_i$ with size defined by ρ_i such that each edge of the hypercube has length $\rho_i(\mathbf{b}^u - \mathbf{b}^l)$.

The search is performed along a single component of $\mathbf{x}_i$ at a time, in a random order: given an agent $\mathbf{x}_i$, a sample $\mathbf{y}^+$ is taken within D_{ρ_i} along the j-th coordinate with random step size $r \in \mathcal{U}(-1, 1)$, where $\mathcal{U}(-1, 1)$ is a uniform distribution over the closed interval [-1 1], leaving the other components unchanged. If $\mathbf{y}^+$ dominates $\mathbf{x}_i$, the search is interrupted and $\mathbf{y}^+$ replaces $\mathbf{x}_i$, otherwise another sample $\mathbf{y}^-$ is taken in the opposite direction with step size $0.5r$. Again, if $\mathbf{y}^-$ dominates $\mathbf{x}_i$ the search is interrupted and $\mathbf{y}^-$ replaces $\mathbf{x}_i$. If the child $\mathbf{y}_i$ is not dominating the father and is not dominated by the father and the index of $\mathbf{x}_i$ belongs to I_λ, then the child replaces the father if the child improves the value of the subproblem associated to its father and the exploration terminates. This is a key innovation that exploits Tchebycheff decomposition and allows the agents to perform moves that improve one objective function at the time. If more than one quarter of the components of $\mathbf{x}_i$ have been examined, the search terminates even if all the children are dominated.

If all children are dominated by their parent, the size of the neighborhood ρ_i is reduced according to the following mechanism: if the size is larger than 0.1, it is simply reduced by 10%; if it is lower than 0.1 instead, the size is halved. Finally, if ρ_i is smaller than a tolerance tol_{conv}, it is reset to 1. This dual rate of contraction of the step size favors global exploration of the search space.

All the children generated by each agent $\mathbf{x}_i$ during the exploration form the local archive $A_{l,i}$. The elements of $A_{l,i}$ are inserted in the global archive A_g if they are not dominated by any element in A_g.

3.2 Social Actions

Social actions (line 20 of Algorithm 1) are performed by each agent whose index is in the set I_λ. Social actions are meant to improve the subproblem defined by the weight vectors λ_k in I_a and associated to the agents $\mathbf{x}_i$ in I_λ. This is done by exploiting the information carried by either the other agents in the population P_h or the elements in the archive A_g. Social actions implement the Differential Evolution (DE) heuristic:

$$\mathbf{y}_i = \mathbf{x}_i + K[(\mathbf{s}_1 - \mathbf{x}_i) + F(\mathbf{s}_2 - \mathbf{s}_3)], \tag{5}$$

where F is the differential weight and the vectors $\mathbf{s}_l$, with $l = 1, 2, 3$, are randomly taken from the local social network I_T of each social agent $\mathbf{x}_i$. K is sampled uniformly between 0 and 1. The local social network is formed by either the T agents closest to $\mathbf{x}_i$ or the T elements of A_g closest to $\mathbf{x}_i$. The probability of choosing the archive vs. the population is directly proportional to p_{AvsP}. The offspring $\mathbf{y}_i$ replaces $\mathbf{x}_i$ if it improves the subproblem associated to $\mathbf{x}_i$ otherwise $\mathbf{y}_i$ is added to the archive A_g if it is not dominated by any of its elements. Social actions, dramatically improve the convergence speed once a promising basin of attraction has been identified. On the other hand, in some cases social actions lead to a collapse of the subpopulation of social agents in one or more single points. This is in line with the convergence behavior of DE dynamics presented in [10]. This drawback is partially mitigated by the implementation of *local restart* actions and by the remaining agents which perform only *individualistic actions*.

3.3 Subproblem Selection

Every f_{sel} iterations the active subproblems in I_a and the associated agents in I_λ performing *social* actions are updated (line 28 of Algorithm 1). The improvement γ between $\underline{\phi}_k$ (i.e. the value of ϕ that minimizes g_k at current iteration) and $\underline{\phi}_{old,k}$ (the the value of ϕ that minimizes g_k, f_{sel} iterations before) is calculated. Then, the utility function U_k associated to λ_k is updated according to the rule described in [12]. Once a value U_k is associated to each λ_k, n_{social} new subproblems and associated λ vectors are selected. The first m λ vectors are always the orthogonal ones. The remaining $n_{social} - m$ are selected by taking $t_{size} = round(n_\lambda/60)$ random indexes and then choosing the one among the t_{size} with the largest value of U_k. This is repeated till I_a is full. Note that t_{size} cannot exceed the size of I_{tmp} if the number of social agents n_{social} is small compared to n_λ. Finally, the agent $\mathbf{x}_i$, that minimizes the scalar objective function in Eq. (3) is associated to each λ_k with index in I_a, and its index is included in the new subset I_λ.

4. Experimental Results

The new implementation of MACS, here called MACS2, is applied to the solution of the UF problems proposed by Zhang et al. [13] for the CEC'09 competition following the same rules and using the same performance metric: the IGD. MACS2 is then tested on a difficult space trajectory design problem described in [11]: the multi-gravity assist transfer to Saturn for the Cassini mission. For the Cassini case the algorithm is run for 600000 evaluations as it was demonstrated to have multiple

nested local minima with a funnel structure [10]. The metrics which will be used to evaluate the performance of the algorithms on the Cassini case are the success rate on the convergence M_{conv} and spreading M_{spr} as introduced in [11]. The success rates p_{conv} and p_{spr} are computed over 200 independent runs. They count the number of times M_{conv} and M_{spr} are below their respective thresholds τ_{conv} and τ_{conv}. According to the theory developed in [10], 200 runs provide a 5% error interval with a 95% confidence level. For Cassini $\tau_{conv} = 7.5e-3$ and $\tau_{spr} = 5e-2$.

The settings reported in Table 1 are recommended and were obtained after an extensive experimental analysis of the sensitivity of the performance of the algorithm to each parameter.

Table 1. Settings for MACS2.

n_{pop}	$\rho_{pop} = n_{social}/n_{pop}$	F	T	p_{AvsP}	Tol_{conv}
150(20)	0.2	0.9	30	0.9	10^{-6}

With these settings, the performance of MACS2 was compared, on the UF test suite in Table 2, with that of MACS [11], MOEAD [12], MTS [8] and DMOEADD [5]. The last three are the best performing algorithms in the CEC'09 competition [13].

Table 2. Performance comparison on UF test cases: Average IGD (variance within parenthesis).

	MACS2	MACS	MOEAD	MTS	DMOEADD
UF1	**4.26e-3 (8.49e-9)**	1.15e-1 (1.66e-3)	4.35e-3	6.46e-3	1.04e-2
UF2	**4.31e-3 (8.96e-9)**	5.43e-2 (4.19e-4)	6.79e-3	6.15e-3	6.79e-3
UF3	**1.56e-2 (1.02e-5)**	6.56e-2 (1.42e-3)	7.42e-3	5.31e-2	3.34e-2
UF4	3.32e-2 (7.03e-7)	3.36e-2 (1.66e-5)	6.39e-2	**2.36e-2**	4.27e-2
UF5	8.60e-2 (1.68e-4)	6.44e-2 (1.17e-3)	1.81e-1	**1.49e-2**	3.15e-1
UF6	**2.46e-2 (5.31e-5)**	2.40e-1 (1.43e-2)	1.76e-1	5.91e-2	6.67e-2
UF7	**4.16e-3 (2.49e-8)**	1.69e-1 (1.22e-2)	4.44e-3	4.08e-2	1.03e-2
UF8	**4.58e-2 (1.93e-6)**	2.35e-1 (1.77e-3)	5.84e-2	1.13e-1	6.84e-2
UF9	**2.76e-2 (1.82e-6)**	2.68e-1 (1.71e-2)	7.90e-2	1.14e-1	4.90e-2
UF10	1.56e-1 (5.94e-5)	1.25 (4.28e-1)	4.74e-1	**1.53e-1**	3.22e-1

As shown in Table 2, MACS2 outperforms the other algorithms on UF1, 2, 3, 6, 7, 8 and 9, while it ranks second on the remaining UF4 and 10 and finally third on UF5. Table 3 shows a comparison of the

Table 3. Comparison of MACS, MACS2 and MOEAD on the Cassini test case: success rates on Convergence and Spreading metrics.

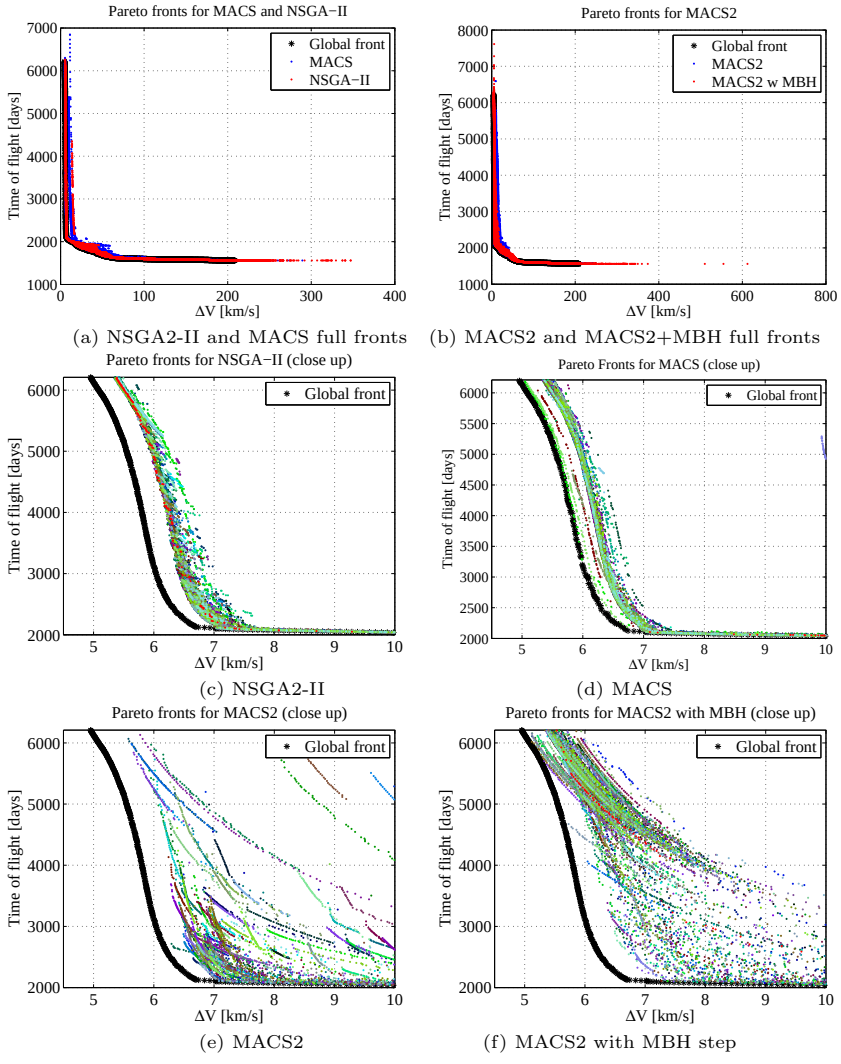
	MACS	MACS2	MOEAD	MTS	NSGA-II
p_{conv}	0.87	0.79	0.51	0.05	0.90
p_{spr}	0.49	0.29	0.01	0.32	0.26

performance of MACS2 on the Cassini case, against MACS, MOEAD, MTS and NSGA-II [1]. Only MACS, MACS2 and NSGA-II reach a high convergence rate, but for the last two, their spreading is still lower than for MACS. After inspection of each of the 200 Pareto fronts one can see that such a low spreading implies that the algorithm did not converge to the global Pareto front. Figure 1.a shows the global Pareto front together with the collection of all the fronts found by NSGA-II and MACS over 200 runs. Likewise, Fig. 1.b shows the global Pareto front together with the collection of all the fronts found by MACS2 and MACS2 with MBH step over 200 runs. Figure 1.c, instead, illustrates closeup of the distribution of the fronts of NSGA-II over the 200 runs. MACS achieves the best known value for objective function Δv, see Fig. 1.d. Both NSGA-II and MACS2, instead, tend to fall in the basin of attraction of the second best value for objective function Δv [10]. The distribution of the fronts of MACS2 can be seen in Fig. 1.e. Note that although NSGA-II in Fig. 1.f appears to cover quite well the vertical side of the front, in fact over 200 runs only a few succeed to capture the vertical part. That justifies the poor spreading rate.

The performance of MOEAD and MTS on Cassini is rather poor, with the former attaining only 50% convergence but with almost zero p_{spr}; conversely, only one third of the latter's runs are below the spreading threshold and almost none meets the convergence criterion.

5. MACS2 with Monotonic Basin Hopping

The use of Tchebycheff decomposition leads to the possibility to introduce Monotonic Basin Hopping steps [4] in the action set of each agent. MBH steps are introduced as a sampling technique in the individualistic actions but only for the individuals which are solving the m pure single objective subproblems. In this variant of the individualistic actions, at each iteration, first the standard search along the coordinates is performed (as described in Section 3.1); then a sample is taken in the domain D and MatLab®'s fmincon is used to solve the m-th scalar prob-



(a) NSGA2-II and MACS full fronts (b) MACS2 and MACS2+MBH full fronts

(c) NSGA2-II (d) MACS

(e) MACS2 (f) MACS2 with MBH step

Figure 1. NSGA-II, MACS, MACS2 on the Cassini case:a) NSGA-II and MACS b) MACS2 and MACS2 with MBH step c) NSGA-II closeup d) MACS closeup e) MACS2 closeup and f) MACS2 with MBH step closeup.

lem from this starting point, attaining convergence to a local minimum. If this latter point is better than the current individual, it will replace it. This variant has been tested on the Cassini introducing the MBH step only in the action set of the two agents that are solving the extremal single objective problems. The Cassini case resulted to be quite challenging for MACS2 it is known to have an interesting funnel structure

[10]. The MBH steps yield a marked improvement in both convergence and spreading compared to the standard version of MACS2, with a p_{conv} increasing to 85% and p_{spr} to 99%. This improvement is related essentially to a much closer convergence in the part of the front corresponding to the minimum ΔV solution. Figure 1.f shows the distribution of the Pareto fronts of MACS2 with MBH steps over the 200 runs.

The MBH steps effectively contribute to increase the probability of identifying the single objective minimum, leading to a improved convergence also in its neighborhood. However, the current implementation of MBH steps is less effective at improving other parts of the front and more work is required in this direction.

6. Conclusions

This paper has presented a version of Multi-Agent Collaborative Search based on Tchebycheff decomposition. The new version, denominated MACS2, demonstrated remarkable performance on a known difficult benchmark outperforming known algorithms. On the Cassini real case application MACS2 initially fell back behind its predecessor. However, the introduction of MBH steps in the Tchebycheff decomposition framework provided a net improvement of the performance. As future developments, it is envisioned to perform further testing of MACS2 with MBH on different test cases. Different performance metrics, available in the literature, will also be considered and comparisons will be made with other state-of-the-art MOO algorithms.

References

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

[2] I. Hisao and Y. Tadashi. Hybrid evolutionary multi-objective optimization algorithms. In A. Abraham, J. Ruiz-Del-Solar, M. Köppen, editors. *Soft computing systems: design, management and applications*, pages 163–172, IOS Press, 2002.

[3] A. Lara, G. Sanchez, C. A. Coello Coello, and O. Schütze. HCS: A new local search strategy for memetic multiobjective evolutionary algorithms. *IEEE T. Evolut. Comput.*, 14(1):112–132, 2010.

[4] R. H. Leary. Global optimization on funneling landscapes. *J. Global Optim.*, 18(4):367–383, 2000.

[5] M. Liu, X. Zou, Y. Chen, and Z. Wu. Performance assessment of DMOEA-DD with CEC 2009 MOEA competition test instances. In *Proc. IEEE Congress on Evolutionary Computation*, pages 2913–2918, 2009.

[6] E. Rigoni and S. Poles. NBI and MOGA-II, two complementary algorithms for multi-objective optimizations. In J. Branke, K. Deb, K. Miettinen, and R. E.

Steuer, editors. *Practical Approaches to Multi-Objective Optimization, Dagstuhl Seminar Proceedings*, pages 1862–4405, 2005.

[7] K. Sindhya, A. Sinha, K. Deb, and K. Miettinen. Local search based evolutionary multi-objective optimization algorithm for constrained and unconstrained problems. In *Proc. IEEE Congress on Evolutionary Computation*, pages 2919–2926, 2009.

[8] L. Y. Tseng and C. Chen. Multiple trajectory search for unconstrained/constrained multi-objective optimization. In *Proc. IEEE Congress on Evolutionary Computation*, pages 1951–1958, 2009.

[9] M. Vasile. Robust mission design through evidence theory and multiagent collaborative search. *Ann. NY Acad. Sc.*, 1065:152–173, 2005.

[10] M. Vasile, E. Minisci, and M. Locatelli. An inflationary differential evolution algorithm for space trajectory optimization. *IEEE T. Evolut. Comput.*, 15(2):267–281, 2011.

[11] M. Vasile and F. Zuiani. Multi-agent collaborative search: an agent-based memetic multi-objective optimization algorithm applied to space trajectory design. *P. I. Mech. Eng. G-J. Aer.*, 225(11):1211–1227, 2011.

[12] Q. Zhang and H. Li. MOEA/D: A multiobjective evolutionary algorithm based on decomposition. *IEEE T. Evolut. Comput.*, 6(11):712–731, 2007.

[13] Q. Zhang and P. N. Suganthan. Final report on CEC09 MOEA competition. In *Proc. IEEE Congress on Evolutionary Computation*, 2009.

MIN-DEGREE CONSTRAINED MST PROBLEM: AN EVOLUTIONARY APPROACH

Ana Maria de Almeida, Rui Salgueiro
CISUC - Center for Informatics and Systems, University of Coimbra, Portugal
{amca; rps}@mat.uc.pt

Orlando Oliveira
Center for Computational Physics, University of Coimbra, Portugal
orlando@teor.fis.uc.pt

Abstract This work describes a genetic approach for the combinatorial optimization problem named the *Min-Degree Constrained Minimum Spanning Tree* (m*d*-MST). Given a connected, undirected and weighted graph, the m*d*-MST aims to find a minimal spanning tree that obeys an additional lower bound (d) on the degree of each node of the tree. This problem is NP-hard and thus a suitable target for heuristics. We decided to approach the problem via evolutionary algorithms. Two novel genetic algorithms are presented differing in their representations of candidate trees and the operations that they apply to these chromosomes. While one uses a set of node leaves randomly chosen as chromosome and enforces the construction of a complete spanning tree, the other uses a representation of the set of arcs instead. Both genetic versions perform very favorably when compared with the best results known so far, both in terms of running time as well as in better quality solutions.

Keywords: Combinatorial optimization, Degree constrained spanning tree, Evolutionary approach, Genetic algorithms, Heuristics.

1. Introduction

Consider a classical version of a connected weighted undirected graph, $G = (V, E)$, where $V = \{1, \ldots, n\}$ is the set of nodes and $E = \{e = \{i, j\} : i, j \in V\}$ the set of m edges. Assume that all edge costs are positive: $c_{ij} > 0$ for any edge connecting nodes i and j. As usual, the (total) cost of the graph G is given by $C(G) = \sum_{e \in V} c_e$, $e \in E$. $Sub(G)$

denotes the set of all subgraphs of G. Note that, in the following, only connected graphs will be considered.

A common optimization task over graph models is that of finding a connected acyclic subgraph that covers all the nodes of the graph, that is, a spanning tree. Let T denote a spanning tree for G, i.e., $T = (V, E_T)$, where $E_T \subseteq E$ and is a maximal set of edges of G that contains no cycles. The degree of node i represents the number of edges having i is an end point. Denote by $deg_T(i)$ the degree of the node $i \in V$. A tree with minimal cost, that is, $T^* = min\{C(T) : T \in Sub(G) \wedge \text{T is a tree}\}$ is known as a *minimal spanning tree* (MST).

Formally, the general *Min-Degree Constrained Minimum Spanning Tree* (md-MST) problem is defined as: given a positive integer $d \in \mathbb{N}$, find the spanning tree T for G with *minimal total edge cost* such that each tree node either has degree, at least, d or is a leaf node (node with degree one).

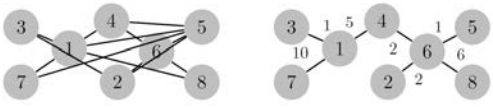


Figure 1. Example: Graph G_1 (left) and a spanning tree T for G_1 (right). Note that T does not obey the min-degree restriction of $deg_T(i) \geq 4$.

This problem was recently introduced in [1], where the authors proved it to be a NP-hard problem for $\lfloor n/2 \rfloor > d > 3$. In [2] the proof was generalized to include the degree 3 also as a hardness lower bound.

The authors of [1] introduced commodity flow formulations for the md-MST and applied Linear Programming relaxations performing extensive numerical tests. However, the results seem to reflect the hardness of the problem. Trying to overcome some of the computational difficulties found, Martins and Souza [9] designed new algorithmic approaches, based on Variable Neighborhood Search (VNS) meta-heuristics transformed for the md-MST. An adaptation of a greedy heuristic based on the Kruskal [7] algorithm for determining minimal cost spanning trees was also presented.

The next reference to the md-MST can be found in [3] where the authors discuss a new set of degree-enforcing constraints and use Miller–Tucker–Zemlin sub-tour elimination constraints as an alternative to single or multi-commodity flow constraints for the tree-defining part of the md-MST formulation. To test their own algorithm's performance, the authors performed several computational experiments. They did not, however, compare their results. On the other hand, the authors of [8] compared their new algorithmic approaches, a Branch-and-cut

method with a Branch-and-bound algorithm based on the Improved Miller-Tucker-Zemlin formulation (IMTZDEF2) introduced in [3], using the test bed from [1]. On the whole, the kind of issues arising from the three independent frameworks are basically the same: large computational running times; the larger the instances the more inconsistent are the results.

The present work proposes a new evolutionary approach to the optimal solution of the m*d*-MST problem, by exploring new codings for the candidate spanning trees and operators. Two different genetic algorithms are presented. The first one uses a set of randomly chosen leaves as chromosome and enforces the construction of a complete spanning tree using Kruskal's algorithm. The second genetic version instead of the set of nodes uses a representation of the set of arcs.

The remaining of this paper is organized as follows. Next section introduces the novel approach to the m*d*-MST problem, which is based on the use of genetic algorithms. In Section 3 computational experiments are reported and we discuss the relative efficiency of the various heuristic approaches. Finally, Section 4 presents the conclusions and a discussion of possible directions for future work.

2. An Evolutionary Approach for the m*d*-MST

Contrasting with the methods found in the related literature, our approach relies on genetic algorithms as meta-heuristics that can be used to investigate a large number of different types of problems. As usual, we take the concept of genetic algorithm as a method that intends to mimic a species evolution process: starting with an *initial population* of candidate solutions randomly generated, where each individual is represented by a *chromosome* (its genotype: usually an array of bits or numbers), each step (or iteration) of the algorithm consists on the *evolution* of the candidates population guided by a *fitness function* [4, 10]. The genetic algorithm is eventually stopped with a usually good quality solution to the problem at hand.

This kind of heuristic approaches are used for the combinatorial optimization of NP-hard problems [11], where the fitness function is usually referred to as the *cost function*. The evaluation of the cost function is done on the phenotypes, i.e., on the individual chromosomes of the population, which is also known as the *search space*. Genetic algorithms are generally very good performing meta-heuristics, both in terms of solution quality and computational resources. The key issues for defining of a successful genetic algorithm are: a) a good choice for the chromosome representation; b) the definition of an appropriate fitness function;

and c) the tuning of the parameters of the algorithm like the size of the population, the number of generations and the genetic operators.

2.1 Chromosome Representations for Trees

Genetic version *gen1*: using the graph's leaf set. Within this context, the representation of a tree as a chromosome consists of using a set of leaves randomly chosen. To avoid the case when two leaves that belong to the same tree are join into the same chromosome (thus forming a cycle) we used a modified version of Kruskal's algorith enforcing the "construction" of acyclic complete graphs (spanning trees). However, when non–complete graphs are used this may preclude the generation of a (complete) spanning tree. In order to tackle this issue, the algorithm goes through the set of arcs once more but this time not excluding any edge. With a complete graph this last phase is never executed because after the first round all the leaf nodes are connected to central nodes.

The major drawback of this representation is the fact that it cannot guarantee that all existing trees can be represented. For instance, consider a complete graph with eight nodes of which the first six are leaves (see Fig. 2).

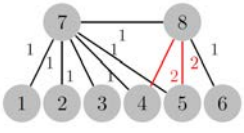


Figure 2. Example of a simple graph for which *gen1* will generate a non-admissible tree solution.

Kruskal's method is a greedy strategy that always chooses the arcs with lower cost to connect new leaves to already existing internal nodes. For the example graph in Fig. 2, five of the leaves will always be connected to node number 7 while only one edge will be connected to internal node number 8, which will have degree 2. Therefore, for a m3–MST problem, the resulting tree would be unfeasible.

Genetic version *gen2*: using the graph's edge set. Another possibility is that of representing the chromosome using the set of arcs in the tree. This version was implemented using an array of bits.

Note that, when generating a random set of the graph's edges, the probability of hitting a spanning tree is very low. In fact, for any complete graph with n nodes and $m = n(n-1)/2$ edges, there are $2^m = 2^{n(n-1)/2}$ possible different sets of edges. But only $n^{(n-2)}$ of these represent spanning trees.

We once again we resorted to adapt Kruskal's algorithm in a very similar fashion to the two step algorithm used in *gen1*: after a first round of a randomly generated edges set, if an admissible tree is not obtained, the second phase is applied repeating Kruskal's algorithm but, this time, considering all the arcs. Unlike the previous version, *gen2* usually needs to use the second phase to obtain a spanning tree, even when complete graphs are involved.

2.2 Parameters and Operators for the New Genetic Algorithms

A classical result states that the behavior of any genetic algorithm is affected by the various parameters associated with the genetic operators. For both genetic variants the same basic evolutionary strategy was used. As usual, an initial population several times the size of the evolutionary population is randomly generated. The evolution then starts by choosing the best fitted individuals of the initial population and, in each successive generation, only the best half of the solutions are kept through a fitness-based procedure.

Operators in use. A variant of an uniform crossover was used. With 50% of probability each bit of the chromosome is inherited from one of the parents. Within *gen1*, after the crossover, each bit can yet be flipped accordingly to a probability parameter control in the program. Finally, the bottom half of the population is replaced by the sons of the top half of the population thus created.

The version *gen2* is analogously implemented but now we have reproduction and mutations done in alternate generations. In fact, for this particular version, this alternation produced better results than the usual method.

Strategic parameters. Among the various parameters which define a genetic algorithm those having major impact on the performance are the size of the population (including both the size of the original and the evolutionary population), the number of generations and the percentage of mutations.

Since the md-MST is a new problem whose optima are unknown and no real benchmarks are available, it is hard to decide on the more adequate values for any of these parameters. Within this work, we will entirely rely on the empirical tests that were performed.

The first experiments intended to find what percentage of mutations should be used on each iteration. After testing several instances of the problem and different values of percentages, it became obvious that the

value of 3% was the one that obtained the best performance. This value was in accordance to the ones referred to in the related literature and it was the one we chose to apply.

Both the number of generations and the population size affect the number of candidate solutions and therefore the amount of calculations executed. Thus, these parameters were not only studied apart but also their interconnection was analyzed. Nevertheless, the major drawback into identifying possible values (or ranges) for these parameters is that the algorithm in itself is not deterministic: the results from each run are dependent on the pseudo-random generator that creates the initial population and controls the mutations. It is not enough to execute the algorithm once to determine whether the change of a parameter is or is not beneficial but it is necessary to perform several runs (more than 30, really) and use statistical criteria to infer if the difference in performance is of real significance.

2.3 m*d*-MST Fitness Evaluation

First of all, the genetic algorithm needs to be forced to evolve into feasible trees through an adequate fitness function. To evaluate the candidate solutions, we defined a linear combination of the total cost of the tree (see Section 1) and a measure of the tree's inadmissibility.

To define this measure, for each unfeasible degree node i in the tree, $1 < deg_T(i) < d$, we add the difference between the node's degree and the desired minimum d. Thus for each candidate tree T we combine the classical cost of the tree and the measure of inadmissibility to form a fitness function. We propose a convex combination by using a parameter $0 \leq wt < 1$:

$$wt \sum_{e \in E_T} c_e \; + \; (1 - wt) \sum_{i \,:\, 1 < deg_T(i) < d} min(d - deg_T(i), deg_T(i) - 1) \; .$$

Intensive computational tests have shown that, without this procedure, the genetic algorithm seldom finds admissible trees. Thus, the parameter wt starts at 0.9 and during the execution of the algorithm it is gradually reduced, thus forcing infeasible solutions to be refused.

3. Computational Tests

For performance evaluation, the results of genetic versions were compared with the results presented in [9]. All the experiments were performed using exactly the same test bed instances. In particular, the three classes of instances used were CRD, SYM and ALM. These are classical instances for testing algorithmic performance for the degree-constrained problem [6, 13]. However, hereinafter only the results for

the ALM class will be considered, since these are the more challenging ones. In fact, the results obtained by the methods in [9] showed a relatively big gap to the theoretical lower bound. The ALM class presents the larger dimension problems of the three classes with the instances' number of nodes ranging from 100 to 500 nodes. The nodes are evenly distributed points in a grid of size 480 x 640 and the weights of the edges are the truncated Euclidean distances between the points. Depending on the number of nodes of the instances, values for d ranging from 3 to 20 were used. All the graphs used are complete, thus, $m = n^2 - n$.

Table 1. Comparison of best results for the ALM class in terms of tree solutions cost functions.

Prob.	n	d	LB	BUB	$gen1$	$gen2$	$\frac{BUB}{LB}$	$\frac{gen1}{LB}$	$\frac{gen2}{LB}$
ALM-1	100	5	4617	**5439**	5650	5590	**1.18**	1.22	1.21
ALM-2			4513	5207	5333	**5178**	**1.15**	1.18	**1.15**
ALM-3			4883	**5456**	5623	5539	**1.12**	1.15	1.13
ALM-1		10	6883	**7180**	7540	7460	**1.04**	1.10	1.08
ALM-2			6551	**6915**	7191	7117	**1.06**	1.10	1.09
ALM-3			6967	**7509**	7713	7788	**1.08**	1.11	1.12
ALM-1	200	5	6018	7467	7461	**7433**	**1.24**	**1.24**	**1.24**
ALM-2			6188	**7680**	8017	7944	**1.24**	1.30	1.28
ALM-3			6463	8217	8116	**7996**	1.27	1.26	**1.24**
ALM-1		10	8862	10391	10283	**10252**	1.17	**1.16**	**1.16**
ALM-2			9167	**10238**	10913	11061	**1.12**	1.19	1.21
ALM-3			9150	**10533**	10995	10561	**1.15**	1.20	**1.15**
ALM-1	300	5	7599	9871	9538	**9520**	1.30	1.26	**1.25**
ALM-2			7474	10532	**9276**	9367	1.41	**1.24**	1.25
ALM-3			7712	10887	10031	**10016**	1.41	**1.30**	**1.30**
ALM-1		10	11177	13899	14039	**13445**	1.24	1.26	**1.20**
ALM-2			10712	13210	13094	**12704**	1.23	1.22	**1.19**
ALM-3			11245	13792	14016	**13466**	1.23	1.25	**1.20**
ALM-1	400	5	8369	12487	**10811**	11422	1.49	**1.29**	1.36
ALM-2			8051	13877	**11090**	11600	1.72	**1.38**	1.44
ALM-3			7914	12379	**11098**	11117	1.56	**1.40**	**1.40**
ALM-1		10	12457	17309	**15788**	16262	1.39	**1.27**	1.31
ALM-2			11841	16595	**15318**	15772	1.40	**1.29**	1.33
ALM-3			11731	16439	15238	**15214**	1.40	**1.30**	**1.30**
ALM-1		20	18430	**21339**	**21340**	21379	**1.16**	**1.16**	**1.16**
ALM-2			17868	21299	**20511**	21484	1.19	**1.15**	1.20
ALM-3			18023	**22049**	22743	22932	**1.22**	1.26	1.27
ALM-1	500	5	8819	14626	**12334**	12762	1.66	**1.40**	1.45
ALM-2			8912	14039	**12405**	13666	1.58	**1.39**	1.53
ALM-3			8872	13521	**11730**	13009	1.52	**1.32**	1.47
ALM-1		10	12855	19342	**17415**	17480	1.50	**1.35**	1.36
ALM-2			13211	18138	**17653**	18707	1.37	**1.34**	1.42
ALM-3			13156	18269	**16539**	17789	1.39	**1.26**	1.35
ALM-1		20	19023	**24999**	25387	26312	**1.31**	1.33	1.38
ALM-2			19582	24823	**23935**	25918	1.27	**1.22**	1.32
ALM-3			19245	25468	**23327**	24538	1.32	**1.21**	1.28

For each instance of the above referred ALM class Table 1 presents the best results for both of the genetic versions - gen1 and gen2 - and the best value obtained in [9]. The LB ("Lower Bound") column presents

the theoretical minimum for the value of the objective function - the cost of the tree solution obtained (Sec. 1). The BUB column indicates the best value obtained in [9] for each instance and the following columns the best values obtained using gen1 and gen2. The best of the three values is highlighted. In case none of the genetic versions runs achieved a better result than the one found by Martins and Souza, the better one of the genetic results is underlined. The last three columns show the ratios between each of the algorithm values and the theoretical lower bound, thus indicating a measure of the relative quality of the obtained solution.

From the observation of the results in Table 1 it is noticeable that, for the graph instances presenting larger sets of nodes (from 300 to 500 nodes), the best value (lower cost md-MST solution) was found by one of the evolutionary versions, with only two exceptions and a match for ALM-3 with 400 nodes and $d = 20$.

In order to understand the true significance of these values, we can observe the last three columns of Table 1, where it is clear that the genetic versions obtain lower ratios, with the best values never exceeding 1.3 for the major part of the test instances. This signifies that the quality of the genetic algorithms solutions is, in general, better than the one shown in [9]. As exceptions, we have two ratios for the genetic algorithm values going up to 1.4 but, in these cases, the ratio BUB/LB always exceeds 1.5. Notice that, in the few cases that Martins and de Souza found strictly better quality solutions, the difference from our version values lies in the second decimal place (with only one exception in Alm-1, with 100 nodes and $d = 5$).

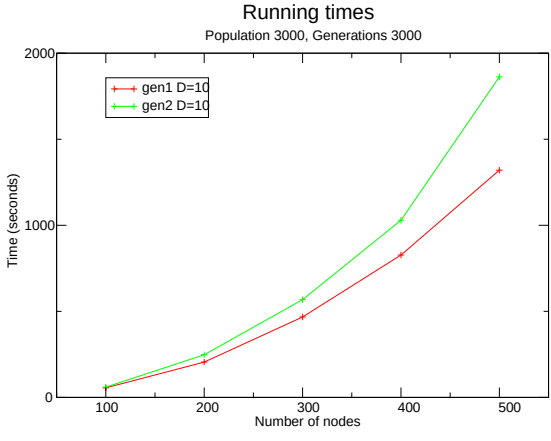


Figure 3. Comparison between average times for the genetic algorithms over the ALM class instances, using $d = 10$.

The most intriguing result presented by Martins and de Souza, especially for higher dimensional instances, is that the running times of their heuristic versions vary immensely, without presenting any relation with any of the instance parameters (the value of d or the set of nodes dimensions). In contrast, the stability of the genetic approach time performance is perfectly clear, increasing in a direct proportion with the number of nodes in the instances graph (Fig. 3). For instance, in the case of ALM-1 with 300 nodes and $d = 10$, Martins and Souza report that the best solution (with a value of 13 899) was obtained after 12 855 seconds. The corresponding best result found by the genetic algorithms (*gen1* using ten repetitions, a population size of 1000 and the number of generations limited to 500) is the value 13 701 after only 1 369 seconds. Thus a better result was obtained using a time 8 or 9 times faster.

In all of the experiments it was noticeable that, for each number of nodes, the different classes of instances did not affect the genetic algorithms. Moreover, there was no direct relation between the running times and the increase in the value of the parameter d.

Although it may seem arguable to directly compare our running times with those found in [9] this argument is easily refuted. The instance tests in the latter reference were run in a computer with a Pentium IV 3.2 GHz processor. Thus for a fairer comparison we used a computer with a Pentium IV 3.0 GHz processor, therefore, slightly slower than the one used by Martins and Souza. The RAM capacity has no consequence here since the instances dimensions are rather small and never need to use more than a tiny amount of the RAM capacity.

4. Conclusions

This paper presents an alternative approximate algorithmic approach to the resolution of the NP-hard problem *Min-Degree Constrained Minimum Spanning Tree (md-MST)*. Motivated by the success of the evolutionary approaches when used over combinatorial problems, two novel genetic algorithms were proposed. The results obtained with the new algorithms are very promising, being computationally consistent on the increase of running times with the instances dimensional node increase. Over higher instance dimensions, the genetic versions present better results than those found in the related literature. Furthermore, our approach presents very competitive results in less time (in general) than those presented in [9]. This turns the genetic algorithms a more effective approach and promising method to tackle the md-MST problem.

However some open questions remain. Presently we are working on a new chromosome representation based on a suggestion found in [12] for

using a vector of weights of the nodes to influence Kruskal's algorithm iterations. At the same time, we are also inspecting the representations that can be found in [5, 14] so that to derive a comparative and richer test set for evolutionary approaches to the md-MST.

Another interesting study would consist of changing the master program so that it tries a variety of strategies and automatically focus on the one who is obtaining the best results for the particular instance to be solved.

References

[1] A. de Almeida, P. Martins, and M. de Souza. Min-degree constrained minimum spanning tree problem: Complexity, properties and formulations. *International Transactions in Operational Research*, in print, 2011.

[2] A. de Almeida, P. Martins, and M. de Souza. The md-MST problem is NP-hard for $d \geq 3$. *Electronic Notes in Discrete Mathematics*, 36:9–15, 2010.

[3] I. Akgün and B. Tansel. Min-degree constrained minimum spanning tree problem: New formulation via Miller-Tucker-Zemlin constraints. *Comput. Oper. Res.*, 37(1):72–82, 2010.

[4] R. L. Haupt and S. E. Haupt. *Practical Genetic Algorithms*. John Wiley & Sons, 1998.

[5] J. Knowles and D. Corne. Variable neighborhood search for the degree constrained minimum spanning tree problem. *IEEE T. Evolut. Comput.*, 4(2):125–134, 2000.

[6] M. Krishnamoorthy, A. T. Ernst, and Y. M. Sharaiha. Comparison of algorithms for the degree constrained spanning tree. *J. Heuristics*, 7(6):587–611, 2001.

[7] J. B. Kruskal. On the shortest spanning subtree of a graph and the travelling salesman problem. In *P. Am. Math. Soc.*, 7(1):48–50, 1956.

[8] L. Martinez and A. Cunha. Finding min-degree constrained spanning trees faster with a branch-and-cut algorithm. *Electronic Notes in Discrete Mathematics*, 36:311–318, 2010.

[9] P. Martins and M. de Souza. VNS and second order heuristics for the min-degree constrained minimum spanning tree problem. *Comput. Oper. Res.*, 36(11):2969–2982, 2009.

[10] Z. Michalewicz. *Genetic algorithms + data structures = evolution programs*, 2nd, extended edition. Springer-Verlag, 1994.

[11] C. M. Papadimitriou. *Computational complexity*. Addison-Wesley, 1994.

[12] G. R. Raidl and B. A. Julstrom. A weighted coding in a genetic algorithm for the degree-constrained minimum spanning tree problem. In *Proc. ACM Symposium on Applied Computing*, pages 440–445, 2000.

[13] C. C. Ribeiro and M. C. de Souza. Variable neighborhood search for the degree constrained minimum spanning tree problem. *Discrete Appl. Math.*, 118(1-2):43–54, 2002.

[14] S.-M. Soak, D. W. Corne, and B.-H. Ahn. Variable neighborhood search for the degree constrained minimum spanning tree problem. *IEEE T. Evolut. Comput.*, 10(2):124–144, 2006.

BENCHMARKING OF THE ASYMPTOTIC GENETIC ALGORITHM FOR THE NOISELESS FUNCTION TESTBED

Pavel Galushin, Eugene Semenkin
Siberian State Aerospace University, Krasnoyarsk, Russia
galushin@gmail.com; eugenesemenkin@yandex.ru

Abstract This paper describes a new estimation of distribution algorithm for pseudo-boolean optimization – Asymptotic Genetic Algorithm, which uses genetic operators, effecting not on particular solutions, but on the probabilities distribution of solution vector's components. Proposed algorithm considers interaction between variables, using technique of generating random vectors with given covariance matrix and marginal distributions. Performance of the proposed algorithm was measured using BBOB COCO (COmparing Continuous Optimizers) platform.

Keywords: Cholesky decomposition, Correlation matrix, Estimation of distribution algorithms.

1. Introduction

Estimation of distribution algorithms (EDA) are the family of optimization methods, inspired by evolutionary optimization algorithms and the theory of machine learning. The first algorithm of this type was the Population-Based Incremental Learning (PBIL) algorithm, proposed by Baluja [2]. PBIL is similar to the genetic algorithm (GA) and preserves the basic properties of the genetic operators, but is defined in terms of the theory of incremental learning [2]. The PBIL explicitly (as opposed to the traditional GA) computes the components of the vector of genes probabilities and updates it using incremental learning.

Other well-known EDA is the Bayesian Optimization Algorithm [4]. This optimization algorithm models the probabilistic distribution of promising solutions using technique of Bayesian networks: the structure of variable dependencies is represented by the directed oriented graph (DAG), vertexes of this graph correspond to variables and edges represent dependencies between variables. Baluja also mentioned [2] that

this is possible to develop an optimization method with mutation and selection operators, effecting not on particular individuals, but on the gene values distribution as a whole, and preserving statistical properties of the standard genetic operators. The purpose of this study is the developing such an optimizer.

Sections 2 and 3 investigate statistical properties of mutation and selection operators of the traditional GA, respectively. Based on the obtained results we propose so called asymptotic mutation and asymptotic selection procedures. Section 4 describes a method, based on Cholesky decomposition of covariance matrix, incorporating variable dependencies into the proposed algorithm. Section 5 measures performance of the proposed algorithm, using benchmark problems from COCO (COmparing Continuous Optimizers) platform [3].

2. Mutation

Let us consider the mutation procedure of GA. Suppose during selection procedure N vectors of dimensionality M $X_1, \ldots, X_N$ was chosen, $X_k = \{x_{k1}, ..., x_{kM}\}$. Let us introduce a vector of average values of each vector's component $P = \{p_1, \ldots, p_M\}$:

$$p_i = \frac{1}{N} \sum_{k=1}^{N} x_{ki}.$$

Vector, obtained by traditional mutation procedure from X_k, will be denoted as X'_k. Vectors X'_k are random, since they are generated by the stochastic procedure of mutation. The arithmetic mean for X_k will be denoted as $P' = \{p'_1, \ldots, p'_M\}$:

$$p'_i = \frac{1}{N} \sum_{k=1}^{N} x'_{ki}.$$

It can be shown that

$$E\left[p'_i\right] = r + (1 - 2r)\, p_i,$$

$$\mathrm{Var}\left[p'_i\right] = \frac{r\,(1 - r)}{N},$$

where r is mutation probability.

It is easy to see, that the variance tends to zero if $N \to \infty$. Replacing p'_i by the expected value, we obtain the mutation procedure, effecting not on particular solutions, but on the probability distribution. This procedure can be called *asymptotic mutation*, since it is a limit of the

traditional mutation procedure, when $N \to \infty$. This procedure has following properties:

i. $p'_i \in [r, 1-r]$.

ii. Value $p = 0.5$ is a fixed point for the asymptotic mutation.

iii. Memory and CPU cycles consumption of the program implementation of the asymptotic mutation does not depend on the population size.

iv. The program implementation of the asymptotic mutation does not include any conditional statements.

v. The program implementation of the asymptotic mutation does not include any random numbers generation.

The forth and fifth properties are very useful for the modern computers' CPU with the pipeline architecture and multiple cores: the pseudo-random numbers generators have hidden state and can't be used by several threads of the execution concurrently, and conditional statements often prevent the instruction pipelining [1]. As the result, the asymptotic mutation procedure is sufficiently faster than traditional mutation (especially in the case of its naive implementation).

3. Selection

Same analysis can be done for the selection procedure. Let us suppose, that solution vectors $U_1, \ldots, U_N$, $U_k = \{u_{k1}, \ldots, u_{kM}\}$ are random and the probability of selection passing in one trial)is assigned to each of them: $g_1, \ldots, g_N$. As the rule, these probabilities are calculated during the traditional selection procedure. For example, in the proportional selection g_k is proportional (in the case of maximization) to the

$$f_k - \min_{i=1,\ldots,N}\{f_i\},$$

where f_i is the objective function value for the i-th solution on the current iteration. In the ranking selection g_k depends on the rank of k-th solution's objective value in the sample of all solution' objective values an so on. For the tournament selection these probabilities are implicit, we will obtain their values later in this section.

It can be proven, that

$$E\left[p_i\right] = \sum_{k=1}^{N} g_k u_{k,i}.$$

$$\mathrm{Var}\left[p_i\right] \leq \frac{1}{4N},$$

where $E\left[\cdot\right]$ is the expected value (mean), and $\mathrm{Var}\left[\cdot\right]$ is the variance.

As in the case of the mutation, the variance of the selection procedure's result tends to zero, when $N \to \infty$.

Selection method based on this fact can be called *asymptotic selection*. For the proportional and ranking selection probabilities of passing selection $g_1, \ldots, g_N$ are computed explicitly, therefore they can be used for the asymptotic selection without any modifications. Optimization algorithm, similar to PBIL, but using asymptotic mutation and selection can be called *Asymptotic genetic algorithm* (AGA). Using pseudo-code, AGA can be expressed as following:

Algorithm 1 AGA

$p_i = \frac{1}{2}$, $i = 1, \ldots, M$.
while not EndingCondition() **do**
  Create N boolean vectors $U_1, \ldots, U_N$ such, that $P\left\{u_{ki} = 1\right\} = p_i$.
  $y_k = f(U_k)$, $k = 1...N$
  Compute probabilities of passing selection $g_1, \ldots, g_N$.
  Compute $p_i = \sum_{k=1}^{N} g_k u_{k,i}$, $i = 1, \ldots, M$.
  $p'_i = r + (1 - 2r)p_i$, $i = 1, \ldots, M$.
  $p_i \leftarrow p'_i$, $i = 1, \ldots, M$.
end while

EndingCondition() can be an arbitrary condition, for example: the number of iteration is greater than the given value.

In the conclusion of this section, we consider problem of incorporating tournament selection into the asymptotic selection procedure. In the tournament selection, probabilities of passing selection are not calculated explicitly, but the tournament selection is widely known to be more effective than other selection methods.

Let the population to be consisted of N solutions with fitness function values $y_1, \ldots, y_N$. It is necessary to calculate the probabilities of passing selection $g_1, \ldots, g_N$ in the tournament selection procedure (tournament group size will be denoted as t). We will assume, that some of the values y_k are equal to each other. Let us also assume that among the values $y_1, \ldots, y_N$ there exist K unique values: $Y_1, \ldots, Y_K$, and the value Y_k has n_k instances, sum of all numbers n_k is N:

$$\sum_{k=1}^{K} n_k = N.$$

Let values Y_k to be sorted in the increasing order. The probability of passing selection by the solution with the fitness value Y_k will be denoted as b_k. We also introduce cumulative probabilities:

$$B_k = \sum_{j=1}^{k} b_j.$$

By the definition, B_k is the probability of such an event, when the solution with fitness value less or equal than Y_k will be selected. As the tournament group creation does not take into account fitness values, this probability is equal to the number of possible tournament groups, which contains only solution with fitness less or equal to Y_k, divided by total number of all possible tournament groups. We will denote number of tournament groups of size t, chosen from population of size m, as $s(m,t)$. This number depends on a particular method (see below) of tournament group creation. Let us consider following partial sums:

$$\sum_{j=1}^{k} n_j = N_k.$$

It is easy to see, that

$$B_k = \frac{s(N_k,t)}{s(N,t)}.$$

After cumulative probabilities are known, we can compute b_k:

$$b_1 = B_1,$$

$$b_k = B_k - B_{k-1},\ 2 \le k \le K.$$

Since value Y_k has n_k instances, and all solutions with the same fitness value should has equal probability to pass the selection, then

$$g_k = \frac{b_j}{n_j},\ j : Y_j = y_k,\ k = 1, \ldots, N.$$

Depending on the certain method of the tournament group creation, the function $s(k,t)$ has two forms. If one solution can be chosen to the tournament group several times, this function has simple form:

$$s_1(k,t) = k^t.$$

If each of solutions can be chosen to the tournament group only once, then the function definition is more complex:

$$s_2(k,t) = \frac{k!}{(k-t)!}.$$

It can be shown, that if the population size N is sufficiently big and t is small (compared with N), the difference between distributions generated by these two functions is not statistically significant.

4. Variables Dependencies

AGA, as it was described above, considers only marginal distributions of variables. This section discusses the method taking into account dependencies between variables. We use the correlation matrix as the measure of dependency. In the context of EDA optimizers, it is necessary to develop the procedure of the synthesis of the random vectors generator with given marginal distributions, each of them determined by single parameter

$$p_i = P\left\{x_{ki} = 1\right\},\ i = 1, \ldots, M,$$

and covariance matrix C with components

$$C_{ij} = M\left[x_{ki}x_{kj}\right] - p_i p_j,\ i,j = 1, \ldots, M.$$

In the AGA, elements of the covariance matrix are calculated as weighted sum

$$C_{ij} = \frac{1}{N}\sum_{k=1}^{N} g_k \cdot x_{ki} \cdot x_{kj}^T - p_i \cdot p_j.$$

The first step is the computation of the correlation matrix R with components:

$$\rho_{ij} = \frac{C_{ij}}{\sqrt{C_{ii} \cdot C_{jj}}},\ i,j = 1, \ldots, M.$$

Then statistically insignificant components are replaced by zero. The correlation matrix can be represented, using Cholesky decomposition, as $R = S^T S$, where S is the upper-triangular matrix.

It also necessary to compute threshold values to transform standard normal distribution to the Bernoulli distribution:

$$t_i = \Phi_{inv}\left(p_i\right)\ i = 1, \ldots, M,$$

where Φ_{inv} is inverse CDF of the standard normal distribution.

The procedure of the random vector generation can be expressed using pseudo-code in the following way (we use Iverson's notation, $[A]$ is equal to 1, if A is true, and 0 otherwise):

Algorithm 2 Random vector generation

1: generate M independent standard normal random variables z_i, they consist random vector $Z = \{z_1, \ldots, z_M\}$.
2: compute $Y = SZ$.
3: $x_i = [y_i < t_i]$, $i = 1, \ldots, M$.

It should be noted, that the covariance matrix of vectors, generated by this procedure, is not equal to C due to the discretization. But it is not the important drawback, since it is necessary to determine the structure of inter-variable dependencies, and not particular covariance values.

To use this procedure in the AGA we need to find out a form of transformation of the covariance matrix after mutation. Let X_k be an independent and identically distributed random vector with components $\{x_{ki}\}$ and

$$E[x_{ki}] = p_i,$$

$$\mathrm{Cov}(x_{ki}, x_{kj}) = C_{ij}.$$

Covariance after mutation will be denoted as C'_{ij} and mutation probability as r. Then it can be proven, that:

$$C' = (1 - 2r)^2 C + r(1 - r) I,$$

where I is the identity matrix of the dimensionality M. It can be proven, that the condition number of the matrix C' is less or equal to the condition number of the matrix C. Therefore, from the numerical methods view point, the mutation decreases relative error in the Cholesky decomposition.

5. Estimation of the Proposed Algorithm Performance Using Benchmarking Problems

To measure the performance of the proposed algorithm we used benchmark functions from platform COmparing Continuous Optimizers [3]. AGA was compared with two other Estimation of distribution algorithms: Bayesian optimization algorithm (BOA) and Population Based Incremental Learning (PBIL). Results are collected in Tables 1 and 2. In these tables Δf is the median difference of the function value found from the optimal function value, reached by the given algorithm, RT_{succ} is the median number of the function evaluations to get best function value. Results are shown for 5-dimensionality and 20-dimensionality functions.

Table 1. Performance on the COCO Benchmarks problems in 5-D.

Algorithm	AGA	AGA	PBIL	PBIL	BOA	BOA
Function	Δf	RT_{succ}	Δf	RT_{succ}	Δf	RT_{succ}
f1	6.5e-3	2.9e3	5.8e-3	4.8e3	7.6e-3	2.6e3
f2	1.9e-3	6.9e3	1.9e-2	2.2e4	1.9e-2	7.9e3
f3	1.0e0	1.0e4	8.6e-3	1.7e4	8.5e-3	7.6e3
f4	4.0e0	8.9e3	1.0e0	2.3e4	5.0e0	1.3e4
f5	7.6e-3	6.5e3	1.5e-4	2.0e4	8.5e-3	5.9e3
f6	2.7e0	1.3e4	8.7e-3	2.6e4	8.9e-3	4.3e4
f7	3.1e-1	6.2e3	5.9e-3	1.1e4	1.7e-2	6.7e3
f8	2.2e0	1.3e4	1.6e0	4.9e4	1.1e0	5.0e4
f9	3.7e0	1.1e4	2.7e0	4.5e4	2.3e0	5.0e4
f10	2.0e1	3.2e4	7.0e1	4.5e4	2.5e1	5.0e4
f11	3.9e0	4.0e4	6.5e0	3.2e4	4.0e0	4.5e4
f12	2.0e0	8.9e3	1.4e0	1.6e4	2.9e0	7.9e3
f13	1.1e0	1.0e4	2.0e0	1.8e4	7.2e-1	8.9e3
f14	5.8e-3	3.6e3	6.8e-3	6.1e3	6.1e-3	3.3e3
f15	2.0e0	3.2e4	2.0e0	4.0e4	1.1e0	3.2e4
f16	1.7e0	2.5e4	1.2e-2	3.2e4	1.6e0	3.2e4
f17	8.9e-3	9.6e3	8.9e-3	2.0e4	8.9e-3	7.1e4
f18	2.9e-3	1.3e4	4.8e-2	2.0e4	7.2e-2	1.3e4
f19	2.5e-1	4.5e4	1.5e-1	3.5e4	2.6e-1	4.0e4
f20	7.7e-3	8.0e3	2.4e-1	1.8e4	8.8e-3	1.0e4
f21	1.2e0	5.7e3	6.9e-3	5.3e3	6.6e-3	1.1e4
f22	6.8e-1	1.1e4	6.9e-1	2.5e4	6.9e-1	8.9e3
f23	6.1e-1	2.2e4	7.6e-1	3.5e4	7.4e-1	3.2e4
f24	7.5e0	3.2e4	6.6e0	3.5e4	7.8e0	4.0e4

Proposed AGA is the pseudo-boolean optimizer, and optimization of the functions with real variables is just benchmarking. The aim was to prove the workability of approach proposed and not demonstrate high performance relatively real functions optimizers.

Since considered algorithms are in essence pseudo-boolean algorithms (global optimum may have no representation in the selected encoding scheme) and are not tailored to optimize functions with real variables, search process was halted after $\Delta f < 0.01$ is reached, and then direct comparison with continuous optimizers were not performed.

The best algorithm for given function is the algorithm, which surpass minimal Δf of all algorithms, if more than one algorithm surpass minimal Δf, than the best is the algorithm with minimal RT_{succ}.

For all algorithms the maximum fitness functions evaluation number was equal to 49952 for the dimensionality 5 and 200256 for the dimensionality 20. Gray code was used to encode real numbers as boolean vectors, grid step was equal to 0.001. Following settings was used. AGA: tournament selection (group size is equal to 2), mutation probability is equal to $\frac{1}{M}$. For PBIL the mutation probability was the same as in the AGA, positive learning shift was equal to 0.1, negative learning shift –

0.05, mutation shfit – 0.1. For the BOA maximum number of graph's edges was equal to 5; the greedy algorithm was used to construct the graph, Bayesian Dirichlet metric (without prior information) was used [4] to estimate the quality of graph, the selection ratio was equal to 0.5.

Obtained results show, that AGA outperforms PBIL in the case of the dimensionality 5 on the benchmark problems f2, f10, f11, f13, f14, f15, f17, f18, f20, f22, f23: 11 of the total 24. AGA outperforms BOA in the case of the dimensionality 5 on the benchmark problems f1, f2, f4, f5, f10, f11, f12, f14, f17, f18, f19, f20, f22, f23, f24: 15 of the total 24.

Table 2. Performance on the COCO Benchmarks problems in 20-D.

Algorithm	AGA	AGA	PBIL	PBIL	BOA	BOA
Function	Δf	RT_{succ}	Δf	RT_{succ}	Δf	RT_{succ}
f1	7.1e-3	2.0e+4	3.2e-2	1.0e5	7.5e-3	1.6e+4
f2	6.1e-2	5.0e+4	1.3e+0	1.8e+5	4.2e-3	4.5e+4
f3	8.0e+0	5.0e+4	1.5e+1	1.3e+5	8.0e+1	5.0e+4
f4	2.1e+1	5.0e+4	2.3e+1	1.6e+5	1.9e+1	1.0e+5
f5	8.5e-3	3.5e+4	2.1e+0	1.0e+5	7.9e-3	3.1e+4
f6	2.9e+1	7.1e+4	2.4e+1	1.8e+5	1.0e+0	2.0e+5
f7	4.5e+0	5.6e+4	2.1e+0	8.9e+4	3.9e+0	5.6e+4
f8	7.6e+0	5.6e+4	9.3e+0	1.0e+5	1.7e+1	2.0e+5
f9	2.3e+1	5.6e+4	2.7e+1	8.9e+4	1.8e+1	2.0e+5
f10	5.7e+3	1.8e+5	1.8e+3	1.0e+5	7.6e+3	2.0e+5
f11	6.7e+1	2.0e+5	2.7e+1	1.3e+5	5.6e+1	2.0e+5
f12	9.0e+0	4.5e+4	1.3e+4	1.0e+5	1.3e+0	4.5e+4
f13	3.0e+1	5.6e+4	4.6e+1	1.0e+5	1.6e+0	1.3e+5
f14	8.8e-3	3.1e+4	3.1e-2	1.6e+5	8.7e-3	2.0e+4
f15	8.1e+1	1.6e+5	4.3e+1	1.0e+5	8.2e+1	1.3e+5
f16	1.6e+1	1.3e+5	8.6e+0	1.0e+5	1.7e+1	7.1e+4
f17	1.3e-2	7.1e+4	2.4e-1	1.6e+5	1.1e-2	7.1e+4
f18	2.8e-1	8.9e+4	4.8e-2	2.0e+4	1.7e-1	7.9e+4
f19	3.5e+0	1.1e+5	2.6e+0	1.3e+5	3.7e-1	1.3e+5
f20	4.6e-1	7.1e+4	8.1e-1	8.9e+4	4.3e-1	7.9e+4
f21	2.1e+0	5.0e+4	2.2e+0	1.4e+5	2.1e+0	7.1e+4
f22	2.0e+0	5.0e+4	2.0e+0	8.9e+4	2.0e+0	4.5e+4
f23	1.9e+0	1.0e+5	1.5e+0	1.3e+5	2.1e+0	1.1e+5
f24	9.9e+1	1.1e+5	8.2e+1	1.2e+5	1.0e+2	1.4e+5

Table 2 shows that proposed Asymptotic Genetic Algorithm can be used for the complex functions optimization. AGA outperforms PBIL in the case of the dimensionality 20 on the benchmark problems f1, f2, f3, f4, f5, f8, f9, f12, f13, f14, f17, f20, f21, f22: 14 problems of the total 24 ones. AGA outperforms BOA in the case of the dimensionality 20 on the benchmark problems f1, f10, f15, f16, f23, f24: 6 of the total 24 ones.

6. Conclusion

Asymptotic Genetic Algorithm (AGA) is an estimation of distribution algorithm, which preserves statistical properties of the conventional genetic algorithm, but replaces traditional genetic operations of the mutation and selection by the asymptotic mutation and selection operators, effecting on the whole components values' probability distribution. To take into account dependencies between variables, the technique of the sampling random vectors with given covariance matrix and given marginal distributions was used.

We can conclude, that in the case of the dimensionality 5 AGA and PBIL have approximately equal performance, and both outperform BOA. In the case of dimensionality 20 BOA outperforms AGA, which in turn is better than PBIL.

References

[1] A. V. Aho, M. S. Lam, R. Sethi, and J. D. Ullman. *Compilers: Principles, Techniques, and Tools*, 2nd edition. Addison-Wesley, 2007.

[2] S. Baluja. Population-Based Incremental Learning: A method for integrating Genetic Search Based Function Optimization and Competitive Learning. Technical Report. Carnegie Mellon University, Pittsburgh, 1994.

[3] S. Finck, N. Hansen, R. Ros, and A. Auger. Real-parameter black-box optimization benchmarking 2009: Presentation of the noiseless functions. Technical Report 2009/20, Research Center PPE, 2009.

[4] M. Pelikan, D. E. Goldberg, and E. Cantu-Paz. BOA: The Bayesian optimization algorithm. In *Proc. Genetic and Evolutionary Computation Conference*, volume 1, pages 525–532, 1999.

PARAMETER CONTROL OF EAS USING EXPLORATION AND EXPLOITATION MEASURES: PRELIMINARY RESULTS

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

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

Adam Cardenas
Department of Computer Science, California State University, Fresno, USA
alcardenas@mail.fresnostate.edu

Abstract This paper introduces a parameter control approach that utilizes recently introduced exploration and exploitation measures as feedback to adaptively control evolution processes. Schwefel's Problem 2.22 and Shekel's Foxholes Function are selected as benchmark functions to show the competence and usefulness of the measures. The paper shows that with new exploration and exploitation measures, the evolution processes generate relatively well results in terms of fitness and/or convergence rate.

Keywords: Exploitation, Exploration, Parameter control.

1. Introduction

Exploration and exploitation [5] are two essential cornerstones of Evolutionary Algorithms (EAs) that drive an evolution process toward optimization and/or convergence. Exploration is defined as visiting entirely new regions of a search space, whilst exploitation is defined as visiting those regions of a search space within the neighborhood of previously visited points [3]. However, to our best knowledge, not until exploration

and exploitation measures using an ancestry tree approach [2] were recently introduced, there had not been a quantitative way to measure and analyze how exploration and exploitation influence and balance the inner work of an evolution process. Our previous work in [2] primarily focused on introducing exploration and exploitation measures and validating the usefulness of such measures on analyzing a multi-objective EA. In [8] we further investigated the inner work of VEGA [11] and SPEA2 [14] applied to the 0/1 Knapsack problem using exploration and exploitation measures. With such measures, the outperformance of SPEA2 over VEGA can be also explained in a finer-grained manner.

Given the usefulness on investigating the inner work of EAs using exploration and exploitation measures, this paper attempts to validate a hypothesis: By using finer-grained exploration and exploitation measures as feedback, an evolution process adapted by parameter control approaches [4] may perform relatively competitive or generate even better results in terms of optimization and/or convergence on selected benchmark functions. To validate the hypothesis, this paper extends the implementation of an existing domain-specific language [10], called PPCea (Programmable Parameter Control for Evolutionary Algorithms) [7], so that all exploration and exploitation measures from [2] can be computed by the PPCea interpreter on-the-fly. With such, users may introduce PPCea programs to adaptively control evolution processes using the measures as feedback. Although the experimental results are compared under *one single different* experimental setting (i.e., the number of runs), the promising data raise hope to invest more time on conducting more experiments with various benchmark functions and on introducing various adaptive (or self-adaptive [4]) algorithms using exploration and exploitation measures.

The paper is organized as follows. Section 2 reviews the ancestor-tree approach. Section 3 shows how PPCea is applied to control EAs along with the experimental results of two benchmark functions, followed by the conclusion in Section 4.

2. The Ancestry Tree Approach

The ancestry tree approach borrows the ideas from genealogy. An ancestry tree describes the history of all individuals that were created during the evolution process: An individual's parent, its representation (genome), as well as how (e.g., by mutation, by crossover) and when (generation) the individual was created are all stored during the construction of ancestry trees. For example, in Fig. 1 (left), τ_2 represents the 2^{nd} individual generated at the initial stage. During each gener-

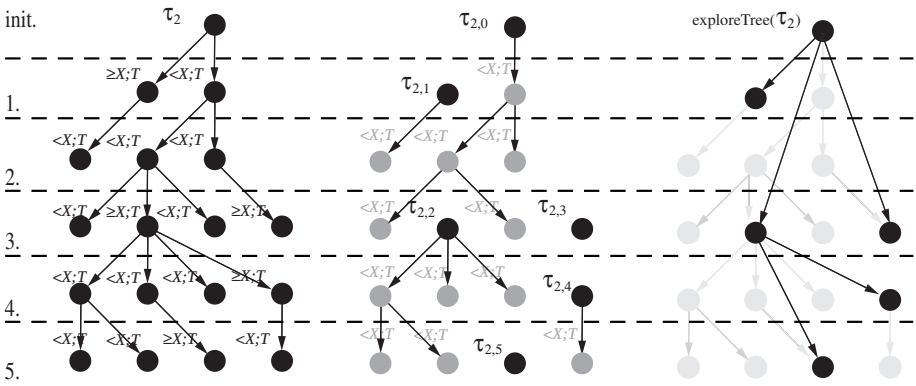


Figure 1. Splitting process of $\tau_2^{(x)}$ (left) and generation of exploitation (middle) and exploration trees (right) $\tau_{2,j}^{(x)}$.

ation, if a candidate child (generated by mutation (m), crossover (c), random creation (rnd), repair (r), or cloning (cln), etc.) survives after selection, the ancestry tree will be expanded to further depth until no more offspring is survived. But, how can we see which node is a result of exploration or exploitation? In [2], a number x is introduced to delimit exploration from exploitation and can be seen as identification of an individual's neighborhood. Because x is problem dependent, various diversity measures (e.g., hamming distance and Euclidean distance) can be used for this purpose (This paper chooses Euclidean distance and sets $x = 0.05$ based on a number of experiments). When the selected diversity measure is larger than x, the splitting process will split an ancestry tree into several subtrees (see Fig. 1 (middle)), called exploitation trees, because all nodes, except the root node, are obtained by exploitation of neighborhood regions. Such a process can be observed in Fig. 1: There are six exploitation trees in Fig. 1 (middle). For each exploitation tree, gray colored nodes represent exploitation nodes (obtained by exploiting within the individual's neighborhood) and black root node represents exploration node (obtained by exploring outside the neighborhood). As for exploration tree, it is constructed by linking all the black root nodes of Fig. 1 (middle) together (see Fig. 1 (right)) With such, a number of exploration and exploitation measures can be defined based on the characteristics of exploration and exploitation trees. For space consideration, we concisely summarize the measures as follows. Readers who are interested in the comprehensive definitions and mathematic formulae of such measures may refer to [2].

Exploration ratio (*exploreRatio*) can be defined as the percentage of nodes obtained by exploration (black nodes) over all nodes in all ancestry trees. And exploitation ratio (*exploitRatio*) can be defined as $1-$ *exploreRatio*. An ancestry tree also records evolution operators that were involved in individual creation/evolution. Given this, the involvement of each operator in exploration/exploitation is defined as the ratio between number of exploration/exploitation nodes obtained using a particular operator and all exploration/exploitation nodes regardless of operators. For example, *exploreType(c)* means the ratio that crossover is involved in exploration; and *exploitType(m)* measures the ratio that mutation is involved in exploitation.

Figure 1 (right) shows the creation of an exploration tree from the ancestry tree $\tau_2^{(x)}$. It shows that the neighborhood of $\tau_2^{(x)}$ has been changed five times, three times from the initial individual that represents the first level ($depth = 1$) of exploration and two times from the second level ($depth = 2$) of exploration. We define two measures on exploration trees. The first measure, *exploreGap*, calculates the average number of generations that was needed to change neighborhood. Namely, how many generations in average are needed to identify new and unexplored parts of the search space. For example, *exploreGap* of Fig. 1 is computed as $(1 + 3 + 2 + 1 + 3)/5$. The second measure, *exploreProgressiveness*, calculates the average depth of nodes in exploration trees. Namely, it measures the progressiveness of exploration. For example, *exploreProgressiveness* of Fig. 1 is computed as $(1 + 1 + 1 + 2 + 2)/5$. Measure *exploitProgressiveness* is similar to *exploreProgressiveness*, but this metric is defined on an exploitation tree.

Conversely, from exploration and exploitation trees' wideness (*numberOfLeafs*), we can describe the influence of selection on exploration and exploitation ($exploreSelectionPressure$ and $exploitSelectionPressure$). If same individual was selected more than once, an ancestry tree becomes wider. Also, revisiting the same search point should be avoided as much as possible, since the processing time on re-evaluating the same solutions should be minimized. $nonRevisitedRatio(\tau)$ gives us the ratio between all unique individuals/nodes and all individuals/nodes in ancestry trees. The following section presents how such measures can be used to control and analyze EAs in a refined way.

3. Experimental Results Run by PPCea

Before presenting experimental results of f_2 and f_{14} from [13], this section reviews PPCea and then presents the experimental results. As mentioned before, PPCea [7] is a domain-specific language for EAs. The

language comprises statements for general purposes (e.g., conditions, loop, and assignment statements) and statements and parameters specific for EAs (e.g., initialization, mutation, crossover, selection, resize, and evaluation, mutation and crossover rates, population size, among others). PPCea has been used to reproduce a number of parameter tuning, deterministic [4] and adaptive parameter control EAs (e.g., DGEA, PROFIGA and GAVaPS in [9]). In this paper, all the exploration and exploitation measures are newly implemented and integrated in PPCea interpreter and can be computed on-the-fly during an evolution process. An example PPCea pseudocode to control EAs using exploration and exploitation measures (i.e., EE-driven approach) can be seen in the following code snippet (Algorithm 1). Note that the code snippet encores the generally agreed concept of "exploration and then exploitation." The programmable fashion of PPCea actually enables users to introduce much more complex adaptive algorithms using exploration and exploitation measures. We leave such potentials as our future work.

Algorithm 1 EE-Driven Approach using PPCea

```
Initialize all variables
while not reaching Maximum Round do
  init; //initialize population
  while not reaching Maximum Generation do
    callGA; //invoke a GA
    if exploreRatio > 0.1 then
      pm := pm * 1.22 //pm is mutation rate
    else
      pm := pm * 0.82
    end if
    Increment to Epoch generation(s), a user-defined stride
  end while
  Increment to next Round
end while
```

The pseudocode (Algorithm 1) expresses that when an EA is performing more exploration during an evolution process (i.e., *exploreRatio* > 0.1), which usually occurs in the beginning phase, PPCea increases the mutation rate (pm) to further explore unvisited regions of the search space. Conversely, the mutation rate will be decreased when less exploration influences the evolution process, which usually occurs at the later phase. As for coefficients for changing mutation rate (i.e., 0.82 and 1.22), they are set as the same as those in 1/5 success rule [1] and entropy-driven approaches [7] (briefly introduced later) for fair comparisons. The initial values of the parameters needed for *all* approaches are set the same. Namely, mutation rate (pm) $= 0.005$, crossover rate

$pc = 0.75$, population size $popsize = 100$, and total number of experiments $round = 5$ for f_2 and 50 for f_{14}. Maximum generation of each function ($maxgen$) is set as the same in [13] (i.e., 2000 for f_2 and 100 for f_{14}). Threshold 0.1 for increasing or decreasing pm is set based on empirical studies on both functions, as a compromise for identical experimental setting purpose. Because the value is problem-specific, more suitable threshold values for individual functions may be found by more studies. This paper does not concentrate on such discussions. Table 1 shows the benchmark functions selected in this paper due to time and space consideration. Our goal in the near future is to finish experiments with all various kinds of benchmark functions from [7, 13].

Table 1. Selected benchmark functions.

Test Function	n	S	f_{min}
$f_2(x) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	30	$[-100, 100]^n$	0
$f_{14}(x) = [\frac{1}{500} + \sum_{j=1}^{25} \frac{1}{j + \sum_{i=1}^{2} (x_i - a_{ij})^6}]^{-1}$	2	$[-65.363, 65.363]^n$	1

Before showing the experimental results, we first briefly introduce the approaches that will be used for comparisons. The Schaffer approach is a parameter tuning [4] approach that fixes the rates of mutation (0.005) and crossover (0.75) during an entire evolution process; The Fogarty approach is a deterministic one [4] that predefines a formula for mutation rate, which will be gradually decreased when the generation is incremented; The 1/5 success rule is an adaptive approach [4] – When the ratio of successful mutations is above 1/5, mutation rate will become 1.22 times larger. Conversely, if the ratio of successful mutations is smaller than 1/5, mutation rate will be 0.82 times smaller than before. The entropy-driven approach is an adaptive one – when entropy measure (representing disorder of an evolution process) is above a threshold, mutation rate is decreased to encourage exploration. Conversely, when the entropy measure is below the threshold, mutation rate is decremented to prohibit exploration.

Table 2 shows the experimental results of f_2 using Schaffer [12], Fogarty [6], 1/5 success rule [1], entropy-driven [7] and EE-driven approaches. Because convergence rate is also an important performance indicator for unimodal functions, the results are shown within the parentheses next to the *Best* result of each approach. Again, one of the main objectives of this paper is to utilize EE measures to explain the results of different

approaches in a finer-grained manner. We will show such explanations in the following paragraphs. Note that there are some EE measures not presented in the table. Since they are relatively the same to all approaches, for space consideration we ignored such measures.

Table 2. Schaffer, Fogarty, 1/5 Success Rule, Entropy, and EE approaches for f2.

Measure \ *gen*	1	250	500	750	1000	1250	1500	1750	2000
Schaffer									
Best (675)	1.43E+11	1.11E-2	3.45E-4	1.45E-4	1.43E-4	1.43E-4	1.43E-4	1.43E-4	1.43E-4
exploreRatio	84.20%	4.01%	2.01%	1.34%	1.00%	0.80%	0.61%	0.57%	0.50%
Type(m)	29.43%	49.46%	49.46%	49.46%	49.46%	49.46%	49.46%	49.46%	49.46%
exploreGap	1.00	1.45	1.45	1.45	1.45	1.45	1.45	1.45	1.45
exploreProg.	1.00	9.00	9.00	9.00	9.00	9.00	9.00	9.00	9.00
exploreSel.P.	88.73%	76.52%	76.52%	76.52%	76.52%	76.52%	76.52%	76.52%	76.52%
exploitProg.	1.23	96.74	220.72	345.39	470.22	595.12	720.06	845.01	969.97
exploitSel.P.	15.80%	28.75%	19.41%	15.80%	13.84%	12.67%	11.89%	11.32%	10.90%
nRRatio.	90.70%	29.38%	17.97%	12.22%	9.17%	7.34%	6.11%	5.24%	4.59%
Fogarty									
Best (741)	2.43E+11	1.19E+2	3.48E-3	1.58E-4	1.43E-4	1.43E-4	1.43E-4	1.43E-4	1.43E-4
exploreRatio	83.80%	10.12%	6.07%	4.71%	4.04%	3.63%	3.36%	3.16%	3.02%
Type(m)	31.16%	55.37%	55.26%	54.87%	54.77%	54.77%	54.77%	54.77%	54.77%
exploreGap	1.00	4.30	3.74	3.34	3.04	2.81	2.63	2.48	2.36
exploreProg.	1.00	4.95	4.28	3.80	3.45	3.17	2.95	2.77	2.62
exploreSel.P.	88.18%	80.49%	83.61%	85.77%	87.39%	88.77%	89.89%	90.80%	91.56%
exploitProg.	1.24	13.28	19.37	21.29	22.24	22.80	23.17	23.43	23.63
exploitSel.P.	16.20%	13.63%	16.19%	15.33%	13.99%	13.10%	12.50%	12.09%	11.76%
nRRatio.	83.80%	14.38%	14.88%	8.48%	8.81%	6.78%	5.65%	4.84%	4.24%
1/5									
Best (951)	9.44E+10	7.27E-2	9.76E-3	1.22E-3	1.49E-4	1.43E-4	1.43E-4	1.43E-4	1.43E-4
exploreRatio	82.10%	5.63%	3.82%	3.21%	2.91%	2.73%	2.61%	2.52%	2.46%
Type(m)	28.92%	50.48%	50.57%	50.22%	50.08%	50.08%	50.08%	50.08%	50.08%
exploreGap	1.00	1.29	1.21	1.17	1.14	1.12	1.10	1.09	1.08
exploreProg.	1.00	6.67	5.11	4.22	3.65	3.17	2.95	2.73	2.55
exploreSel.P.	88.29%	78.32%	83.86%	87.19%	89.38%	90.91%	92.07%	92.97%	93.69%
exploitProg.	1.26	21.47	23.26	23.84	24.11	24.31	24.43	24.51	24.57
exploitSel.P.	17.9%	22.96%	14.88%	12.36%	12.08%	11.64%	11.30%	11.06%	10.87%
nRRatio.	89.80%	23.92%	14.34%	10.99%	8.81%	7.05%	5.88%	5.07%	4.41%
Entropy									
Best (358)	2.56E+10	3.52E-3	1.43E-4	1.43E-4	1.43E-4	1.43E-4	1.43E-4	1.43E-4	1.43E-4
exploreRatio	81.90%	5.43%	3.72%	3.26%	2.86%	2.69%	2.57%	2.49%	2.43%
Type(m)	29.01%	49.52%	48.03%	48.03%	48.03%	48.03%	48.03%	48.03%	48.03%
exploreGap	1.00	1.31	1.22	1.17	1.14	1.12	1.10	1.09	1.08
exploreProg.	1.00	6.13	4.68	3.87	3.35	2.99	2.73	2.53	2.37
exploreSel.P.	87.79%	80.09%	83.80%	87.23%	89.46%	91.03%	92.19%	93.09%	93.80%
exploitProg.	1.27	21.39	23.22	23.82	24.12	24.29	24.41	24.50	24.56
exploitSel.P.	18.1%	40.71%	31.35%	24.09%	20.48%	18.30%	16.84%	15.78%	15.00%
nRRatio.	88.80%	35.15%	19.10%	12.74%	9.56%	7.65%	6.38%	5.47%	4.78%
EE									
Best (469)	1.59E+11	2.33E-2	1.43E-4	1.43E-4	1.43E-4	1.43E-4	1.43E-4	1.43E-4	1.43E-4
exploreRatio	82.60%	6.03%	4.02%	3.35%	3.01%	2.81%	2.67%	2.58%	2.51%
Type(m)	29.86%	50.07%	48.90%	48.90%	48.90%	48.90%	48.90%	48.90%	48.90%
exploreGap	1.00	1.31	1.23	1.18	1.15	1.13	1.11	1.10	1.09
exploreProg.	1.00	7.60	5.87	4.86	4.20	3.73	3.38	3.11	2.90
exploreSel.P.	89.48%	78.55%	83.82%	86.47%	88.71%	90.32%	91.52%	92.46%	93.22%
exploitProg.	1.26	21.25	23.16	23.78	24.09	24.27	24.39	24.48	24.55
exploitSel.P.	17.4%	27.38%	25.01%	19.88%	17.32%	15.77%	14.75%	14.00%	13.44%
nRRatio.	89.50%	27.78%	17.50%	11.67%	8.76%	7.01%	5.84%	5.01%	4.38%

First, let us analyze the results of the Schaffer approach: After generation (g) 250, exploration related measures become steady (every exploration related measure is fixed except *exploreRatio*). It means that there is no more splitting process after $g = 250$. Hence, *exploitProgressiveness* grows drastically. Although the Fogarty approach has 6.07% to 10.12% *exploreRatio* between $g = 250$ and 500, the non-revisited ratio (*nRRatio*) indicates that the exploration is not as efficient as other approaches. Possible reasons, based on observing EE measures, may be that muta-

Table 3. Schaffer, Fogarty, 1/5 Success Rule, Entropy, and EE approaches for f14.

Schaffer\gen	1	10	20	30	40	50	60	70	80	90	100
Best (12)	25.12	2.29	2.27	2.23	2.23	2.19	2.19	2.19	2.19	1.99	1.95
exploreRatio	64.25%	14.78%	7.74%	5.25%	3.97%	3.19%	2.67%	2.29%	2.09%	1.79%	1.61%
Type(m)	4.90%	8.96%	8.96%	8.98%	8.99%	8.99%	8.99%	8.99%	9.01%	9.03%	9.04%
exploreSel.P.	89.38%	86.23%	86.54%	86.53%	86.53%	86.51%	86.51%	86.51%	86.51%	86.50%	86.50%
exploitRatio	35.75%	85.22%	92.26%	94.75%	96.03%	96.81%	97.33%	97.71%	97.81%	98.01%	98.05%
exploitProg.	1.42	3.23	7.09	11.58	16.31	21.09	25.84	30.67	35.54	40.31	44.95
exploitSel.P.	35.75%	66.78%	65.89%	56.37%	45.40%	38.22%	33.42%	29.92%	27.36%	24.45%	23.83%
nRRatio.	77.70%	33.10%	20.29%	14.17%	10.72%	8.64%	7.23%	6.22%	5.46%	4.88%	4.40%
Fogarty\gen	1	10	20	30	40	50	60	70	80	90	100
Best (12)	30.32	1.20	1.15	1.15	1.15	1.15	1.15	1.15	1.15	1.15	1.15
exploreRatio	77.34%	33.18%	26.94%	24.70%	23.55%	22.86%	22.39%	22.05%	21.80%	21.60%	21.44%
Type(m)	43.54%	43.61%	41.62%	38.16%	37.90%	37.80%	37.76%	37.72%	37.63%	37.58%	37.57%
exploreSel.P.	87.54%	74.91%	83.47%	85.15%	86.46%	88.72%	90.37%	91.60%	92.55%	93.31%	93.92%
exploitRatio	22.66%	66.82%	73.06%	75.30%	76.45%	77.14%	77.61%	77.95%	78.20%	78.40%	78.56%
exploitProg.	1.31	2.17	2.33	2.38	2.41	2.43	2.44	2.45	2.46	2.46	2.46
exploitSel.P.	22.66%	36.02%	42.41%	40.56%	36.91%	34.53%	32.95%	31.81%	30.98%	30.32%	29.80%
nRRatio.	82.84%	46.43%	34.75%	23.97%	18.13%	14.58%	12.19%	10.47%	9.19%	8.18%	7.37%
1/5 \gen	1	10	20	30	40	50	60	70	80	90	100
Best (13)	30.45	2.30	2.23	2.23	2.23	2.23	2.23	2.23	2.23	2.23	2.23
exploreRatio	64.78%	23.71%	21.95%	21.32%	20.99%	20.80%	20.67%	20.57%	20.50%	20.44%	20.40%
Type(m)	5.00%	9.81%	9.50%	8.69%	8.52%	8.48%	8.48%	8.48%	8.47%	8.47%	8.47%
exploreSel.P.	89.01%	83.20%	86.44%	86.72%	87.72%	89.73%	91.36%	92.54%	93.44%	94.14%	94.71%
exploitRatio	35.22%	76.29%	78.05%	78.68%	79.01%	79.20%	79.33%	79.43%	79.50%	79.56%	79.60%
exploitProg.	1.41	2.29	2.39	2.42	2.44	2.45	2.46	2.47	2.47	2.47	2.48
exploitSel.P.	35.22%	61.85%	59.09%	53.11%	47.13%	42.76%	39.84%	37.70%	36.12%	34.86%	33.87%
nRRatio.	78.29%	33.42%	20.30%	14.16%	10.75%	8.64%	7.23%	6.21%	5.44%	4.84%	4.37%
Entropy\gen	1	10	20	30	40	50	60	70	80	90	100
Best (83)	28.62	2.26	2.21	2.21	1.92	1.62	1.29	1.23	1.08	1.08	1.08
exploreRatio	64.65%	24.00%	22.10%	21.41%	21.07%	20.87%	20.73%	20.63%	20.56%	20.50%	20.45%
Type(m)	5.64%	11.09%	11.88%	11.31%	11.19%	11.76%	12.78%	13.61%	14.29%	14.66%	14.76%
exploreSel.P.	89.20%	83.32%	87.32%	86.63%	87.83%	89.69%	91.28%	92.46%	93.34%	94.05%	94.63%
exploitRatio	35.35%	76.00%	77.90%	78.59%	78.93%	79.13%	79.27%	79.37%	79.44%	79.50%	79.55%
exploitProg.	1.41	2.27	2.37	2.41	2.43	2.45	2.45	2.46	2.46	2.47	2.47
exploitSel.P.	35.35%	61.47%	59.78%	53.15%	47.43%	43.98%	41.95%	40.10%	38.47%	37.06%	35.87%
nRRatio.	77.85%	34.11%	21.67%	15.25%	11.76%	9.99%	9.06%	8.23%	7.51%	6.86%	6.24%
EE\gen	1	10	20	30	40	50	60	70	80	90	100
Best (66)	28.44	2.99	2.59	2.48	2.26	1.85	1.29	1.10	1.10	1.09	1.09
exploreRatio	65.49%	23.77%	21.98%	21.34%	21.02%	20.83%	20.71%	20.62%	20.55%	20.48%	20.43%
Type(m)	5.56%	11.61%	13.33%	12.76%	12.66%	13.11%	14.01%	16.23%	16.97%	17.16%	17.19%
exploreSel.P.	88.81%	83.04%	86.85%	86.42%	87.94%	89.74%	91.31%	92.46%	93.36%	94.07%	94.64%
exploitRatio	34.51%	76.23%	78.02%	78.66%	78.98%	79.17%	79.29%	79.38%	79.45%	79.52%	79.57%
exploitProg.	1.41	2.33	2.41	2.43	2.45	2.46	2.46	2.47	2.47	2.47	2.48
exploitSel.P.	35.41%	62.03%	59.28%	52.51%	47.51%	44.05%	41.66%	39.71%	37.89%	36.33%	35.03%
nRRatio.	78.22%	33.74%	21.56%	15.39%	12.16%	10.34%	9.38%	8.68%	7.86%	7.07%	6.38%

tion operator plays more exploration role (higher *exploreType(m)* value), which results in about 2 to 4 longer time (i.e., *exploreGap*) to explore new regions. Additionally, lower *exploitSelectionPressure* indicates that exploitation is not as efficient as other approaches. For the 1/5 success rule, its convergence rate is worse than the Fogarty approach, even though all but *exploreType(m)*, *exploitSelectionPressure*, and *nRRatio* are relatively close. A reasonable inference may be that relatively active mutation operator defers the convergence of 1/5 success rule approach. Lastly, the only difference between entropy-driven and EE-driven approaches is *exploitSelectionPressure* and *nRRatio*. Namely, the reason why the entropy-driven approach converges faster than the EE-driven approach might be higher *nRRatio* and higher *exploitSelectionPressure*. We found that for all approaches *exploitSelectionPressure* and *nRRatio*

have relatively reverse-proportional relationship to best fitness, which also indicates the importance of balance between exploration and exploitation.

Table 3 shows the experimental results of the Schaffer, Fogarty, 1/5 success rule, entropy-driven [7] and EE-driven approaches for Shekel's Foxholes Function (f_{14}). As mentioned in [13], an important performance indicator for multimodal functions is whether an EA can find its optimum. In Table 3, the Schaffer approach cannot retain *exploreRatio* until the end of evolution process. Hence, its optimization result is among the worst, except that it is better than the 1/5 success rule. Additionally, *exploitSelectionPressure* increases drastically, which implies that exploitation trees' width is too wide and very limited exploration is involved. Such characteristics also result in lower non-revisited ratio. For the Fogarty approach, an interesting observation can be found in *exploreType(m)* – everything but this measure is different from the entropy-driven and EE-driven approaches. It means that by utilizing different combinations of operators and their parameters, it is possible to generate similar results. As for the 1/5 success rule, almost every set of data is close to the entropy-driven and EE-driven approaches. However, its fitness is worse than the two approaches. A primary reason is that the 1/5 success rule may get into premature convergence sometimes. Our experimental results showed that about 10% of the experiments have such a problem. All EE measures will be also influenced in different magnitudes.

4. Conclusions

With more refined exploration and exploitation measures [2], it may be easier for EA researchers or practitioners to further analyze the performance of an EA and then conclude with sound rationales. Our experimental results show that f_2 requests less exploration but more exploitation after passing 250 generations. Exceeding exploration power on f_2 between 500 and 1000 generations may delay the convergence of the evolution process. Additionally, the ratios of mutation and crossover operators performing on exploration and exploitation may also influence the fitness. For f_{14}, our results show that maintaining exploration power within a suitable range (problem-specific that can be observed through EE measures) is of most importance to discover optimization. Without such appropriate exploration power, either worse fitness or premature convergence may occur.

A main objective of this paper is to demonstrate the competence and usefulness of using EE measures as analysis "tools" and feedback to

perform parameter control. To the best of our knowledge this is the first adaptive parameter control approach based on EE measures. Our future plan is to further investigate all benchmark functions from [13] to validate the presented hypothesis.

References

[1] T. Bäck and H.-P. Schwefel. Evolution Strategies I: Variants and Their Computational Implementation. *Genetic Algorithms in Engineering and Computer Science* John Wiley & Sons, 1995.

[2] M. Črepinšek, M. Mernik, and S.-H. Liu. Analysis of exploration and exploitation in evolutionary algorithms by ancestry trees. *International Journal of Innovative Computing and Applications*, 3(1):11–19, 2011.

[3] M. Črepinšek, S.-H. Liu, and M. Mernik. Exploration and exploitation in evolutionary algorithms: A survey. *ACM Comput. Surv.*, in press.

[4] Á. E. Eiben, R. Hinterding, and Z. Michalewicz. Parameter control in evolutionary algorithms. *IEEE T. Evolut. Comput.*, 3(2):124–141, 1999.

[5] Á. E. Eiben and C. Schippers. On evolutionary exploration and exploitation. *Fundame. Inform.*, 35(1-4):35–50, 1998.

[6] T. C. Fogarty. Varying the probability of mutation in the genetic algorithm. In *Proc. 3rd International Conference on Genetic Algorithms*, pages 104–109, 1989.

[7] S.-H. Liu, M. Mernik, and B. R. Bryant. To explore or to exploit: An entropy-driven approach for evolutionary algorithms. *International Journal of Knowledge-Based and Intelligent Engineering Systems*,13(3-4):185–206, 2009.

[8] S.-H. Liu, M. Črepinšek, and M. Mernik. Analysis of VEGA and SPEA2 using exploration and exploitation measures. In *Proc. 5th International Conference on Bioinspired Optimization Methods and their Applications*, pages 97–108, 2012.

[9] S.-H. Liu, M. Mernik, M. Zubair, M. Črepinšek, and B. R. Bryant. PPCea: A Domain-Specific Language for Programmable Parameter Control in Evolutionary Algorithms. In *Evolutionary Algorithms*, pages 177–200, Tech Open Access Publisher, 2011.

[10] M. Mernik, J. Heering, and A. M. Sloane. When and how to develop domain-specific languages. *ACM Comput. Surv.*, 37(4):316–344, 2005.

[11] J. D. Schaffer. Multiple objective optimization with vector evaluated genetic algorithms. In *Proc. 1st International Conference on Genetic Algorithms*, pages 93–100, 1985.

[12] J. D. Schaffer, R. A. Caruana, L. J. Eshelman, and R. Das. A study of control parameters affecting online performance of genetic algorithms for function optimization. In *Proc. 3rd International Conference on Genetic Algorithms*, pages 51–60, 1989.

[13] X. Yao, Y. Liu, and G. Lin. Evolutionary programming made faster. *IEEE T. Evolut. Comput.*, 3(2):82–102, 1999.

[14] E. Zitzler, M. Laumanns, and L. Thiele. SPEA2: Improving the Strength Pareto Evolutionary Algorithm for Multiobjective Optimization. In *Evolutionary Methods for Design: Optimisation and Control*, pages 95–100, 2002.

EXPLORING THE PARAMETER SPACE OF A SEARCH ALGORITHM

Katerina Tashkova
NELA Development Center, Železniki, Slovenia
katerina.taskova@ijs.si

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

Abstract This paper addresses the problem of tuning the performance of a metaheuristic search algorithm with respect to its parameters. The operational behavior of typical metaheuristic search algorithms is determined by a set of control parameters, which have to be fine-tuned in order to obtain a best performance for a given problem. The principle challenge here is how to provide meaningful settings for an algorithm, obtained as result of better insight in its behavior. In this context, we discuss the idea of learning a model of an algorithm behavior by data mining analysis of parameter tuning results. The study was conducted using the Differential Ant-Stigmergy Algorithm as an example metaheuristic search algorithm.

Keywords: Data mining, Differential ant-stigmergy algorithm, Low-discrepancy sequences, Metaheuristic optimization, Parameter tuning.

1. Introduction

The research interest for metaheuristic search algorithms has been significantly growing in the last 20 years as a result of their efficiency and effectiveness to solve large and complex problems across different domains [2]. The state-of-the-art nature-inspired metaheuristic algorithms for high-dimensional continuous optimization include also algorithms inspired from the collective behavior of social organisms [13].

One such algorithm, which we will address in this paper, is the Differential Ant-Stigmergy Algorithm (DASA) initially proposed by Korošec [6], and further improved in [7]. DASA is inspired by the efficient self-

organizing behavior of ant colonies emerging from a pheromone-mediated communication, known as stigmergy [3]. Naturally, DASA can be classified within the Ant Colony Optimization (ACO) framework. However, the use of a continuous pheromone model in the form of Cauchy probability distribution with representation of the search space in the form of the so-called differential graph makes DASA dissimilar from the original ACO paradigm. The rationale behind DASA is in memorizing (via the pheromone model updates) the "move" in the search space that improves the current best solution and using it in further search.

As it is the case with most of the metaheuristic algorithms, the operational behavior of DASA is determined by a set of control parameters. In practice, these parameters have to be fine-tuned in order to obtain best performance of the algorithm. It can be quite inconvenient for the users as: (i) they usually do not have insight into the behavior of the algorithm, and (ii) even if a default setting exists, it may not be adequate for a specific instance or type of a problem. Moreover, parameter tuning is computationally expensive task.

The principle challenge here is how to provide meaningful (default) settings for DASA, obtained as result of better insight into the algorithm's behavior. Furthermore, can we find optimal regions in DASA parameter space by analyzing the patterns in the algorithm's behavior with respect to the problem characteristics? Related to this, we discuss the preliminary findings based on data mining analysis of parameter tuning results. More precisely, the parameter tuning task is approached by two-step procedure that combines a kind of experimental design with data mining analysis.

We use Sobol' sequences [9] for even sampling of the algorithm parameter space to generate a large and diverse set of parameter settings. These are used as input to DASA to be tuned on a given function optimization problem. The performance of DASA on the given function optimization problem, in terms of function error, is captured at different time points for all sampled parameter settings. The data collected in the first step, DASA performance with corresponding parameter settings, is subject for intelligent data analysis, i.e., multi-target regression with Predictive Clustering Trees [1].

Parameter sampling combined with regression has been already used by Stoean et al. [10] for tuning metaheuristics: Latin hypercubes parameter sampling is combined with single-target regression with Support Vector Machines. Our approach modifies the former by replacing the Latin hypercube sampling by Sobol' sequences, as the former is best suited in the case when a single parameter dominates the algorithm's performance, while it should be used with care if there are interactions

among the sampled parameters [8]. Moreover, we define the regression task as multi-target regression, taking into account more the one target (in this case the function error at few time points) with the goal to find parameter settings for the given algorithm that will not only solve the problem but will also solve the optimization problem fastest.

The reminder of this paper is structured as follows. Section 2 introduces the differential ant-stigmergy algorithm. Then, Section 3 addresses the parameter tuning task and Section 4 presents the experimental evaluation with the results. After that, Section 5 discusses the idea of post-hoc analysis of parameter tuning by data mining. Finally, Section 6 summarizes this study and outlines possible directions for further work.

2. The Differential Ant-Stigmergy Algorithm

The version of DASA used in our experimental evaluation is described in details by Korošec et al. [7]. In principle, DASA relies on two distinctive characteristics, differential graph and continuous pheromone model. Here, we will briefly discuss these two characteristics and outline the main loop of the DASA search process.

First, DASA transforms the D-dimensional optimization problem into a graph-search problem. The corresponding differential graph is a directed acyclic graph obtained by fine discretization of the continuous parameters' offsets. The graph has D layers with vertices, where each layer corresponds to a single parameter. Each vertex corresponds to a parameter offset value that defines a change from the current parameter value to the parameter value in the next search iteration. Furthermore, each vertex in a given layer is connected with all vertices in the next layer. The set of possible vertices for each parameter depends on the parameter's range, the *discretization base*, b, and the maximal computer arithmetics precision, which defines the minimal possible offset value. Ants use these parameters' offsets to navigate trough the search space. At each search iteration, a single ant positioned at a certain layer moves to a specific vertex in the next layer, according to the amount of pheromone deposited in the graph vertices belonging to this layer.

Second, DASA performs pheromone-mediated search that involves best-solution-dependent pheromone distribution. The amount of pheromone is distributed over the vertices according to the Cauchy Probability Density Function (CPDF) [8]. DASA maintains a separate CPDF for each parameter. Initially, all CPDFs are identically defined by a location offset set to zero and a scaling factor set to one. As the search process progresses, the shape of the CPDFs changes: CPDFs shrink and

stretch as the scaling factor decreases and increases, respectively, while the location offsets move towards the offsets associated with the better solutions. The search strategy is guided by three user-defined real positive factors: the *global scale increase factor*, s_+, the *global scale decrease factor*, s_-, and the *pheromone evaporation factor*, ρ. In general, these three factors define the balance between exploration and exploitation in the search space. They are used to calculate the values of the scaling factor and consequently influence the dispersion of the pheromone and the moves of the ants.

Finally, the main loop of DASA consists of an iterative improvement of a temporary-best solution, performed by searching appropriates paths in the differential graph. The search is carried out by m ants, all of which move simultaneously from a starting vertex to the ending vertex at the last level, resulting in m constructed paths. Based on the found paths, DASA generates and evaluates m new candidate solutions. The best among the m evaluated solutions is preserved and compared to the temporary-best solution. If it is better than the temporary-best solution, the latter is replaced, while the pheromone amount is redistributed along the path corresponding to the path of the preserved solution and the scale factor is accordingly modified to improve the convergence. If there is no improvement over the temporary-best solution, then the pheromone distributions stay centered along the path corresponding to the temporary-best solution, while their shape shrinks in order to enhance the exploitation of the search space. If for some fixed number of tries all the ants only find paths composed of zero-valued offsets, the search process is restarted by randomly selecting a new temporary-best solution and re-initializing the pheromone distributions.

3. Parameter Tuning

To obtain the best possible performance on a given problem, one should consider a task specific tuning of the parameter setting for the optimization algorithm used. Determining the optimal parameters is an optimization task in itself, which is extremely computationally expensive. There are two common approaches for choosing parameters values: parameter tuning and parameter control. The first approach selects the parameter settings before running the optimization algorithm (and they remain fixed while performing the optimization). The second approach optimizes the algorithm's parameters along with the problem's parameters. Here, we will focus on the first approach, parameter tuning.

A detailed discussion and survey of parameter tuning methods is given by Eiben and Smit [4]. According to this survey, one way to approach

parameter tuning is by *sampling* methods. Sampling methods reduce the search effort by decreasing the number of investigated parameter settings as compared to the full factorial design: the basic full factorial design investigates 2^k parameter settings, subject to k parameters, each of which have 2 possible values; in the more general case, parameters can have arbitrary number of values; moreover, an increase in the number of investigated parameters means an exponential increase in the number of parameter settings to be tested. Two widely used sampling methods are Latin-squares [8] and Taguchi orthogonal arrays [11]. However, these are not the most robust sampling techniques, e.g., Latin-squares or Latin hypercube sampling is good in the case where one of the parameters dominates the algorithm's performance, while it should be used with care if there are interactions among the parameters.

Ultimately, we would like to find a sampling schema that will be able to detect the interactions among the parameters, will be independent from user-specified information regarding the particular parameter values to be considered (typical for factorial design), and will deliver small but representative sample of the parameter search space. The first two requirement are satisfied by the pure random sampling, but the last is not, as random sampling does not guarantee that the sampled values are evenly spread across the entire domain. The so-called low-discrepancy sequences were specially designed to fulfill all three requirements. Therefore, Sobol' sequences, a representative variation of low-discrepancy sequences introduced by Sobol' [9], was considered for sampling the parameter space of DASA in this study.

Sobol' sequences, sampled from a D-dimensional unit search space, are quasi-random sequences of D-tuples that are more uniformly distributed than uncorrelated random sequences of D-tuples. These sequences are neither random nor pseudo-random: they are cleverly generated not to be serially uncorrelated by taking into account which tuples in the search space have already been sampled. For a detailed explanation and overview of the schemas for generating Sobol' sequences, we refer to Press et al. [8]. The particular implementation of Sobol' sampling used in our analysis is based on the Gray code order [5].

4. Experimental Evaluation

Since data mining methods can only discover patterns actually present in the data, the dataset subject to analysis must be large enough and informative enough to contain these patterns, i.e., to describe different types of algorithm's behavior. For this reason, we decided to use a simple

test function, which matched this requirement and was used for building an example case model.

Therefore, the performance of DASA was evaluated on the Sphere function:

$$f(\vec{x}) = \|\vec{z}\|^2 + f(\vec{x}_{\text{opt}}),$$

where $\vec{z} = \vec{x} - \vec{x}_{\text{opt}}$ and $\vec{x}_{\text{opt}}$ is optimal solution vector, such that $f(\vec{x}_{\text{opt}})$ is minimal. Function $f(x)$ is defined over D-dimensional real-valued search space x and is scalable with the dimension D. It has no specific value of its optimal solution (it is randomly shifted in x-space) and has an artificially chosen optimal value (it is randomly shifted in f-space). In this study, we considered the Sphere function with respect to two dimensions, $D = 20$ and $D = 40$.

The performance of DASA is dependent on the values of five parameters: three real-valued parameters that directly influence the search heuristic (s_+, s_-, and ρ) and two integer-valued parameters (m and b). Therefore, we considered all of them for tuning DASA performance on the Sphere function for both search space dimensions, $D = 20$ and $D = 40$. Using the Gray-code-based Sobol' generator we generated 5000 parameter settings (5-tuples). Note that the Sobol' sampling generates numbers on the unit interval: in order to obtain the true parameter settings, we mapped these values on the predefined search range of parameter values. The latter for each of the five tuned parameters was defined as follows: $4 \leq m \leq 200$, $0 \leq \rho \leq 1$, $0 \leq s_+ \leq 1$, $0 \leq s_- \leq \rho$, and $2 \leq b \leq 100$. Moreover, the mapped values for the integer-valued parameters m and b were rounded to the closest integer value. Finally, due to implementation reasons, the upper bound of the global scale decrease factor s_- was actually limited by the value of the evaporation factor ρ.

In the next step, the performance of the Sobol' sampled parameter settings were tested on the Sphere benchmark function. Due to the stochastic nature of DASA, every parameter setting was used in a multiple-run experimental evaluation. Each run included $25,000 \times D$ function evaluations (FEs). The number of runs was set to 15.

The results gathered by the parameter tuning process are most often subjected to ordinal data analysis, which includes ranking of the different sampled parameter sets according to some calculated statistics, e.g., best or mean performance of the algorithm in some predefined number of runs [12]. In this case, performance of the algorithm is expressed in terms of the function error, i.e., the difference between the obtained and optimal function value. In order to find a setting that will be satisfactory in terms of convergence speed, we captured the error values at four different time

points, corresponding to $25 \times D$, $250 \times D$, $2{,}500 \times D$, and $25{,}000 \times D$ FEs.

The optimal performing parameter setting was chosen based on the median best performance over all runs aggregated over all time points for a given dimension ($D = 20$ and $D = 40$). A common approach is to use the mean performance, but we took the median in order to avoid the problems that the mean has when observing large variance in the function values across the runs. More precisely, given a function, an individual rank is assigned to every setting (out of 5000) for the four time points. A single final rank is calculated by ranking the sum of the four individual rankings assigned to the parameter settings. The best-ranked parameter setting for a given dimension defines instance of DASA referred to as DASA*.

The results of DASA tuning subject to ordinal data analysis are presented in Tables 1 and 2. Table 1 reports the tuned parameter settings for both DASA* instances. In addition, the default parameter setting for DASA from [7] is given as a reference for comparison. The corresponding instance is referred to as DASA$^\circ$.

Table 1. Parameter settings for DASA$^\circ$ and DASA*.

Algorithm	Dimension	m	ρ	s_+	s_-	b
DASA$^\circ$	20, 40	10	0.2	0.02	0.01	10
DASA*	20	5	0.324	0.201	0.289	6
	40	7	0.388	0.136	0.344	8

Results in Tables 2 represent the median values of the function errors, at the four time points, obtained by DASA* and DASA$^\circ$ instances for both dimensions. Note, that error value below 10^{-8} was treated as zero. The tables clearly show that DASA* is better than DASA$^\circ$.

Table 2. Median values of the function errors for the Sphere function.

Algorithm	DASA$^\circ$		DASA*	
FEs	$D = 20$	$D = 40$	$D = 20$	$D = 40$
$25 \times D$	16.7	18.3	2.53	9.31
$250 \times D$	0.0003	0.0021	0	0
$2{,}500 \times D$	0	0	0	0
$25{,}000 \times D$	0	0	0	0

5. Data-Mining Analysis

Parameter tuning of an algorithm leads to a better performance, however it is a computationally expensive and problem-dependent task. Considering this, the idea is to extend the simple tuning that delivers a single parameter set and analyze the gathered data in an intelligent way. The intelligent analysis can extract patterns (regularities) in the explored parameter space that define a specific behavior of DASA. To this end, data mining methods for automated discovery of patterns in data can be used. As data mining methods can only discover patterns that are present in the data, the dataset subject to analysis must be large enough and informative enough to contain these patterns, i.e., to describe different types of algorithm's behavior. Related to this, we considered a data mining approach on a representative example, i.e., error model of the Sphere function.

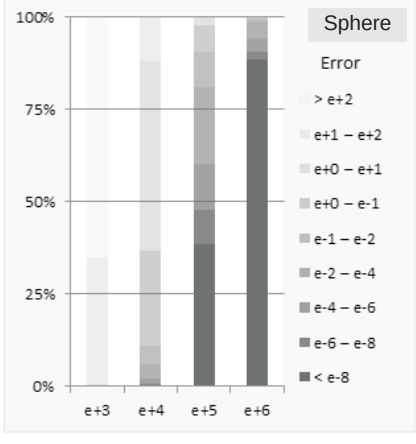


Figure 1. Median error distributions for the Sphere function in the case of $D = 40$.

To begin with, consider the graph in Fig. 1 that visualize the Sphere function error distribution. The graph depicts the distribution of the median error values obtained by 5000 parameter settings at four different points for $D = 40$. As we are more concerned with the practical significance between a large and a small error value than the statistical significant difference between two actual error values, the error values are discretized into nine intervals, each of which is represented with a color chosen according to the error magnitude between black (error below 10^{-8}) and white (error above 10^{2}). The graph clearly shows that the sampled settings determine different DASA performance. As evident, there is a big cluster of parameter settings that solve this function

to the aimed accuracy (error below 10^{-8}) in the given time budget (10^6 FEs). Moreover, subset of this cluster solves the function for an order of magnitude less FEs. Our aim, therefore, is to find a (common) description of this cluster, in terms of DASA parameter relations, that represents a good behavior of DASA (as well as what parameter relations lead to a bad DASA performance).

For this purpose, we formulated the problem as a predictive modeling task using decision trees to model the function error values in terms of the values of DASA parameters. Since the function error variables are continuous, the task at hand is a regression task. Furthermore, as our goal is to model the behavior of DASA at all time points simultaneously, the problem at hand is then a multi-target regression. To this end, we used Predictive Clustering Trees (PCTs) which are implemented in Clus system [1]. In this case, the median error at the four time points define the four target attributes considered for modeling (with the PCT) in dependence from the descriptive attributes, i.e., the function dimension and the five DASA parameters. The resulting dataset is composed of $10,002$ rows described with the 5001 parameter settings (including the default setting) of DASA applied to the two different dimensions, and 10 columns corresponding to the 10 attributes.

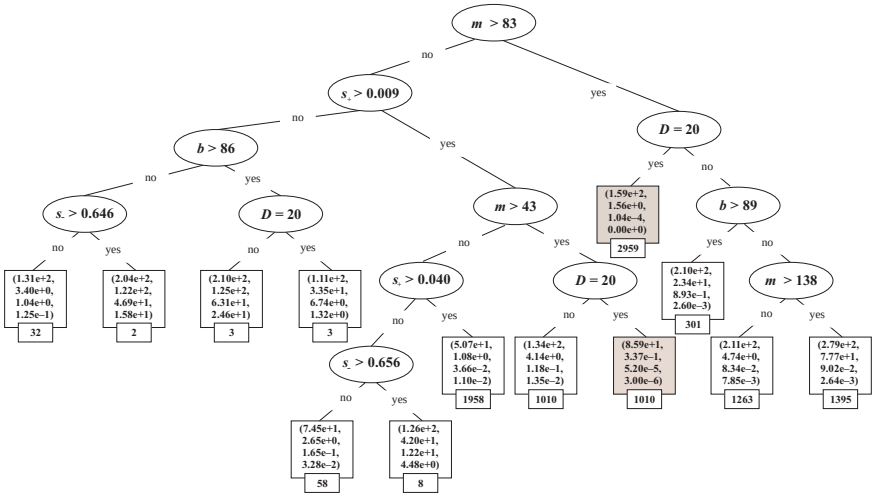


Figure 2. Predictive clustering tree representation of the error model for the Sphere function.

Figure 2 presents a PCT model for the Sphere function error. Each internal node of the tree contains a test on a descriptive attribute, while the leaves store the model predictions for the function error, i.e., a tu-

ple of four values. The predictions are calculated as the mean values of the corresponding error values for the data instances belonging to the particular leaf (represented by a box). In fact, each leaf identifies one cluster of data instances (the size of the cluster is the value in the small box). The predictive performance of the model was assessed with 10-cross-validation. Note that this particular model was learned on the complete dataset subject to constraints on the maximal tree size of 25 nodes. We did this because the original model contained 1643 nodes (of which 822 leaves) and despite its better predictive performance, both training and testing, it was not comprehensible; aiming for an explanatory model, small and comprehensible, we considered the smaller tree obtained with the limitation of the size. The predictive performance of both models in terms of Root Relative Mean Squared Error (RRMSE) and Pearson Correlation Coefficient (PCC) are given in Table 3. Note that RRMSE represents the relative error with respect to the mean predictor performance, while PCC represents the linear correlation between the data and the model predictions. Good models have RRMSE values closer to 0 and PCC closer to 1.

Table 3. Model performance with respect to RRMSE and PCC.

Performance	PCT nodes	Measure	FEs				Mean
			2.5e+2 × D	2.5e+3 × D	2.5e+4 × D	2.5e+5 × D	
Training	1643	RRMSE	0.166	0.373	0.487	0.472	0.396
		PCC	0.986	0.928	0.873	0.882	0.843
	25	RRMSE	0.408	0.679	0.562	0.546	0.557
		PCC	0.913	0.734	0.827	0.837	0.689
Testing	1643	RRMSE	0.219	0.638	1.019	1.053	0.806
		PCC	0.976	0.777	0.304	0.234	0.426
	25	RRMSE	0.450	0.817	1.039	1.055	0.875
		PCC	0.893	0.584	0.246	0.218	0.312

The model in Fig. 2 outlines 13 clusters of data instances, of which two (depicted with light-gray boxes) represent a good DASA performance. According to this model, the number of ants, m, is the most important DASA parameter for its performance on the Sphere function. More precisely, if $m > 83$, independent of the values of the other parameters, DASA solves the 20-dimensional functional problem for the given time budget. Moreover, if $m \leq 83$ another DASA parameters become important as well. For example, if $43 < m \leq 83$ and $s_+ > 0.009$ and $D = 20$ then DASA solves the function with error $3 \cdot 10^{-6}$, while the pattern $m \leq 43$ and $s_+ \leq 0.040$ and $s_- > 0.656$ describes a poor DASA performance regardless of the function dimension. An interesting fact is

that, the evaporation factor is not essential for DASA performance on the Sphere function. Moreover, the model also shows that is more difficult to describe the behavior of DASA for the 40-dimensional function problem than the 20-dimensional one.

Finally, note that the training performance (learned on the complete dataset) of the model in terms of the error and the correlation coefficient is best for the first target, while it gets worse with respect to the other three targets (see Table 3). This is especially significant if we take into account the testing performance of the model estimated with 10-cross-validation. However, the training performance is acceptable in our case, as we are interested in understanding the behavior of DASA and not aiming to obtain a model for prediction.

6. Conclusions

The principle challenge of metaheuristic design is providing a default algorithm configuration, in terms of parameter setting, that will perform reasonably well in general (problem) case. However, while it is a good initial choice, the default algorithm configuration may result in low quality solutions on a specific optimization problem. In practise, the algorithms parameters have to be fine-tuned in order to obtain best algorithm's performance for the problem at hand, leading to the computational expensive task of parameter tuning. So, if the tuning task is unavoidable, the question is: can we use the results from the parameter tuning to extract some knowledge about the algorithm's behavior?

Related to this, the study focused on the problem of tuning the performance of the Differential Ant-Stigmergy Algorithm (DASA) with respect to its parameters. As it is the case with most of the metaheuristic algorithms, the operational behavior of DASA is determined by a set (five) of control parameters. The existing default setting of DASA parameters [7] is obtained by experimentation with both real and benchmark optimization problems, but not as a result of some systematic evaluation. Furthermore, there is no deeper understanding of the impact of a particular parameter or parameters relations on the performance of DASA. In this context, we performed an systematic evaluation of DASA performance obtained by solving the Sphere function optimization problem with 5000 Sobol' sampled DASA parameter settings regarding two dimensions, 20 and 40. Furthermore, we discussed the idea of learning a model of DASA behavior by data mining analysis of the parameter tuning results. In this context, we formulated the problem as multi-target regression and applied predictive clustering trees for learning a model of DASA behavior with respect to the function error performance. The ob-

tained model revealed that the parameter denoting number of ants is the most important parameter for DASA performance on the 20-dimensional function problem. On the other hand, the evaporation factor is not essential for DASA performance on the Sphere function.

Further work will focus on additional experimental evaluation and data mining analysis of data with respect to more complex functions problems. This idea can be further extended to building models of DASA behavior that will include the optimization problem characteristics (such as multimodality, separability, and ill-conditioning) as descriptive attributes as well. The latter can provide insights on how to configure DASA performance with respect to the type of the optimization problem. Moreover, these insights can serve as a valuable information for improvement of DASA design.

References

[1] H. Blockeel and J. Struyf. Efficient algorithms for decision tree cross-validation. *J. Mach. Learn. Res.*, 3:621–650 (2002).

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

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

[4] A. E. Eiben and S. K. Smit. Parameter tuning for configuring and analyzing evolutionary algorithms. *Swarm Evol. Comput.*, 1(1):19–31 (2011).

[5] S. Joe and F. Y. Kuo. Constructing Sobol sequences with better two-dimensional projections. *SIAM J. Sci. Comput.*, 30(5):2635–2654 (2008).

[6] P. Korošec. Stigmergy as an approach to metaheuristic optimization. Doctoral Dissertation, Jožef Stefan International Postgraduate School, Ljubljana, 2006.

[7] P. Korošec, J. Šilc and B. Filipič. The differential ant-stigmergy algorithm. *Inform. Sciences*, 192(1):82–97 (2012).

[8] W. H. Press, S. A. Teukolsky, W. T. Vetterling and B. P. Flannery. *Numerical Recipes, 2nd edition.* Cambridge University Press, 1992.

[9] I. M. Sobol'. Distribution of points in a cube and approximate evaluation of integrals. *USSR Comput. Maths. Math. Phys.*, 7(4):86–112 (1967).

[10] R. Stoean, T. Bartz-Beielstein, M. Preuss, and C. Stoean. A Support Vector Machine-Inspired Evolutionary Approach for Parameter Setting in Metaheuristics Research Center CIOP (Computational Intelligence, Optimization and Data Mining). CIOP Technical Report. Cologne University of Applied Science, Faculty of Computer Science and Engineering Science, 2009.

[11] G. Taguchi and T. Yokoyama. *Taguchi Methods: Design of Experiments.* ASI Press, 1993.

[12] E-G. Talbi. *Metaheuristics: From Design to Implementation.* John Wiley and Sons, 2009.

[13] X.-S. Yang. *Nature-Inspired Metaheuristic Algorithms.* Luniver Press, 2008.

A GRAPH-BASED EVOLUTIONARY ALGORITHM: CELL BASED GENETIC PROGRAMMING

Marko Corn, Gregor Černe
INEA d.o.o., Ljubljana, Slovenia
{marko.corn; gregor.cerne}@inea.si

Maja Atanasijević-Kunc
Faculty of electrical engineering, University of Ljubljana, Slovenia
maja.atanasijevic-kunc@fe.uni-lj.si

Abstract In this paper new evolutionary algorithm and graph based method called Cell Based Genetic Programming (CBGP) is presented. CBGP method uses graph based structures to evolve solutions of optimization problems. Main advantage of CBGP method is its ability to form feedback loops and sub-process. Creation of feedback loops enables construction of discrete differential equations and creations of sub-processes enables construction of sub programs. Method has been tested on modeling of hydrogen fuel cell which represents nonlinear optimization problem.

Keywords: Graph structure, Genetic programming, Feedback loops, Sub-processes.

1. Introduction

Evolutionary algorithms are optimization methods used for solving complex problems where other methods aren't so successful. They are so successful because in generally these methods don't make any assumptions about the search space in which they are seeking for a solution of the problem. This ability also makes them very useful on fields as diverse as engineering, art, biology, marketing, genetics, robotics, social sciences, politics, chemistry and others. Their usefulness also triggered great interests in research which leaded into development of huge amount of different methods and approaches. Evolutionary methods are currently in strong development because of their great potential. In this article a new method called Cell based Genetic Programming (CBGP) is in-

troduced. First the overview of evolutionary methods is done, to place CBGP method in the field of evolutionary algorithms. Overview is followed by the description of the method. Finally the method is tested on modeling of hydrogen fuel cell.

2. Overview of Evolutionary Algorithms

Evolutionary methods can be divided in two distinct groups: parametrical methods and structural methods [4]. Parametrical group represents methods like genetic algorithms [8], evolutionary strategies [1], differential evolution [12] and others [2]. This method searches for solution of the optimization problem in form of values of optimization parameters. Structural methods search solutions of optimization problem by building functions. Structural methods doesn't just sets the parameters of the function this methods can build their own functions. Further structural group can be divided into two groups based on their structure type: tree based and graph based methods. Methods based on three structures are genetic programming [9], grammatical programing [13] and others. Graph based methods can be further divided into two groups: methods that bases on cellular automata and methods that are modification of three based methods. Methods based on cellular automata are evolutionary programming [5], cellular genetic programming [6], genetic network programming [10] and others. Methods that are modification of tree based methods are parallel distributed genetic programming [11], multiple interactive outputs in a single tree [7] and others. CBGP method [3] belongs to structural graph based methods, but it is not based on cellular automata structure nor modified tree based structure.

3. Cell Based Genetic Programming

All evolutionary computation methods imitate natural evolution process. In natural evolution there are three basic elements environment, organism and reproduction of organisms. Analogue to the natural evolution, evolutionary computation methods have environment, entity and reproductions of entities.

3.1 Environment

Natural environment provides a habitat for organisms in which they live and mechanism of selection that determinate which organisms will reproduce. Replacement of natural environment with simulation environment will provide entities appropriate habitat for living and replacement for selection mechanism is appropriate fitness function.

3.2 Entity - Cell

Basic description. In CBGP method entity is named cell. Example off the cell is represented on Fig. 1 Cell consists of cell membrane,

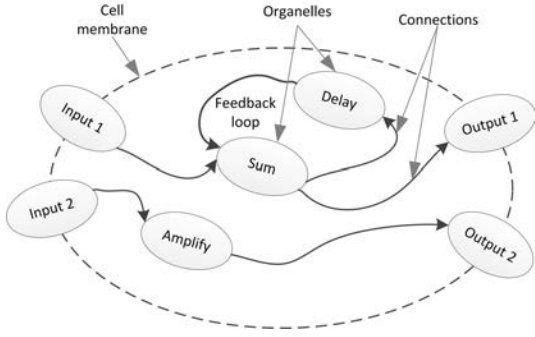


Figure 1. Cell example.

organelles and connections. Cell membrane is virtual boundary between environment and interior of the cell. Organelles are the smallest individual elements that are located inside off the cell and can translate their input data through some basic mathematical functions like arithmetical, logical and others into their output data. Organelles that are lying on cell membrane insure a flow of data from the environment to the interior of the cell and back to the environment. Figure 2 represents example of general organelle. Organelle can have multiple inputs $(u_1, \ldots, u_n)$ and single output (y). Organelles are divided in types based on their mapping function $(f(u_1, \ldots, u_n))$ that translate their input values to their corresponding output values. Connections $(C_{inp_1}, \ldots, C_{inp_m}, C_{out_1}, \ldots, C_{out_n})$ connect inputs and outputs of different organelles and provide a transfer of data through the entire cell. With different sets of organelles types a set

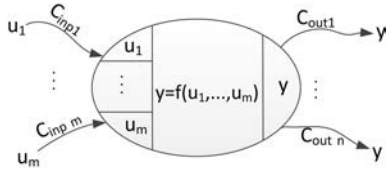


Figure 2. Generale organelle.

of functions that will assemble the solution can be chosen. This ability enables adaptation of the method to a different types of problems. Design off the cell enables implementation of feedback loops (Fig. 1) and formation of sub-processes which are the main advantaged accord-

ing to other methods. In order to understand how feedback loops and sub-process works the data flow through the cell must be explained.

Data flow through cell. CBGP method can be used in discrete simulation environments. Cell calculates output data from its input data and internal organelles every discrete time event of the simulation run. On Fig. 3 the data flow through cell for one discrete event is presented. First all input data from simulation environment is transferred to input

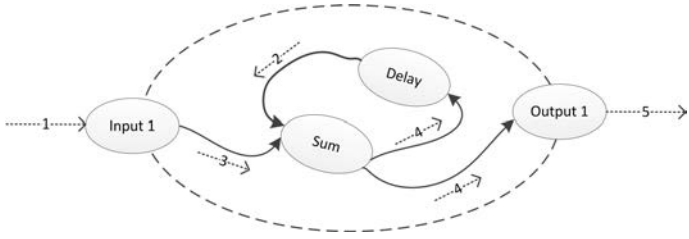


Figure 3. Example of data flow through cell.

organelles (doted arrow 1). Second all organelles which have initial state sends values of their initial state pass their output connections to other organelles (doted arrow 2). Organelles that have initial states represent functions like time delay or discrete integral. Thirdly all input organelles send their data pass their output connections to other organelles (doted arrow 3). Fourthly all internal organelles calculate their outputs through their mapping function and sends calculated output values pass their output connections to other organelles (doted arrow 4). Finally the output organelles transfer their input data to the simulation environment (doted arrow 5).

Feedback loop. By allowing organelles to randomly connect to each other a feedback loops can be created. Cell on Fig. 3 has feedback loop that forms discrete integral. In general feedback loops enables formation of discrete differential equations and with proper design of organelles even partial discrete differential equations.

Sub processes. Sub process is a process that runs inside of the cell and can be processed multiple times within one discrete simulation time event of the cell. Figure 4 shows one sub process inside of the cell. Sub processes can be formed with special type of organelle that is called Counter organelle. Counter organelle determines how many times other organelles that are connected to him will calculate their outputs in one discrete time event of a cell simulation. Organelles can also be partly

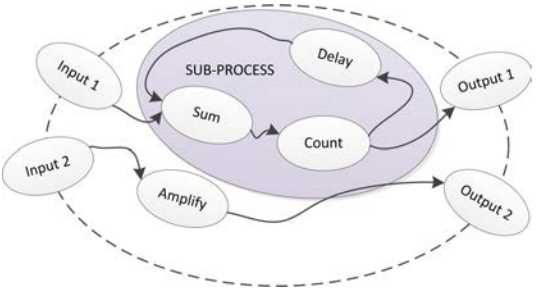


Figure 4. Example of sub-process formed inside of the cell.

connected to sub process and partly to other organelles in the cell. It is also possible to form sub processes inside of a sub processes. This option allows CBGP method to form complex dynamic mathematical functions.

3.3 Reproductions

Reproduction of cells in CGBP method imitates reproduction of organisms in nature where genetic material is transferred from parent to children. In general reproduction methods are divided into two groups: mutations and crossovers. Mutations are a transfer of genetic material from one parent to one child where part of genetic material is randomly changed. On the other hand crossovers are a transfer of genetic material from multiple (normally two) parents to one child were the genetic material of the child is a combination of the genetic material from both parents.

Mutations. CGBP method has tree type of mutations: Change of organelles mapping function, change of connections and change of number of organelles in the cell. First changing of organelle mapping function means that function parameters are randomly changed (this type of mutations are similar to reproductions methods used in genetic algorithms) or the entire function is randomly changed which leads to changing of organelle type. Second changing of organelles connections means that two randomly chosen connections switch their connection points (Fig. 5). Finally changing number of organelles means that number of organelles in the cell is reduced or enlarged with their connections deleted in case of reduction and randomly connected in case of enlargement.

Crossovers. CGBP method has one type of crossover reproduction. From the first parent a random number of connected organelles

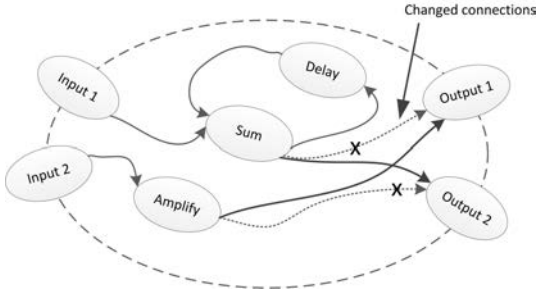


Figure 5. Mutation example of swapped connections.

are chosen and ripped out from the cell. All the connections that connect chosen organelles to the cell are deleted. From the second parent the opposite process is done. Randomly chosen organelles with their connections are deleted from the cell. Then the ripped bundle of organelles from first parent is inserted into space that has been left by the bundle of deleted organelles from second parent. The empty connections of ripped organelles are randomly connected to empty connections of organelles that stayed in second parent cell. Figure 6 represents example of crossover reproduction. Crossover reproduction begins with

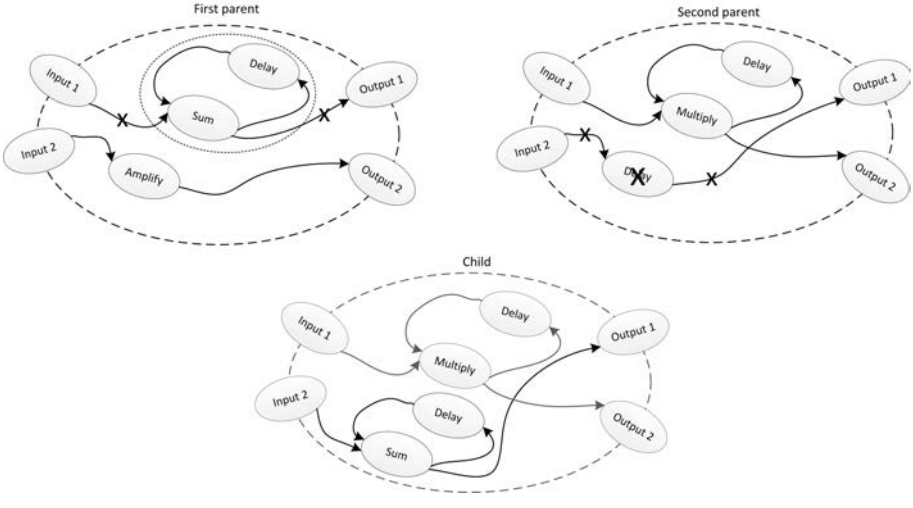


Figure 6. Example of cross reproduction.

extraction of Sum and delay organelles from the first parent. Then from the second parent the Delay organelle is deleted together with its connections. Further the child is formed by inserting a group of organelles

extracted from the first parent into place where the Delay organelle from the second parent has been deleted. Finally the connections are made to connect Sum organelle with Input1 organelle and Output1 organelle.

4. Simulations

Experimental testing of CGBP method has been done on hydrogen fuel cell system. Hydrogen fuel cells are one of the most promising technologies for production of electrical energy. They produce electrical energy, heat energy and water by chemical reactions between hydrogen and oxygen. They represent complex dynamical systems and will be used for experimental testing of CGBP method [4]. Experimental test of modeling of hydrogen fuel cell has been done.

4.1 Modeling of the Fuel Cell

Complete fuel cell system is very complex and consist of fuel cell stack, pumps, coolers and other peripheral devices that enables its proper working. This model represents only fuel cell stack in one working point (Fig. 7). Output voltage of the stack (U'_{st}) is dependent from current

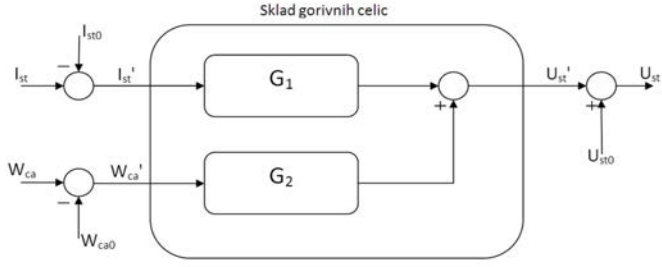


Figure 7. Model of hydrogen fuel cell stack.

working power that is set indirectly by input electrical current (I'_{st}) of the stack and mass flow of fuel into the cell (W'_{ca}). Because model of the stack is designed for certain working conditions the working points ($U_{st_0}, I_{st_0}, W_{ca_0}$) must be added to get final inputs and outputs of the model (U_{st}, I_{st}, W_{ca}). Equation (1) describes model of the fuel cell stack in working point condition.

$$U'_{st} = U_1 + U_2 = G_1(I'_{st}) + G_2(W'_{ca}) \tag{1}$$

G_1 and G_2 represent functions of the model that has to be calculated with CGBP method. Method that has been used to model stack is

called identification. Identification is process where output signals are measured in response to change of input signals on the real device. With these measurements the model is designed in way that to the same input change gives same output response. Figures 8 and 9 represents output response to step changes of input signals I'_{st} and W'_{ca}.

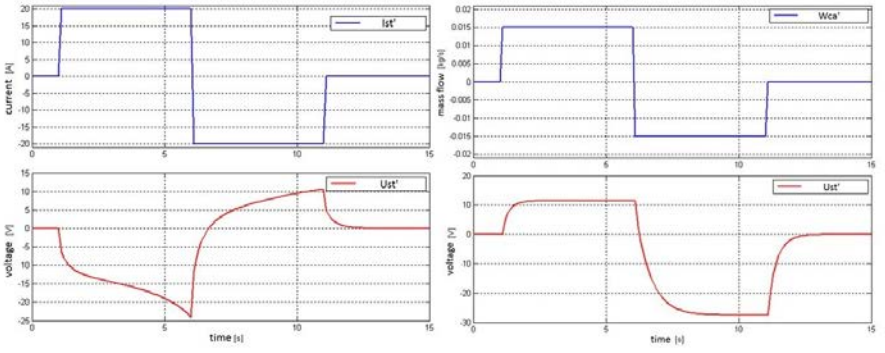


Figure 8. Output response to the change of input signal I'_{st} and W'_{ca}.

Model that has been generated with CGBP model is presented in Eqs. (2)–(5).

$$U'_{st}(k) = U_1(k) + U_2(k) \tag{2}$$

$$U_1(k) = 0.69(-0.15 \sum_{k=0}^{T_{sim}} \sum_{k=0}^{T_{sim}} \sum_{k=0}^{T_{sim}} [I'_{st}(k-1)] - 0.84 I'_{st}(k-1)) \tag{3}$$

$$U_2(k) = 94.28(7.85 W'_{ca}(k-2) + X(k)) \tag{4}$$

$$X(k) = 0.41 \sum_{k=0}^{T_{sim}} [W'_{ca}(k-19)] + X(k-17) + 15.71 W'_{ca}(k-5)^{0.0013} \tag{5}$$

Equations (3) and (4) describes functions G_1 and G_2. Equation (3) has to be written with special character X that represents internal state of the cell and indicates that feedback loop has been formed. Figure 9 represents comparison of measured responses of the stack and calculated model response to the same input change of signals.

For the design of this model using CBGP method 1000 generations were generated.

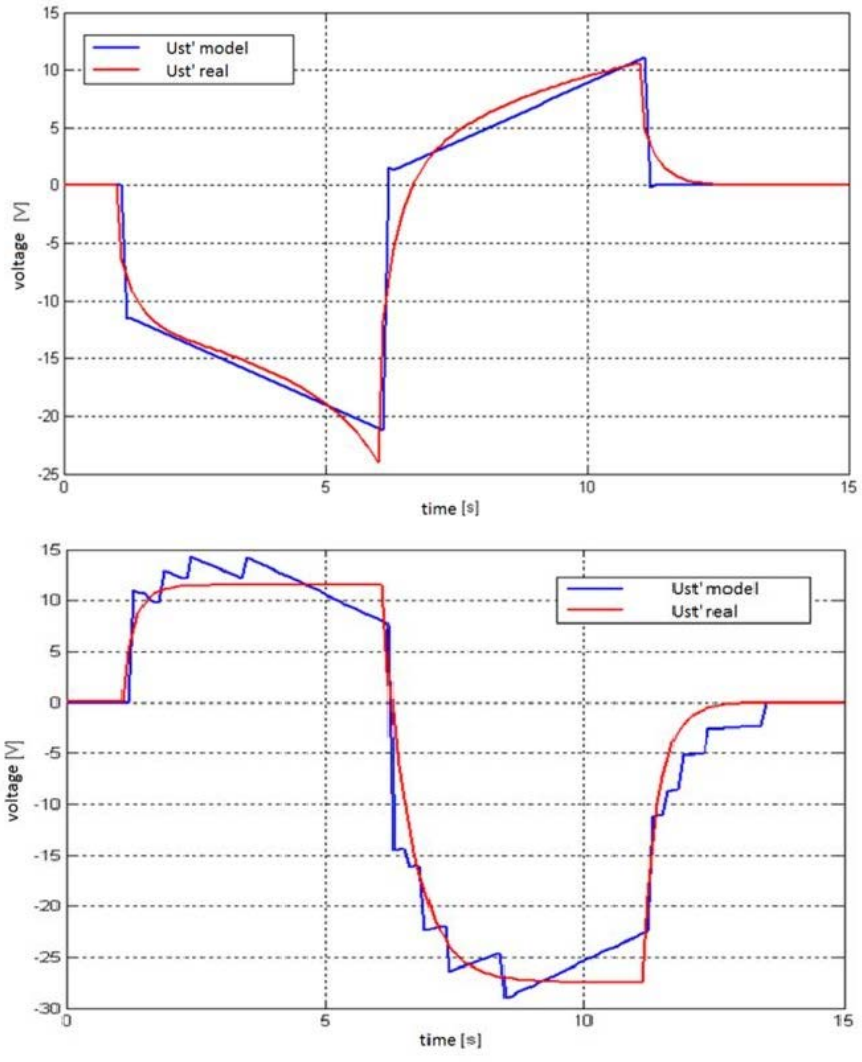


Figure 9. Comparison of stack measurements with model response.

5. Conclusions

Method gives promising results in way of graduate convergence to desired solution. In the solution we can see that a feedback loop has been formed. Solution didn't form any sub-processes. Reason for that is that the sub-processes formation has been disabled because this part of method is still in development. CGBP method has encounter problem with computational power but in future parallelization and better soft-

ware and hardware equipment will help. This method has also ability to include initial knowledge of the system in to the development of solution which can make significant contribution towards the optimum solution.

References

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

[2] J. Brownlee. Clever Algorithms: Nature-Inspired Programming Recipes. PhD Thesis, Swinburne University, Melbourne, Australia, 2011.

[3] M. Corn and M. Atanasijević-Kunc. *Evolucijsko računanje in njegova uporaba pri obravnavi dinamičnih sistemov.* Bachelor thesis, Faculty of electrical engineering, University of Ljubljana, Slovenia, 2010.

[4] M. Corn and M. Atanasijević-Kunc. Orodje za genetsko programiranje na osnovi celic (Cell based genetic programming toolbox). In *Proc. 20th ERK Conference*, pages 295–298, 2011.

[5] D. B. Fogel and L. J. Fogel. An introduction to evolutionary programming. *Lect. Notes in Comput. Sc.*, 1063:21–33, 1996.

[6] G. Folino, C. Pizzuti, and G. Spezzano. A cellular genetic programming approach to classification. In *Proc. Genetic and Evolutionary Computation Conference*, volume 2, pages 1015–1020, 1999.

[7] G. L. Edgar. Efficient graph-based genetic programming representation with multiple outputs. *International Journal of Automation and Computing*, 5(1):81–89, 2008.

[8] J. Holland. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence.* A Bradford Book, 1992.

[9] J. R. Koza. *Genetic Programming On the Programming of Computers by Means of Natural Selection*, 6th edition. MIT Press, 1992.

[10] S. Mabu, K. Hirasawa, and J. Hu A graph-based evolutionary algorithm: Genetic network programming (GNP) and its extension using reinforcement learning. *Evolut. Comput.*, 15(3):369–398, 2007.

[11] R. Poli. Evolution of graph-like programs with parallel distributed genetic programming. In *Proc. 7th International Conference on Genetic Algorithms*, pages 346–353, 1996.

[12] R. M. Storn and K. V. Price. Differential evolution – A simple and efficient heuristic for global optimization over continuous spaces. *J. Global Optim.*, 11(4):341–359, 1997.

[13] B. Wolfgang, P. Nordin, and R. E. Keller. *Genetic Programming: On the Automatic Evolution of Computer Programs ands Its Applications.* Morgan Kaufmann Press, 1998.

IMPACT OF NNA IMPLEMENTATION ON GA PERFORMANCE FOR THE TSP

Goran Martinović, Dražen Bajer
Faculty of Electrical Engineering, Josip Juraj Strossmayer University of Osijek, Croatia
{goran.martinovic; drazen.bajer}@etfos.hr

Abstract Genetic algorithms are a frequently used method for search and optimization problem solving. As such they have also been used to solve the traveling salesman problem. Since they are population-based, the initial population plays a very important role and affects the algorithm's convergence speed as well as the quality of the final solution. Commonly, the population is initialized with randomly generated solutions, but in this paper an adapted nearest neighbor algorithm is used for population initialization. Besides that, the algorithm is used during recombination to a lesser extent. Experimental analysis conducted on several instances of the symmetric traveling salesman problem showed that significantly better solution can be achieved with the presented method for the population initialization than with the commonly used random method. Also, the presented method is relatively easy for implementation.

Keywords: Genetic algorithm, Initial population, Nearest neighbor algorithm, Recombination, Traveling salesman problem.

1. Introduction

The traveling salesman problem (TSP) [1] is one of the best known and most studied combinatorial optimization problems. It has found applications in many areas of science and industry, like logistics, electronics, genetics, telecommunications and others. The TSP requires that the shortest possible tour visiting every city from a given set including the return to the departure city is found. Since it is a very well studied problem, many methods and algorithms have been proposed for its solving. The TSP is a NP-hard problem, therefore solving it to optimality is usually very demanding from the viewpoint of required time and necessary computational resources. Many applications of the problem do not require an optimal solution, but instead they require good quality solutions acquired in relatively short time. For that purpose approxi-

mate methods (heuristics), or more often metaheuristics [2] can be used. Although the aforementioned methods cannot guarantee an optimal solution will be found, as a rule they find solutions close to the optimal one in reasonable time. Examples of successful metaheuristics for the TSP are Tabu Search, Iterated Local Search, Ant Colony Optimization and Genetic Algorithms (GAs).

A genetic algorithm [2, 9, 13] is an optimization and search method based on genetics and the process of natural selection. Since GAs are inspired by the process of evolution, a population of solutions evolves throughout generations by applying selection and genetic operators. Solutions, called chromosomes, are evaluated by means of a fitness function that reflects the problem at hand. Selection removes bad solutions from the population, and enables better ones to be subject to recombination through the use of genetic operators which are represented by crossover and mutation. Crossover creates new solutions, i.e. offspring, by exchanging genetic material between two existing ones called parents. Mutation introduces random changes to a chromosome. Its purpose is mainly to recover lost genetic material.

In [5], authors presented a GA, in which a small part of the population is initialized with solutions constructed by the nearest neighbor algorithm (NNA), while the rest of that population is initialized with randomly generated solutions. On a few smaller TSP instances it was shown that in this way a greater convergence speed and better results could be achieved compared to a GA whose entire population is initialized with randomly generated solutions. A GA for the TSP where half of its population is initialized with solutions generated by an improved greedy algorithm and the other half with randomly generated solutions was presented in [12]. Analysis, conducted on one TSP instance showed that this algorithm converged faster and achieved better solutions compared to some other popular algorithms. A novel selective initialization approach for the population initialization of a GA is proposed in [10]. The presented approach uses a tournament selection procedure in which randomly generated solutions participate. The tournament selection is conducted a given number of times and the population is initialized with the tournament winners. Analysis conducted on the 0/1 knapsack problem showed that the presented method is superior to some other population initialization methods.

This paper shows a way of implementing the NNA in a GA for the TSP. The NNA is used to initialize the population with relatively good quality solutions. Also, the NNA is used during the recombination process to a lesser extent. The impact of NNA implementation on the performance of a GA is shown on several TSP instances.

A TSP definition is given in Section 2. Section 3 describes the NNA and its adaption for the purpose of population initialization. Section 4 deals with the implementation of the programming solution used to obtain the experimental results. Results of the conducted experimental analysis on the impact on the performance of a GA with an implemented NNA are presented in Section 5.

2. The Traveling Salesman Problem

Formally, the TSP may be described by a complete weighted graph $G(V, A)$, where $V = \{1, 2, \ldots, n\}$ is the set of vertices representing the cities, and $A = \{(i, j)|i, j \in V \text{ and } i \neq j\}$ is a set of arcs completely connecting those vertices. Every arc (i, j) is assigned a weight (length) $d_{i,j}$. The TSP requires the shortest Hamiltonian cycle in G is found. The length of the cycle is given by sum of the lengths of the arcs in the tour (cycle). Since the cycle length depends on the detour sequence the TSP asks a permutation (ordering), π, is found such that expression (1) is minimal.

$$\sum_{i=1}^{n-1} d_{\pi(i),\pi(i+1)} + d_{\pi(n),\pi(1)} \tag{1}$$

In case graph G is directed (i.e. for at least one arc $d_{i,j} \neq d_{j,i}$ holds), we refer to the asymmetric TSP (ATSP); otherwise we refer to the symmetric TSP (STSP). The Euclidean TSP is an example of the STSP where the vertices (cities) are given in $\mathbb{R}^2$ (more generally in $\mathbb{R}^d$). The Euclidean TSP is also considered in this paper.

3. The Nearest Neighbor Algorithm

The nearest neighbor algorithm [1, 4] is a simple tour construction procedure for the TSP. The algorithm constructs an ordering $\pi(1), \pi(2), ..., \pi(n)$ of the cities, i.e. a tour. Beginning from a given starting city it selects in every step the nearest yet-unvisited city (neighbor) in relation to the last visited one. After all cities are visited the starting city is added to close the cycle. The starting city is usually randomly selected. According to [4], the quality of solutions for the Euclidean TSP constructed by NNA is in the worst case $0.5(\lceil \log_2 n \rceil + 1)$ times greater than the quality of the optimal solution. The time complexity of the described algorithm for a problem with n cities is $\Theta(n^2)$.

3.1 Initializing the Population with the Nearest Neighbor Algorithm

The initial population of a GA is usually randomly generated [7, 9]. As a rule, it is desirable for the initial population to be as diverse as possible [9]. That way a more extensive exploration of the search space is enabled. Randomly generated solutions are usually of very low quality, therefore a GA needs a lot of time to converge toward a high quality or eventually optimal solution. A greater convergence speed should be achieved by initializing the population with good quality solutions.

The NNA constructs always the same solution, i.e. tour for the same starting city. For different starting cities different tours may be constructed, but there is also a possibility that for different starting cities the same or very similar tours will be constructed. For that reason, it cannot be guaranteed that the initial population will be diverse enough. If the initial population is not diverse enough only a small part of the search space is explored, thereby reducing the probability of finding the global optimum. To generate a more diverse population with the NNA, we adapted the algorithm, so that in every step one among the $0 < k < n$ nearest yet-unvisited neighbors is chosen as the next city to be visited (the algorithm is hereinafter referred to as NN-k). Each of the k nearest yet-unvisited neighbors of the current city has the same probability of being selected. After $n - k + a$ cities are already visited, the number of nearest yet-unvisited neighbors of the current city is $k - a$, where $1 \leq a \leq k$, and one among those remaining $k - a$ neighbors is selected as the next city. Considering the aforementioned, the algorithm becomes stochastic (without considering the selection of a starting city) and k becomes an algorithm parameter. Also, the time complexity of the algorithm becomes higher, as it is necessary to compute the $n - 1$ (each city has $n - 1$ neighbors) nearest neighbors for each city. Computing the complete neighbors list guarantees that it is possible to chose among the k or $k - a$ nearest yet-unvisited neighbors of the current city in every step. Since it is enough to compute the list of nearest neighbors only once, it does not require a huge computational effort. The time complexity of computing the complete nearest neighbors list using a simple sorting algorithm is $\Theta(n^3)$.

For parameter values of k close to $n - 1$ the algorithm behaves like a random procedure, since most of the cities have an equal chance of being selected in every step. In case $k = 1$, NN-k behaves like the ordinary NNA. It can also be concluded that by increasing the value of k the solution quality decreases.

4. Solution Implementation

For the purpose of analyzing the impact on performance of population initialization with solutions constructed by NN-k, a GA was implemented in the C# programming language. Two versions were implemented. The first presents a common GA (hereinafter referred to as RndGA), whereas the second (hereinafter referred to as NNGA) differs from the first in the usage of NN-k instead of a random procedure for population initialization and during the recombination process. For that purpose a list of the $n-1$ nearest neighbors for each city was computed for a given problem instance. The used GA, whose mode of operation is depicted in Fig. 1, is similar to the algorithm used in [8].

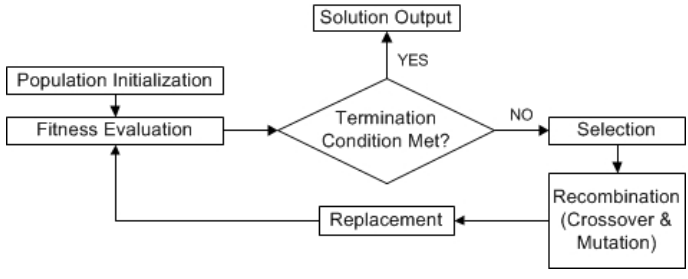


Figure 1. Mode of operation of the used GA.

For the representation of TSP solutions in the GA, path representation was used, that way each chromosome represents a detour sequence of the given cities, i.e. a string containing the ordering of the cities that are to be visited in that given order. Also, adequate crossover and mutation operators were used. Order crossover (OX) [6, 9, 13] was chosen as the crossover operator; it respects the absolute position of the ordering of the cities in the string. Simple inversion mutation (SIM) [6, 9, 13] was chosen as the mutation operator. SIM inverts the ordering of the cities in the string between two randomly chosen cut points. A short description of the operation mode of the aforementioned algorithm follows.

- Population Initialization. The population of size N is initialized, depending on the algorithm version, with differently generated solutions. In case of RndGA, the population is initialized with randomly generated solutions whereby duplicates are discarded making sure that every solution represents a distinct phenotype. Different population parts of NNGA are initialized with solutions generated by NN-k with different values of k. The size of the population parts initialized by NN-k with a corresponding value of k is shown in Fig. 2.

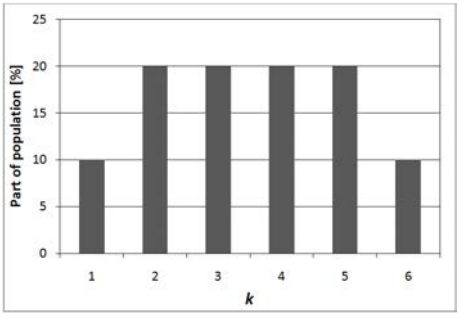


Figure 2. Size of the population parts and the corresponding value of parameter k used for their initialization.

- Fitness Evaluation. The fitness of every solution, i.e. chromosome is given by a negative value of the length of the tour it represents.
- Termination Condition. Algorithm execution is terminated after a preset number of iterations (generations), n_G is completed.
- Selection. Solutions that are to be removed from the population are chosen by a 3-tournament elimination selection without duplicates as described in [3]. The duplicate check is performed at the tournament level, guaranteing that the same solution participates only once in a given tournament. Since the best solution cannot be removed from the population by this selection procedure, elitism is inherently implemented.
- Crossover. The two best solutions participating in a given tournament are chosen as parents. If they have different fitness values, they are crossed over and two new solutions are created, but only the fitter one survives. If the parents have equal fitness values (they probably represent the same solution), one of them is mutated (until its fitness value does not change) and an offspring is generated randomly or by NN-k with $k = 1$ in the case of RndGA or NNGA, respectively.
- Mutation. Mutation is applied with a preset probability p_m ("per gene") based on which the chromosome mutation probability is calculated as $p_M = 1 - (1 - p_m)^n$, where n is the chromosome length, i.e. problem size. In case the offspring was generated randomly or by NN-k, i.e. it was not created by crossover, it is not mutated since a parent was already mutated before. Otherwise it is mutated but again, only if its fitness value it not greater than the population's current maximum.

- Replacement. Mortality M determines the percentage of the population to be replaced by new solutions acquired in the recombination process. The chromosome chosen for removal, i.e. the worst solution participating in a given tournament, is replaced by the offspring acquired in the recombination process. This kind of selection enables selection and recombination to take place in the same step, as described in [3].

5. Experimental analysis

Experimental analysis was conducted on a notebook computer with a dual-core (2×2.53 GHz) processor and 4 GB of RAM. Due to the sequential algorithm implementation, only one core was used during algorithm execution. The performance analysis of both algorithm versions (NNGA and RndGA) was conducted on several Euclidean TSP instances taken over from TSPLIB [11].

Parameter values of the GA used during the experimental analysis were acquired by a preliminary analysis and are se follows: The number of iterations to be executed (n_G) was 10000; the population size (N) was 30 chromosomes; mutation was applied with a probability (p_m) of 0.03; and the mortality (M) was 0.5. Results of the experimental analysis, given in Table 1 and 2, are based on 30 algorithm runs for each of the different initial population types and for each algorithm version and each TSP instance, respectively. The quality of the optimal solution to a particular TSP instance is given in parentheses. The numeric value in the name of a problem instance presents its size, n.

Table 1 shows the average quality (length) of the best (Avg. L_{bst}) and worst (Avg. L_{avg}) solutions in populations initialized with randomly generated solutions (popRnd), solutions generated by NN-k (popNN-k) and solutions generated by NNA (popNN). Also, the percentage of distinct fitness values (p_{df}), the pair-wise Hamming distance divided by the population size (H/N) and the standard deviation (σ) are shown.

Table 1. Diversity of different initial populations.

Instance	Init. pop. type	N	Avg. L_{bst}	Avg. L_{wst}	p_{df} [%]	H/N	σ
berlin52 (7542)	popNN-k	30	9024.3	20502.9	1.0	727.9	3171.9
	popNN		8417.0	10252.9	0.8	660.9	445.5
	popRnd		26476.1	32943.2	1.0	739.6	1590.3

As expected, the least diverse initial population was generated by using NNA. Although the solutions generated by NNA are of a better

quality compared to the solutions generated by the other two methods, it can be concluded that those solutions are concentrated in a small portion of the search space. Such an initial population is not very useful since the GA would likely get stuck in a local optimum early into the search. Although the pair-wise Hamming distance of the randomly initialized population is slightly higher, its standard deviation is significantly lower compared to the population initialized with NN-k. Also, by considering the average best and worst solution quality of those populations and their difference, it can be assumed that the solutions generated by NN-k are scattered around a lager portion of the search space and are concentrated around more promising parts of that space.

Table 2 shows the used algorithm version, the quality of the best (L_{bst}) and worst (L_{wst}) obtained solutions, the quality of average solutions (L_{avg}), the percentage the average solution exceeds the optimal one (D) and the average execution time (t_{avg}). Concerning the stability analysis of the algorithm, the standard deviation of the obtained solutions (σ) and the difference between the quality of the worst and best solutions (ΔL_{wb}) are also given in Table 2.

According to Table 2, it is easy to spot that a significantly better average solution quality for each of the used problem instances was achieved with NNGA than with RndGA. For some of the used problem instances, like d198, eil101, kroA200, pr124, pr152, pr226 and rat99, the quality of the worst solutions obtained with NNGA are still better than the quality of the average solutions obtained with RndGA. An especially big difference in solution quality can be noticed in the results obtained for problem instances d198, kroA200 and pr226 which also present the biggest TSP instances used in the analysis. It can also be noticed that the average execution times for both algorithm versions are very similar and therefore no performance was lost for that manner.

Considering the stability of both versions, according to Table 2 it can be concluded that NNGA is much more stabile, as the standard deviation of the obtained solutions and the difference between the quality of the worst and best obtained solutions is much less than for RndGA. For that reason it can be expected that solutions of quality close to the average will be achieve with a higher probability. Again, such a behavior can be tracked back to the initial population; since the initial population of NNGA is generated by NN-k it contains relatively good quality solutions, therefore getting stuck in a local optimum early into the search does not impact the overall performance as much as it does in the case of RndGA whose initial population is generated randomly and contains only solutions of poor quality.

Table 2. Experimental results.

Instance	Alg. ver.	L_{bst}	L_{avg}	L_{wst}	σ	ΔL_{wb}	D [%]	t_{avg}[s]
berlin52	NNGA	7542	7585.2	7865	90.7	323	0.6	1.6
(7542)	RndGA	7542	7766.2	8013	184.7	471	3.0	1.6
d198	NNGA	16080	16330.0	16541	115.6	461	3.5	15.5
(15780)	RndGA	16804	17307.9	17746	213.9	942	9.7	16.6
eil76	NNGA	545	553.5	559	3.5	14	2.9	3.0
(538)	RndGA	546	561.8	575	7.1	29	4.4	2.9
eil101	NNGA	633	643.5	659	5.8	26	2.3	4.7
(629)	RndGA	652	668.7	689	7.5	37	6.3	4.6
kroA100	NNGA	21343	21769.1	22345	240.0	1002	2.3	4.6
(21282)	RndGA	21357	22006.1	22940	431.4	1583	3.4	4.6
kroA200	NNGA	29837	30506.9	31734	407.8	1897	3.9	15.0
(29368)	RndGA	32960	34243.1	36311	863.0	3351	16.6	16.0
kroB100	NNGA	22199	22622.7	23287	281.8	1088	2.2	4.6
(22141)	RndGA	22454	23118.2	23862	390.0	1408	4.4	4.8
pr124	NNGA	59246	59985.2	61190	444.3	1944	1.6	6.4
(59030)	RndGA	59413	61448.4	63820	920.0	4407	4.1	6.8
pr152	NNGA	74417	75316.5	76169	442.1	1752	2.2	9.2
(73682)	RndGA	75157	77311.3	81156	1433.9	5999	4.9	9.3
pr226	NNGA	82514	83975.9	87256	1011.0	4742	4.5	18.4
(80369)	RndGA	85110	95447.2	107102	5310.3	21992	18.8	19.7
rat99	NNGA	1212	1248.4	1278	17.0	66	3.1	4.7
(1211)	RndGA	1219	1290.7	1324	23.7	105	6.6	4.5
rd100	NNGA	7932	8222.4	8526	135.3	594	3.9	4.9
(7910)	RndGA	8019	8334.5	8683	187.4	664	5.4	4.5
st70	NNGA	675	687.3	717	9.1	42	1.8	2.8
(675)	RndGA	676	695.0	726	11.7	50	3.0	2.7

Figures 3 and 4 depict the absolute fitness values (since the chromosome fitness is given by the negative value of the length of the tour it represents) of the best and worst chromosome, as well as the average fitness value of the whole population throughout the generations for both algorithm versions. The figures are based on 10 algorithm runs for the TSP instance berlin52.

By comparing Figs. 3 and 4 it can be noticed that NNGA reaches a final solution faster than RndGA. The improved convergence speed may be attributed to the quality of the solutions contained within the initial population, since the population size, selection pressure, mortality and mutation probability are the same for both algorithm versions. Big oscillations of the quality of the worst solution as seen in Fig. 4 may be explained by the fact that an offspring is always randomly created if both parents have an equal fitness value. Since the offspring created by NN-k

with $k = 1$ (same as NNA) are of relatively good quality, the oscillations as seen in Fig. 3 are much smaller and not necessarily the consequence of those solutions. As can be seen in Table 1 and Figs. 3 and 4, the quality of the population initialized with randomly generated solutions is significantly worse than the quality of the population initialized with solutions generated by NN-k, and therefore NNGA has a huge advantage over RndGA at the start. That being said, NNGA starts the search around more promising parts of the search space while it initially takes RndGA some time to reach a solution quality similar to the quality of the initial population of NNGA.

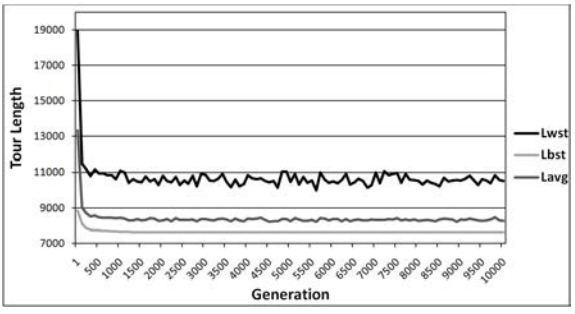


Figure 3. Absolute fitness values of the NNGA population throughout the generations.

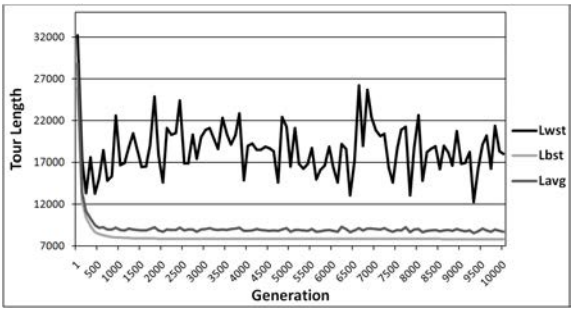


Figure 4. Absolute fitness values of the RndGA population throughout the generations.

6. Conclusion

This paper presented a GA whose population is initialized with solutions generated by NNA. The NNA was adapted for the generation of a diverse population. In every step of the algorithm one among the k yet-unvisited nearest neighbors in relation to the current visited city

is randomly chosen as the next to be visited. Besides the population initialization, the adapted NNA was used during the recombination process and an offspring is generated by NNA if both of the selected parents have an equal fitness value. Experimental analysis conducted on several instances of the symmetric TSP showed that significantly better solutions can be achieved with the described algorithm than a GA whose population is initialized with randomly generated solutions, since it enables a more effective and efficient exploration of the search space. The effectiveness and efficiency may be mainly attributed to the solutions contained within the initial population. Future work may include the testing whether still significantly better results would be achieved if a local search algorithm, like the 2-opt algorithm, was to be incorporated in the GA. Also, the application of the described algorithm to variations of the TSP, like the TSP with backhauls, might prove interesting.

Acknowledgements

This work was supported by research project grant No. 165-0362980-2002 from the Ministry of Science, Education and Sports of the Republic of Croatia.

References

[1] D. L. Applegate, R. E. Bixby, V. Chvatal, and W. J. Cook. *The Traveling Salesman Problem: A Computational Study*. Princeton University Press, 2006.

[2] S. Consoli and K. Darby-Dowman. Combinatorial optimization and metaheuristics. *Ann. Oper. Res.*, 140(1):189–213, 2007.

[3] M. Golub, D. Jakobovic, and L. Budin, Parallelization of elimination tournament selection without synchronization. In *Proc. International Conference on Intelligent Engineering Systems*, pages 85–89, 2001.

[4] D. S. Johnson and L. A. McGeoch. The Traveling Salesman Problem: A Case Study in Local Optimization. In E. H. L. Aarts and J. K. Lenstra, editors. *Local Search in Combinatorial Optimization*. John Wiley and Sons, pages 215–310, 1997.

[5] D. Kaur and M. M. Murugappan. Performance enhancement in solving travelling salesman problem using hybrid genetic algorithm. In *Proc. Annual Meeting of the North American Fuzzy Information Processing Society*, pages 1–6, 2008.

[6] P. Larranaga, C. M. H. Kuijpers, R. H. Murga, I. Inza, and S. Dizdarevic. Genetic algorithms for the traveling salesman problem: A review of representations and operators. *Artif. Intell. Rev.*, 13(2):129–170, 1999.

[7] H. Maaranen, K. Miettinen and A. Penttinen. On Initial Populations of a Genetic Algorithm for Continuous Optimization Problems. *J. Global Optim.*, 37(3):405–436, 2007.

[8] G. Martinović and D. Bajer. Impact of double operators on the performance of a genetic algorithm for solving the traveling salesman problem. *Lect. Notes Comput. Sc.*, 7076:290–298, 2011.

[9] S. N. Sivanandam and S. N. Deepa. *Introduction to Genetic Algorithms*. Springer-Verlag, 2008.

[10] R. Sivaraj, T. Ravichandran, and R. D. Priya. Boosting performance of genetic algorithm through selective initialization. *European Journal of Scientific Research*, 68(1):93–100, 2012.

[11] TSPLIB, Ruprecht-Karls-Universitt Heidelberg, `http://www2.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/`, accessed: April 2011.

[12] Z. Wang, H. Duan, and X. Zhang. An improved greedy genetic algorithm for solving travelling salesman problem. In *Proc. Fifth International Conference on Natural Computation*, pages 374–378, 2009.

[13] Y. Xinjie and G. Mitsuo. *Introduction to Evolutionary Algorithms*. Springer-Verlag, 2010.

AN INVESTIGATION ON THE CHAOS DRIVEN DIFFERENTIAL EVOLUTION: AN INITIAL STUDY

Roman Šenkeřík
Faculty of Applied Informatics, Tomas Bata University in Zlín, Czech Republic
senkerik@fai.utb.cz

Donald Davendra, Ivan Zelinka
Faculty of Electrical Engineering and Computer Science, VSB-Technical University of Ostrava, Ostrava-Poruba, Czech Republic
{donald.davendra; ivan.zelinka}@vsb.cz

Michal Pluháček, Zuzana Oplatková
Faculty of Applied Informatics, Tomas Bata University in Zlín, Czech Republic
{pluhacek; oplatkova}@fai.utb.cz

Abstract This paper outlines the initial investigations on the concept of a chaos driven Differential Evolution. The focus of this paper is the embedding of chaotic systems in the form of chaos number generator for Differential Evolution. The chaotic systems of interest are the discrete dissipative systems. Three chaotic systems were selected as possible chaos number generators for Differential Evolution. Repeated simulations were performed on benchmark function sets. Finally, the obtained results are compared with canonical Differential Evolution.

Keywords: Chaos, Differential evolution, Evolutionary algorithms.

1. Introduction

During the past five years, usage of new intelligent systems in engineering, technology, modeling, computing and simulations has attracted the attention of researchers worldwide. The most current methods are mostly based on soft computing, which is a discipline tightly bound to computers, representing a set of methods of special algorithms, belong-

ing to the artificial intelligence paradigm. The most popular of these methods are neural networks, evolutionary algorithms, fuzzy logic, and genetic programming. Presently, evolutionary algorithms are known as a powerful set of tools for almost any difficult and complex optimization problem. Ant Colony (ACO), Genetic Algorithms (GA), Differential Evolution (DE), Particle Swarm Optimization (PSO) and Self Organizing Migration Algorithm (SOMA) are some of the most potent heuristics available.

Recent studies have shown that Differential Evolution (DE) [4] has been used for a number of optimization tasks, [3, 8] has explored DE for combinatorial problems, [1] has hybridized DE whereas [6, 9, 10] has developed self-adaptive DE variants. This paper is aimed at investigating the chaos driven DE. Although a number of DE variants have been recently developed, the focus of this paper is the embedding of chaotic systems in the form of chaos number generator for DE and its comparison with the canonical DE.

This research is an extension and continuation of the previous initial application based experiment with chaos driven DE [2]. The chaotic systems of interest are discrete dissipative systems. Three different chaotic systems were selected as the chaos number generator for DE.

Firstly, Differential Evolution is explained. The next sections are focused on the description of used chaotic systems and benchmark test functions. Results and conclusion follow afterwards.

2. Differential Evolution

DE is a population-based optimization method that works on real-number-coded individuals [4]. A schematic is given in Fig. 1.

There are essentially five main steps of the code (see Fig. 1). Step 1. describes the input to the heuristic. D is the size of the problem, G_{max} is the maximum number of generations, NP is the total number of solutions, F is the scaling factor of the solution and CR is the factor for crossover. F and CR together make the internal tuning parameters for the heuristic.

Step 2. outlines the initialization of the heuristic. Each solution $x_{j,i,G=0}$ is created randomly between the two bounds $x^{(lo)}$ and $x^{(hi)}$. The parameter j represents the index to the values within the solution and i indexes the solutions within the population. So, to illustrate, $x_{4,2,0}$ represents the fourth value of the second solution at the initial generation.

After initialization, the population is subjected to repeated iterations in step 3.

Step. 4 describes the conversion routines of DE. Initially, three random numbers r_1, r_2, r_3 are selected, unique to each other and to the current indexed solution$i$ in the population in step 4.1. Henceforth, a new index j_{rand} is selected in the solution. j_{rand} points to the value being modified in the solution as given in step 4.2. In step 4.3, two solutions, $x_{j,r_1,G}$ and $x_{j,r_2,G}$ are selected through the index r_1 and r_2 and their values subtracted. This value is then multiplied by F, the predefined scaling factor. This is added to the value indexed by r_3.

However, this solution is not arbitrarily accepted in the solution. A new random number is generated, and if this random number is less than the value of CR, then the new value replaces the old value in the current solution. The fitness of the resulting solution, referred to as a perturbed vector $u_{j,i,G}$ is then compared with the fitness of $x_{j,i,G}$. If the fitness of $u_{j,i,G}$ is greater than the fitness of $x_{j,i,G}$, then $x_{j,i,G}$ is replaced with $u_{j,i,G}$; otherwise, $x_{j,i,G}$ remains in the population as $x_{j,i,G+1}$. Hence the competition is only between the new *child* solution and its *parent* solution.

DE is quite robust, fast, and effective, with global optimization ability. It does not require the objective function to be differentiable, and it works well even with noisy and time-dependent objective functions. Description of the used DERand1Bin strategy is presented in Table 1 together with the description and comparison of the three other most common and used DE strategies. These strategies differ in the way of calculating the perturbed vector $u_{j,i,G}$. Please refer to [4] and [5] for the detailed complete description of all other strategies.

Table 1. Description of selected DE strategies.

Strategy	Formulation
DERand1Bin	$u_{j,i,G+1} = x_{j,r1,G} + F\,(x_{j,r2,G} - x_{j,r3,G})$
DERand2Bin	$u_{j,i,G+1} = x_{j,r5,G} + F\,(x_{j,r1,G} - x_{j,r2,G} - x_{j,r3,G} - x_{j,r4,G})$
DEBest2Bin	$u_{j,i,G+1} = x_{j,Best,G} + F\,(x_{j,r1,G} - x_{j,r2,G} - x_{j,r3,G} - x_{j,r4,G})$
DELocalToBest	$u_{j,i,G+1} = x_{j,i,G} + F_{rand}\,(x_{j,Best,G} - x_{j,i,G}) + F\,(x_{j,r1,G} - x_{j,r2,G})$

3. Chaotic Maps

This section contains the description of three discrete chaotic maps used as the random generator for DE. Iterations of the chaotic maps were used for the generation of real numbers in the process of crossover based on the user defined CR value and for the generation of the integer values used for selection of solutions (individuals).

1. Input: $D, G_{\max}, NP \geq 4, F \in (0,1+), CR \in [0,1]$, and initial bounds: $x^{(lo)}, x^{(hi)}$.

2. Initialize: $\begin{cases} \forall i \leq NP \wedge \forall j \leq D : x_{i,j,G=0} = x_j^{(lo)} + rand_j[0,1] \bullet \left(x_j^{(hi)} - x_j^{(lo)}\right) \\ i = \{1,2,...,NP\},\ j = \{1,2,...,D\},\ G = 0,\ rand_j[0,1] \in [0,1] \end{cases}$

$\begin{cases} \text{3. While } G < G_{\max} \\ \quad \begin{cases} \text{4. Mutate and recombine:} \\ \quad \text{4.1} \quad r_1, r_2, r_3 \in \{1,2,....,NP\}, \text{ randomly selected, except: } r_1 \neq r_2 \neq r_3 \neq i \\ \quad \text{4.2} \quad j_{rand} \in \{1,2,...,D\}, \text{ randomly selected once each } i \\ \quad \text{4.3} \quad \forall j \leq D, u_{j,i,G+1} = \begin{cases} x_{j,r_3,G} + F \cdot (x_{j,r_1,G} - x_{j,r_2,G}) \\ \qquad \text{if } (rand_j[0,1] < CR \vee j = j_{rand}) \\ x_{j,i,G} \quad \text{otherwise} \end{cases} \\ \text{5. Select} \\ x_{i,G+1} = \begin{cases} u_{i,G+1} & \text{if } f(u_{i,G+1}) \leq f(x_{i,G}) \\ x_{i,G} & \text{otherwise} \end{cases} \end{cases} \\ G = G + 1 \end{cases}$

Figure 1. DE shematic.

3.1 Dissipative Standard Map

The dissipative standard map is a two-dimensional chaotic map. The parameters used in this work are $b = 0.1$ and $k = 8.8$ as suggested in [7]. The Dissipative standard map is given in Fig. 2. The map equations are given in Eqs. (1) and (2).

$$X_{n+1} = X_n + Y_{n+1}(mod 2\pi) \tag{1}$$

$$Y_{n+1} = bY_n + k \sin X_n(mod 2\pi) \tag{2}$$

3.2 Arnolds Cat Map

The Arnolds cat map is a simple two dimensional discrete system that stretches and folds points (x, y) to $(x + y, x + 2y) mod 1$ in phase space. The map equations are given in Eqs. (3) and (4). This map uses parameter $k = 2.0$ as suggested in [7]. The x-y plot of this map is depicted in Fig. 2.

$$X_{n+1} = X_n + Y_n(mod 1) \tag{3}$$

$$Y_{n+1} = X_n + kY_n(mod 1) \tag{4}$$

3.3 Sinai Map

The Sinai map is a simple two-dimensional discrete system similar to the Arnolds cat map. The map equations are given in Eqs. (5) and (6).

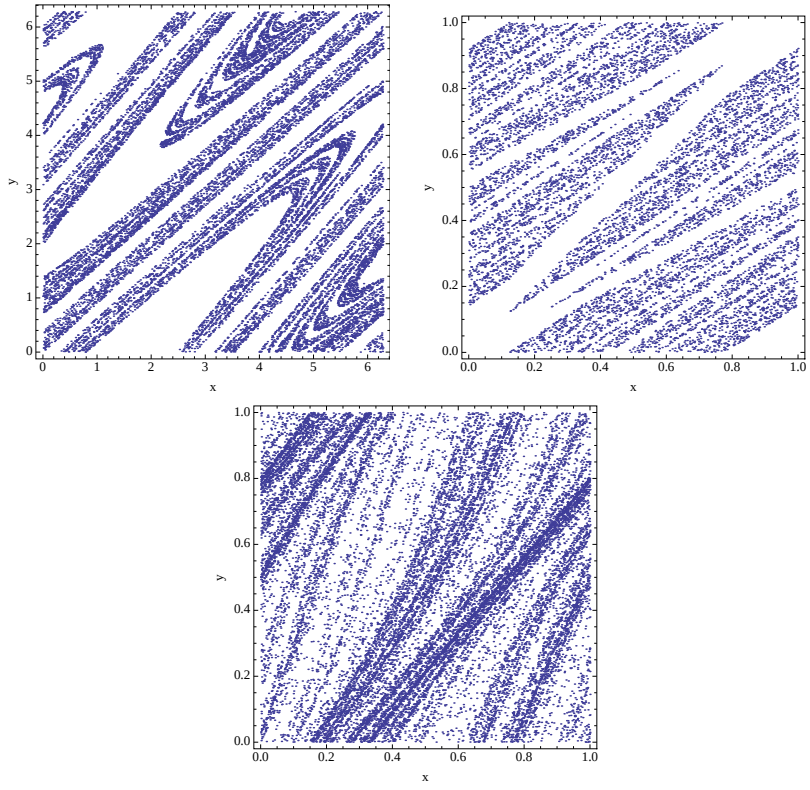


Figure 2. Dissipative standard map (upper left), Arnold's cat map (upper right), Sinai map (below).

The parameter used in this work is $\delta = 0.1$ as also suggested in [7]. The Sinai map is depicted in Fig. 2.

$$X_{n+1} = X_n + Y_n + \delta \cos 2\pi Y_n (mod 1) \tag{5}$$

$$Y_{n+1} = X_n + 2Y_n (mod 1) \tag{6}$$

4. Benchmark Functions

For the purpose of evolutionary algorithms performance comparison within this initial research, the following three test functions were selected: Schwefel function Eq. (7), Ackley I function Eq. (8) and Rastrigin function Eq. (9).

$$f(x) = \sum_{i=1}^{D} -x_i \sin\left(\sqrt{|x_i|}\right) \tag{7}$$

$$f(x) = \sum_{i=1}^{D-1} \left(\frac{1}{e^5} \sqrt{(x_i^2 + x_{i+1}^2)} + 3\left(\cos(2x_i) + \sin(2x_{i+1})\right) \right) \quad (8)$$

$$f(x) = 10D + \sum_{i=1}^{D} x_i^2 - 10\cos(2\pi x_i) \quad (9)$$

5. Results

The novelty of this approach represents the utilization of discrete chaotic maps as a random generator for DE. In this paper, the canonical DE strategy DERand1Bin and three versions of ChaosDE were used. The parameter settings for both canonical DE and ChaosDE were obtained analytically based on numerous experiments and simulations (see Table 2). Experiments were performed in an environment of Wolfram Mathematica, canonical DE therefore used the built-in Mathematica software random number generator. All experiments used different initialization, i.e. different initial population was generated in each run of Canonical or ChaosDE.

Table 2. Parameter set up for canonical and ChaosDE.

Parameter	Experiment 1	Experiment 2
Popsize	10D	10D
F	0.8	0.8
CR	0.8	0.8
Dimensions	2 - 8	10
Generations	200	∞
Max Cost Function Evaluations (CFE)	4000 – 16000	∞

Within this research, two experiments were performed. The first one utilizes the maximum number of generations fixed at 200 generations. This allowed the possibility to analyse the progress of DE within a limited number of generations and cost function evaluations. In the second case, the number of generations was unlimited. The main observed parameters were the total number of cost function evaluations and the time in seconds required for finding of the global minimum of the used test functions.

The results of the first experiment is shown in Tables 3, 5 and 7, which represent the average deviations from the known global minimum for 20 repeated runs of the different variants of utilized DE.

Tables 4, 6 and 8 contain the results for the second experiment. These tables show the average time in seconds and number of CFE (Cost Function Evaluations) required for the finding of global minimum for 20 repeated runs of the evolutionary algorithms.

The bold values depict the best value.

Table 3. Average difference from global minimum for the Schwefel function: $2D$–$8D$.

Dim.	Known Value	Canonical DE	ChaosDE Dissipative	ChaosDE Arnold cat	ChaosDE Sinai
2	-837.966	**2.2545 10^{-4}**	**2.2545 10^{-4}**	**2.2545.10^{-4}**	**2.2545 10^{-4}**
4	-1675.932	7.1499 10^{-4}	**5.0084 10^{-4}**	3.4091 10^{-3}	2.0311 10^{-3}
6	-2513.898	228.818	**158.723**	315.082	306.371
8	-3351.864	**800.731**	805,248	872.714	822.857

Table 4. Average CFE and evaluation time for the Schwefel function: $10D$.

	Canonical DE	ChaosDE Dissipative	ChaosDE Arnold cat	ChaosDE Sinai
CFE	304400	**167100**	340800	263700
Time (seconds)	162.63	**124.62**	249.58	193.77

Table 5. Average difference from global minimum for the Ackley I function: $2D$–$8D$.

Dim.	Known Value	Canonical DE	ChaosDE Dissipative	ChaosDE Arnold cat	ChaosDE Sinai
2	-4.5901	**1.6341 10^{-6}**	**1.6341 10^{-6}**	**1.6341 10^{-6}**	**1.6341 10^{-6}**
4	-10.46137	2.39 10^{-3}	**2.32 10^{-3}**	0.01275	0.07054
6	-16.29877	4.3451	**3.31813**	4.68185	4.5361
8	-22.13611	14.1343	**11.7813**	14.5216	14.6120

6. Brief Analysis of the Results

The results in Tables 3–8 show that using the Dissipative standard map as a random generator has actually improved the performance of DE. The performance of DE significantly improved in both experiments

Table 6. Average CFE and evaluation time for the Ackley I function: $10D$.

	Canonical DE	ChaosDE Dissipative	ChaosDE Arnold cat	ChaosDE Sinai
CFE	377400	**222500**	331300	474100
Time (seconds)	237.598	**172.520**	210.188	371.515

Table 7. Average difference from global minimum for the Rastrigin function: $2D$–$8D$.

Dim.	Known Value	Canonical DE	ChaosDE Dissipative	ChaosDE Arnold cat	ChaosDE Sinai
2	0	**0**	**0**	**0**	**0**
4	0	1.31435	**0.66387**	1.81004	1.83477
6	0	10.8232	**10.4494**	11.9057	11.7926
8	0	24.4106	**21.6622**	23.6621	24.578

Table 8. Average CFE and evaluation time for the Rastrigin function: $10D$.

	Canonical DE	ChaosDE Dissipative	ChaosDE Arnold Cat	ChaosDE Sinai
CFE	1997800	**1022100**	2089600	2414600
Time (seconds)	1686.921	**1123.790**	1720.638	2160.638

for the limited number of generations ($2D$–$8D$) and for unlimited simulation ($10D$). The results achieved in the first experiment for the other two chaotic systems are comparable with the results of canonical DE. However, in the second experiment for higher dimensions, the DE with chaotic systems Sinai (for Schwefel function) or Arnold's cat map (for Ackley I function) exhibits better performance than canonical DE, thus faster finding of the global minimum (see Fig. 3). This figure shows the evolution of the cost function value in time (iterations). Time (iterations) are converted here to the number of cost function evaluations, where in each iteration the cost function is evaluated exactly as many times as there are individuals (solutions) in the population.

Based on data presented in Tables 3–8 and an illustrative example of the time evolution of the cost function values depicted in Fig. 3, the results can be summarized as follows:

Using the Dissipative standard map has significantly improved performance of DE for both experiments.

Using the Sinai map or Arnold's cat map in specific cases helped to improve the behavior of the DE for higher dimension problems; however, for lower dimension problems it is comparable to the performance of canonical DE.

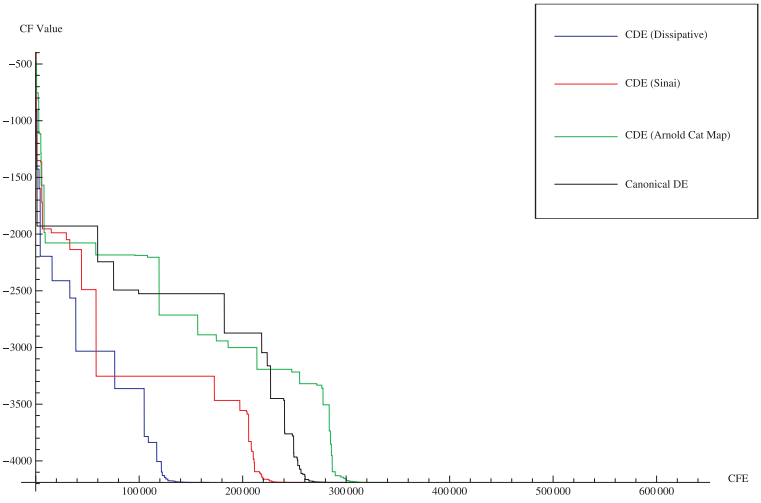


Figure 3. Example of the time evolution of the cost function values for Schwefel function and the second experiment – $10D$.

7. Conclusion

In this paper chaos driven DE were tested and compared with canonical DE strategy.

Based on obtained results, it may be claimed, that the developed ChaosDE driven by means of the chaotic Dissipative standard map gives better results than other compared heuristics. Furthermore, all obtained results point to the fact that they are very sensitive to the selection of chaotic system being used as a random generator. Any change in the selection of chaotic system or its parameter adjustment can cause radical improvement of DE performance, however on the downside it can cause a worsening of observed parameters and subsequently the behavior of the evolutionary algorithms as well.

Since this was an initial study, future plans include experiments with benchmark functions in higher dimensions, testing of different chaotic systems and obtaining a large number of results to perform statistical tests.

Furthermore chaotic systems have additional parameters, which can by tuned. This issue opens up the possibility of examining the impact of these parameters to generation of random numbers, and thus influence on the results obtained using differential evolution. One of possible approach in this issue is to use meta-evolution.

Acknowledgement

This work was supported by European Regional Development Fund under the project CEBIA-Tech No. CZ.1.05/2.1.00/03.0089 and project IT4Innovations Centre of Excellence No. CZ.1.05/1.1.00/02.0070, and by Internal Grant Agency of Tomas Bata University under the project No. IGA/FAI/2012/037.

References

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

[2] D. Davendra, I. Zelinka, and R. Šenkeřík. Chaos driven evolutionary algorithms for the task of PID control. *Comput. Math. Appl.*, 60(4):1088–1104, 2010.

[3] G. Onwubolu and D. Davendra, editors. *Differential Evolution: A handbook for Permutation-based Combinatorial Optimization.* Springer, 2009.

[4] K. V. Price. An Introduction to Differential Evolution. In D. Corne, M. Dorigo, and F. Glover, editors. *New Ideas in Optimization*, pages 79–108, McGraw-Hill, 1999.

[5] K. V. Price and R. M. Storn. Differential evolution homepage. Available: `http://www.icsi.berkeley.edu/~storn/code.html`, 2001.

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

[7] J. C. Sprott. *Chaos and Time-Series Analysis.* Oxford University Press, 2003

[8] M. F. Tasgetiren, P. N. Suganthan, and Q. K. Pan. An ensemble of discrete differential evolution algorithms for solving the generalized traveling salesman problem. *Appl. Math. Comput.*, 215(9):3356–3368, 2010.

[9] J. Zhang and A. C. Sanderson. JADE: Self-adaptive differential evolution with fast and reliable convergence performance. In *Proc. IEEE Congres on Evoutionary Computation*, pages 2251–2258, 2007.

[10] J. Zhang and A. C. Sanderson. Self-adaptive multiobjective differential evolution with direction information provided by archived inferior solutions. In *Proc. IEEE World Congress on Evolutionary Computation*, pages 2801–2810, 2008.

HANDLING NON-SEPARABILITY IN THREE STAGE OPTIMAL MEMETIC EXPLORATION

Ilpo Poikolainen, Fabio Caraffini, Ferrante Neri, Matthieu Weber
Department of Mathematical Information Technology, University of Jyväskylä, Finland
{ilpo.poikolainen; fabio.caraffini; ferrante.neri; matthieu.weber}@jyu.fi

Abstract Three Stage Optimal Memetic Exploration (3SOME) is a powerful algorithmic framework which progressively perturbs a single solution by means of three distinct operators. This paper proposes the integration of a simple and still powerful component to enrich the 3SOME algorithm. This component, based on a special mutation scheme of Differential Evolution, replaces an existing 3SOME operator and allows 3SOME to efficiently handle non-separable problems. The resulting algorithm, namely Rotationally Invariant Three Stage Differential Evolution (RI3SOME) displays a good performance on a set of various problems. In particular, the proposed component in RI3SOME improves upon 3SOME performance for non-separable problems and without excessively deteriorating the algorithmic performance for separable problems. The comparison with modern optimization algorithms shows that RI3SOME, despite its simplicity and modest use of computational resources, displays a respectable performance.

Keywords: Computational intelligence optimization, Memetic computing, Ockham razor, Separability.

1. Introduction

A function of n independent variables is said to be separable if it can be expressed as a sum of n functions, each of them depending on only one variable. From an optimization viewpoint, these functions are very easy to handle as the optimization problem in n variables can be expressed as n problems in one variable. In other words, a separable function can be efficiently optimized by perturbing separately each variable. Unfortunately, in real world applications it often (if not always) happens that the objective functions are non-separable. Nonetheless, while in mathematics a function can be hardly classified either as separable or

non-separable, the concept of separability in computational intelligence optimization is more fuzzy: a function can be fully separable, moderately separable, moderately non-separable, fully non-separable. Modern testbeds tend to classify test problems according to similar criteria, see [6]. To better understand the practical implications of this remark, a function, even if non-separable, can still be handled by perturbing separately each variable. Even though this approach will not lead to the detection of the optimum it can still spot an area of the decision space characterized by a high performance. Thus, if coupled with some other components, the resulting algorithm can turn out being an efficient optimizer.

The idea of searching optima not along preferential directions but performing search moves which simultaneously involve multiple variables (diagonal) has been studied in various contexts over the last few decades. For example, in 60's Rosenbrock and Powell algorithms, unlike the Hooke-Jeeves algorithm, were tackling optimization problems by including diagonal moves.

In modern times, with the development of computational intelligence optimization, many move operators, e.g. several kinds of recombination in Evolutionary Algorithms (EAs), naturally perform diagonal moves. On the other hand, only in a minor portion of literature the separability is explicitly handled in optimization problems. One famous example is the Covariance Matrix Adaptation (CMA) integrated within Evolution Strategy (ES) frameworks. The first algorithm employing this logic, known as CMAES, see [2]. The general algorithmic idea is that new trial solutions are generated by a distribution which progressively adapts to the fitness landscape and thus performs search moves along the most convenient direction. Other relevant algorithms based on the CMA have been proposed in literature, e.g. [1, 3, 5].

In [14], the non-separability of the fitness landscape in handled by the employment of structured populations. Some biologically inspired algorithms, despite their efficiency, naturally perform a biased search along specific axes. This situation occurs in Differential Evolution (DE) where the crossover is executed by an inheritance mechanism of variables from a parent to an offspring solution. As a consequence, even when DE can be very efficient for some fitness landscapes, it may dramatically deteriorate its performance after a rotation operation. It must be observed that the rotation operation over a separable function jeopardizes the separability (with respect to the original axes) of the problem making it non-separable. In order to handle these conditions, several corrections to the DE variation operators have been designed. For example, in [11], a reference rotation procedure is integrated within DE crossover and in

[12] a modified DE crossover is introduced by making use of the centroid point. A classical but still efficient way to obtain a rotationally invariant DE is to combine an arithmetic crossover within the mutation scheme. This mutation scheme, namely DE/current-to-rand/1, has been presented in [10].

Recently, in [4], it has been shown that algorithms with a simple structure can be as efficient as complex algorithms (Ockham's Razor) while requiring little memory and having low computational overhead (see [4] for details). Thus, when an algorithm is designed, it is important to build it up with a bottom-up approach by including the minimum amount of components and figuring out about the role of each operator. In accordance with Memetic Computing (MC) fashion, an algorithm is a structure composed of multiple operators which interact and cooperate to tackle various optimization problems, see [7, 8]. An algorithm composed of three operators and namely Three Stage Optimal Memetic Exploration (3SOME) has also been proposed in [4] as an example for the Ockham's Razor principle in MC. The main drawback of the 3SOME algorithm was that the non-separability was not explicitly addressed and thus the algorithm displayed a not so good performance in some cases. This paper proposes a modified algorithm based on 3SOME structure where a component for explicitly tackling non-separability is included in the algorithmic framework. The remainder of this paper is organized in the following way. Section 2 describes the proposed Rotationally Invariant Three Stage Memetic Exploration (RI3SOME). Section 3 displays the experimental testbed and numerical results related to the proposed RI3SOME with respect to 3SOME and other modern optimization algorithms. Section 4 gives the conclusion of this work.

2. Rotationally Invariant Three Stage Memetic Exploration

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

At the beginning of the optimization problem one candidate solution is randomly sampled within the decision space D. In analogy with compact optimization, see [9], we will refer to this candidate solution as elite and indicate it with the symbol x_e. In addition to x_e, the algorithm makes use of another memory slot for attempting to detect other solutions. The latter solution, namely trial, is indicated with x_t. In the

following sub-sections the three exploratory stages and their coordination are described.

The proposed algorithm, like the 3SOME structure proposed in [4], is composed of three operators which perturb a single solution, thus exploring the decision space from complementary perspectives. The first operator, namely long distance exploration, is a randomized long distance search where the newly generated solution x_t inherits part of the elite solution x_e by means of the exponential crossover typical of DE, see [9]. This operator is continued until a new solution outperforming the original one is generated. The second operator, namely intermediate diagonal exploration, processes the elite solution and samples a trial solution x_t by means of the DE/current-to-rand/1 mutation. This operation is continued until a given budget is expired. If the final solution outperforms the initial one, a replacement occurs and a new elite is generated. Regardless of the success of the intermediate diagonal exploration, the elite solution x_e is processed by the short distance exploration. This third operator perturbs the variables separately and attempts to quickly and deterministically descend the corresponding basin of attraction. In this way a component for handling non-separable functions supports the first and third operators which are especially efficient for separable and moderately non-separable problems.

2.1 Long Distance Exploration

This exploration move attempts to detect a new promising solution within the entire decision space. While the elite x_e is retained, at first, a trial solution x_t is generated by randomly sampling a new set of n genes. Subsequently, the DE exponential crossover is applied between x_e and x_t, see [9]. This exploration stage performs the global stochastic search and thus attempts to detect unexplored promising areas of the decision space. On the other hand, while this search mechanism extensively explores the decision space, it also promotes retention of a small section of the elite within the trial solution. This kind of inheritance of some genes appears to be extremely beneficial in terms of performance with respect to a stochastic blind search (which would generate a completely new solution at each step). If the trial solution outperforms the elite, a replacement occurs. A replacement has been set also if the newly generated solution has the same performance of the elite. This is to prevent the search getting trapped in some plateaus of the decision space. The pseudo-code of this component is shown in Algorithm 1. The long distance exploration is repeated until it does not detect a solution that outperforms the original elite. When a new promising solution is

detected, and thus the elite is updated, the stochastic short distance exploration is activated.

Algorithm 1 Long distance exploration

```
generate a random solution x_t within D
generate i = round(n · rand(0,1))
x_t[i] = x_e[i]
while rand(0,1) ≤ Cr do
   x_t[i] = x_e[i]
   i = i + 1
   if i == n then
      i = 1
   end if
end while
if f(x_t) ≤ f(x_e) then
   x_e = x_t
end if
```

2.2 Intermediate Diagonal Exploration

Around the solution x_e returned by the long distance exploration a hypercube is considered. This hypercube has a volume equal to one fifth of the volume of the entire decision space D and is centred around x_e. Subsequently, three points, x_r, x_s, and x_v, respectively are sampled (from an implementation viewpoint the points are progressively sampled and allocated into the trial solution to occupy only one memory slot). These points are then combined with x_e by means of DE/current-to-rand/1 to generate x_t:

$$x_t = x_e + K\left(x_v - x_e\right) + F'\left(x_r - x_s\right), \tag{1}$$

where K is the combination coefficient, which should be chosen with a uniform random distribution from $[0, 1]$ and $F' = K \cdot F$. The trial solution replaces the elite if it displays a higher performance. This operation is repeated for a given budget.

This component replaces the middle distance search of 3SOME, see [4]. In 3SOME the middle distance exploration samples points in the same hypercube and performs the exponential crossover to increase the exploitation and thus guide the search towards the improvements. This mechanism is equivalent to keep constant some variables and move along the others. In the present paper, we propose the inheritance of x_e to x_t by means of a linear combination and a search along all the directions simultaneously. This component is supposed to tackle, in a simple and computationally inexpensive way, non-separable problems. Trial

solutions can be generated outside the hypercube. The pseudo-code displaying the working principles of the intermediate diagonal distance exploration is given in Algorithm 2.

Algorithm 2 Intermediate diagonal distance exploration

```
while local budget condition do
    generate x_r, x_s, and x_v
    x_t = x_e + K (x_v − x_e) + F′ (x_r − x_s)
end while
if f (x_t) ≤ f (x_e) then
    x_e = x_t
end if
```

2.3 Short Distance Exploration

This exploration move attempts to fully exploit promising search directions. The meaning of this exploration stage is to perform the descent of promising basins of attraction and possibly finalize the search if the basin of attraction is globally optimal. De facto, the short distance exploration is a simple steepest descent deterministic local search algorithm, with an exploratory move similar to that of Hooke-Jeeves algorithm, or the first local search algorithm of the multiple trajectory search, see [13]. The short distance exploration stage requires an additional memory slot, which will be referred to as x_s (s stands for short). Starting from the elite x_e, this local search, explores each coordinate i (each gene) and samples $x_s[i] = x_e[i] - \rho$, where ρ is the exploratory radius. Subsequently, if x_s outperforms x_e, the trial solution x_t is updated (it takes the value of x_s), otherwise a half step in the opposite direction $x_s[i] = x_e[i] + \frac{\rho}{2}$ is performed. Again, x_s replaces x_t if it outperforms x_e. If there is no update, i.e. the exploration was unsuccessful, the radius ρ is halved. This exploration is repeated for all the design variables and stopped when a prefixed budget (equal to 150 iterations as suggested in [13]) is exceeded. The pseudo-code displaying the working principles of the short distance exploration is given in Algorithm 3.

After the application of the deterministic short distance exploration, if there is an improvement in the quality of the solution, the stochastic short distance exploration is repeated subsequently. Otherwise, if no improvement in solution quality is found, the long distance search is activated to attempt to find new basins of attractions.

As a remark, a toroidal management of the bounds has been implemented for the three operators above. This means that if, along the dimension i, the design variable $x[i]$ exceeds the bounds of a value ζ, it

is reinserted from the other end of the interval at a distance ζ from the edge, i.e. given an interval $[a, b]$, if $x[i] = b + \zeta$ it takes the value of $a + \zeta$.

Algorithm 3 Short distance exploration

```
while local budget condition do
  x_t = x_e
  x_s = x_e
  for i = 1 : n do
    x_s[i] = x_e[i] − ρ
    if f(x_s) ≤ f(x_t) then
      x_t = x_s
    else
      x_s[i] = x_e[i] + ρ/2
      if f(x_s) ≤ f(x_t) then
        x_t = x_s
      end if
    end if
  end for
  if f(x_t) ≤ f(x_e) then
    x_e = x_t
  else
    ρ = ρ/2
  end if
end while
```

3. Numerical Results

The proposed RI3SOME has been tested on the testbed in [6] (24 problems) in 10, 40, and 100 dimensions. It must be noted that functions f_1 to f_5 of that testbed are separable, while the remaining ones are not. In order to perform a fair comparison 3SOME and RI3SOME have been run with the same parameters, $\alpha_e = 0.05$ and ρ equal to 40% of the total decision space width. Both the budgets, for middle length and intermediate diagonal explorations, have been fixed equal to $4n$ points at each activation. Regarding the DE/current-to-rand/1 mutation in RI3SOME, the scale factor F has been set equal to 0.75 while K is a random number between 0 and 1. For an extensive discussion on the parameter setting of the 3SOME framework see [4]. In addition, RI3SOME has been compared with DE/current-to-rand/1 and with the same parameter setting of F used for RI3SOME and a population size of 50 individual. Finally, also (1+1)-CMAES proposed in [5] has been included in the comparison. The latter algorithm has been chosen since it is explicitly designed for non-separable problems and, like RI3SOME, is based on a single solution. However, it must be remarked that (1+1)-CMAES and RI3SOME do not require a similar computational overhead

(having an algorithmic structure similar to 3SOME, RI3SOME is very similar to the latter with regards to memory requirements and computational overhead). More specifically, while RI3SOME requires only two memory slots (one memory slot for the elite and one for the trial solution), (1+1)-CMAES requires to store a covariance matrix. Thus, the corresponding (1+1)-CMAES memory employment grows quadratically with the dimensionality of the problem. Each algorithm has been run for $5000 \times n$ fitness evaluations for each run. For each problem 100 runs have been performed.

Tables 1, 2, and 3 display the numerical results (in terms of final value and standard deviation) for the test problems considered in this work. The best results are highlighted in bold face. In order to strengthen the statistical significance of the results, the Wilcoxon Rank-Sum test has also been applied according to the description given in [15], where the confidence level has been fixed at 0.95. The null-hypothesis being that the results of both algorithms come from the same statistical distribution, the symbol "+"("-") indicates that the null-hypothesis is rejected (the reference algorithm thus performs better (worse) than the algorithm labeled on the top of the column) while the symbol "=" indicates that the null-hypothesis cannot be rejected and that both algorithms have a statistically equivalent performance. Note that DE/current-to-rand/1 is outperformed by RI3SOME on at least 22 test problems over dimensions 10, 40 and 100, and outperforms the latter only on f_16 in 10 dimensions. For this reason and due to lack of space, the results of that algorithm have been left out from the tables.

Numerical results show that the proposed RI3SOME algorithm tends to outperform 3SOME for non-separable problems (f_6 and following) while it is sometimes outperformed for separable problems (f_1 to f_5). It must be noted that in the case where the Wilcoxon rank-sum test indicates that RI3SOME is outperformed by 3SOME, the differences in average fitness values between these two algorithms are at least one order of magnitude smaller than the difference between 3SOME and the other two algorithms. It is also shown that RI3SOME appears to be much more promising than 3SOME in relatively high dimensions. This can be interpreted that the coordination of diverse components, the first perturbing the variables separately, the second performing diagonal movements is beneficial for hard to solve problems. The comparison with (1+1)-CMAES shows that RI3SOME and (1+1)-CMAES display a comparable performance. Since RI3SOME imposes lower computational requirements that (1+1)-CMAES, its implementation is preferable for those applications characterized by a limited hardware, see[4] and [9].

Table 1. Average Fitness ± Standard Deviation and Wilcoxon rank-sum test on the Fitness (reference = RI3SOME) for 10D case.

	RI3SOME	3SOME		(1+1)-CMAES	
f_1	$\mathbf{7.95e+01 \pm 1.24e-14}$	$\mathbf{7.95e+01 \pm 1.14e-14}$	=	$\mathbf{7.95e+01 \pm 0.00e+00}$	=
f_2	$\mathbf{-2.10e+02 \pm 1.50e-14}$	$\mathbf{-2.10e+02 \pm 1.77e-14}$	=	$4.16e+05 \pm 2.91e+06$	+
f_3	$-4.58e+02 \pm 1.68e+00$	$\mathbf{-4.61e+02 \pm 1.28e+00}$	-	$-3.92e+02 \pm 3.25e+01$	+
f_4	$-4.57e+02 \pm 2.35e+00$	$\mathbf{-4.60e+02 \pm 1.39e+00}$	-	$-3.63e+02 \pm 5.42e+01$	+
f_5	$6.45e+00 \pm 2.95e+01$	$\mathbf{5.32e+00 \pm 2.91e+01}$	=	$1.18e+01 \pm 5.27e+01$	+
f_6	$2.26e+02 \pm 8.70e+02$	$3.59e+01 \pm 5.97e-02$	-	$\mathbf{3.59e+01 \pm 0.00e+00}$	-
f_7	$1.03e+02 \pm 5.97e+00$	$1.06e+02 \pm 1.36e+01$	=	$\mathbf{1.03e+02 \pm 6.80e+01}$	-
f_8	$\mathbf{1.49e+02 \pm 1.51e-01}$	$\mathbf{1.49e+02 \pm 1.51e-01}$	=	$1.50e+02 \pm 1.20e+00$	+
f_9	$1.26e+02 \pm 1.00e+01$	$1.25e+02 \pm 1.78e+00$	=	$\mathbf{1.25e+02 \pm 1.70e+00}$	-
f_{10}	$2.16e+02 \pm 1.43e+02$	$3.44e+03 \pm 2.25e+04$	=	$\mathbf{-5.49e+01 \pm 0.00e+00}$	-
f_{11}	$1.54e+02 \pm 2.97e+01$	$1.57e+02 \pm 3.26e+01$	=	$\mathbf{7.63e+01 \pm 0.00e+00}$	-
f_{12}	$-6.05e+02 \pm 2.13e+01$	$-6.13e+02 \pm 1.18e+01$	=	$\mathbf{-6.21e+02 \pm 0.00e+00}$	-
f_{13}	$4.14e+01 \pm 1.14e+01$	$4.36e+01 \pm 1.28e+01$	=	$\mathbf{4.05e+01 \pm 1.08e+01}$	=
f_{14}	$-5.23e+01 \pm 2.48e-05$	$-5.23e+01 \pm 3.09e-05$	=	$\mathbf{-5.24e+01 \pm 0.00e+00}$	=
f_{15}	$1.09e+03 \pm 5.13e+01$	$1.10e+03 \pm 6.32e+01$	=	$\mathbf{1.09e+03 \pm 5.90e+01}$	=
f_{16}	$7.91e+01 \pm 4.58e+00$	$7.94e+01 \pm 4.41e+00$	=	$7.89e+01 \pm 4.57e+00$	=
f_{17}	$-1.13e+01 \pm 3.23e+00$	$-9.15e+00 \pm 8.04e+00$	=	$\mathbf{-1.26e+01 \pm 2.50e+00}$	=
f_{18}	$1.01e+01 \pm 2.23e+01$	$\mathbf{4.21e+00 \pm 2.16e+01}$	-	$6.93e+00 \pm 2.58e+01$	=
f_{19}	$-9.89e+01 \pm 1.92e+00$	$-9.81e+01 \pm 2.95e+00$	=	$\mathbf{-9.97e+01 \pm 1.74e+00}$	=
f_{20}	$\mathbf{-5.46e+02 \pm 3.06e-01}$	$\mathbf{-5.46e+02 \pm 2.36e-01}$	=	$-5.45e+02 \pm 3.56e-01$	+
f_{21}	$5.36e+01 \pm 1.49e+01$	$5.22e+01 \pm 1.27e+01$	=	$\mathbf{4.81e+01 \pm 7.52e+00}$	=
f_{22}	$-9.90e+02 \pm 1.26e+01$	$-9.89e+02 \pm 1.46e+01$	=	$\mathbf{-9.91e+02 \pm 1.07e+01}$	=
f_{23}	$\mathbf{7.80e+00 \pm 4.62e-01}$	$7.88e+00 \pm 5.48e-01$	=	$8.00e+00 \pm 6.60e-01$	=
f_{24}	$1.87e+02 \pm 3.65e+01$	$1.88e+02 \pm 4.30e+01$	=	$\mathbf{1.69e+02 \pm 2.68e+01}$	-

Table 2. Average Fitness ± Standard Deviation and Wilcoxon rank-sum test on the Fitness (reference = RI3SOME) for 40D case.

	RI3SOME	3SOME		(1+1)-CMAES	
f_1	$7.95e+01 \pm 2.70e-14$	$7.95e+01 \pm 1.98e-14$	=	$\mathbf{7.95e+01 \pm 0.00e+00}$	=
f_2	$\mathbf{-2.10e+02 \pm 3.08e-14}$	$\mathbf{-2.10e+02 \pm 3.00e-14}$	=	$-2.09e+02 \pm 6.24e-01$	+
f_3	$-4.23e+02 \pm 1.02e+01$	$\mathbf{-4.54e+02 \pm 3.52e+00}$	-	$5.45e+01 \pm 1.47e+02$	+
f_4	$-4.17e+02 \pm 1.08e+01$	$\mathbf{-4.50e+02 \pm 4.57e+00}$	-	$4.07e+02 \pm 2.18e+02$	+
f_5	$\mathbf{2.31e+01 \pm 1.28e+02}$	$4.58e+01 \pm 1.66e+02$	+	$6.57e+01 \pm 1.36e+02$	+
f_6	$3.59e+01 \pm 5.75e-07$	$3.59e+01 \pm 5.63e-07$	=	$\mathbf{3.59e+01 \pm 0.00e+00}$	=
f_7	$1.95e+02 \pm 4.14e+01$	$2.11e+02 \pm 6.18e+01$	=	$\mathbf{1.08e+02 \pm 4.77e+00}$	-
f_8	$1.52e+02 \pm 1.17e+01$	$1.55e+02 \pm 2.08e+01$	=	$\mathbf{1.50e+02 \pm 1.65e+00}$	=
f_9	$1.25e+02 \pm 1.79e+00$	$1.25e+02 \pm 1.42e+00$	=	$\mathbf{1.25e+02 \pm 1.70e+00}$	=
f_{10}	$9.04e+02 \pm 3.34e+02$	$3.89e+05 \pm 1.97e+06$	+	$\mathbf{-5.42e+01 \pm 1.03e+00}$	-
f_{11}	$\mathbf{3.59e+02 \pm 7.12e+01}$	$3.83e+02 \pm 6.35e+01$	=	$7.90e+04 \pm 5.52e+05$	+
f_{12}	$-6.12e+02 \pm 8.76e+00$	$-6.10e+02 \pm 8.87e+00$	=	$\mathbf{-6.21e+02 \pm 5.35e-04}$	-
f_{13}	$\mathbf{4.00e+01 \pm 1.07e+01}$	$4.24e+01 \pm 1.25e+01$	=	$4.18e+01 \pm 1.47e+01$	=
f_{14}	$-5.23e+01 \pm 6.56e-05$	$-5.23e+01 \pm 6.67e-05$	=	$\mathbf{-5.23e+01 \pm 7.89e-09}$	=
f_{15}	$1.76e+03 \pm 2.48e+02$	$2.04e+03 \pm 3.78e+02$	+	$\mathbf{1.69e+03 \pm 1.91e+02}$	=
f_{16}	$\mathbf{8.78e+01 \pm 6.29e+00}$	$8.80e+01 \pm 5.35e+00$	=	$9.01e+01 \pm 5.67e+00$	=
f_{17}	$-8.18e+00 \pm 1.53e+00$	$-5.39e+00 \pm 3.67e+00$	+	$\mathbf{-9.70e+00 \pm 1.44e+00}$	-
f_{18}	$1.80e+01 \pm 8.10e+00$	$2.61e+01 \pm 1.39e+01$	+	$\mathbf{1.13e+01 \pm 6.06e+00}$	-
f_{19}	$-9.51e+01 \pm 2.87e+00$	$-9.37e+01 \pm 3.34e+00$	+	$\mathbf{-9.57e+01 \pm 2.33e+00}$	=
f_{20}	$-5.45e+02 \pm 1.65e-01$	$\mathbf{-5.46e+02 \pm 1.28e-01}$	=	$-5.45e+02 \pm 1.89e-01$	=
f_{21}	$4.65e+01 \pm 6.85e+00$	$5.19e+01 \pm 1.57e+01$	=	$\mathbf{4.57e+01 \pm 8.91e+00}$	-
f_{22}	$-9.85e+02 \pm 1.53e+01$	$-9.85e+02 \pm 1.25e+01$	=	$\mathbf{-9.88e+02 \pm 8.00e+00}$	=
f_{23}	$\mathbf{7.89e+00 \pm 4.38e-01}$	$8.04e+00 \pm 5.27e-01$	=	$8.31e+00 \pm 6.33e-01$	+
f_{24}	$6.84e+02 \pm 1.66e+02$	$9.16e+02 \pm 2.89e+02$	+	$\mathbf{6.61e+02 \pm 1.21e+02}$	=

4. Conclusions

This paper proposes the integration, within 3SOME framework, of a rotationally invariant mutation from differential evolution as an al-

Table 3. Average Fitness ± Standard Deviation and Wilcoxon rank-sum test on the Fitness (reference = RI3SOME) for 100D case.

	RI3SOME	3SOME		(1+1)-CMAES	
f_1	$\mathbf{7.95e+01 \pm 3.37e-14}$	$\mathbf{7.95e+01 \pm 3.31e-14}$	=	$\mathbf{7.95e+01 \pm 0.00e+00}$	=
f_2	$\mathbf{-2.10e+02 \pm 5.50e-14}$	$\mathbf{-2.10e+02 \pm 5.93e-14}$	=	$-3.79e+01 \pm 5.19e+01$	+
f_3	$-3.41e+02 \pm 2.11e+01$	$\mathbf{-4.38e+02 \pm 7.95e+00}$	-	$1.49e+03 \pm 3.57e+02$	+
f_4	$-3.17e+02 \pm 2.13e+01$	$\mathbf{-4.26e+02 \pm 7.83e+00}$	-	$2.67e+03 \pm 5.19e+02$	+
f_5	$\mathbf{-9.21e+00 \pm 3.64e-12}$	$2.40e+01 \pm 2.33e+01$	+	$3.52e+02 \pm 5.04e+01$	+
f_6	$\mathbf{3.59e+01 \pm 1.72e-07}$	$\mathbf{3.59e+01 \pm 3.79e-08}$	=	$\mathbf{3.59e+01 \pm 2.01e-07}$	=
f_7	$4.22e+02 \pm 1.49e+02$	$5.77e+02 \pm 2.57e+02$	+	$\mathbf{2.26e+02 \pm 4.24e+01}$	-
f_8	$1.91e+02 \pm 3.99e+01$	$\mathbf{1.85e+02 \pm 3.31e+01}$	=	$1.93e+02 \pm 1.80e+01$	=
f_9	$1.74e+02 \pm 1.01e+01$	$1.74e+02 \pm 1.41e+01$	=	$\mathbf{1.73e+02 \pm 1.94e+01}$	=
f_{10}	$2.80e+03 \pm 6.37e+02$	$2.77e+03 \pm 7.75e+02$	=	$\mathbf{1.62e+02 \pm 6.56e+01}$	-
f_{11}	$5.52e+02 \pm 1.12e+02$	$3.74e+02 \pm 8.53e+01$	-	$\mathbf{7.63e+01 \pm 0.00e+00}$	-
f_{12}	$-6.12e+02 \pm 1.18e+01$	$-6.11e+02 \pm 1.83e+01$	=	$\mathbf{-6.19e+02 \pm 2.27e+00}$	-
f_{13}	$\mathbf{3.33e+01 \pm 4.07e+00}$	$3.37e+01 \pm 5.33e+00$	=	$3.34e+01 \pm 4.91e+00$	=
f_{14}	$\mathbf{-5.23e+01 \pm 5.96e-05}$	$\mathbf{-5.23e+01 \pm 6.17e-05}$	=	$\mathbf{-5.23e+01 \pm 2.74e-07}$	-
f_{15}	$3.98e+03 \pm 6.59e+02$	$4.60e+03 \pm 5.47e+02$	+	$\mathbf{3.52e+03 \pm 4.94e+02}$	-
f_{16}	$\mathbf{9.25e+01 \pm 5.70e+00}$	$9.41e+01 \pm 6.01e+00$	=	$9.95e+01 \pm 4.99e+00$	+
f_{17}	$-3.10e+00 \pm 2.75e+00$	$-2.60e-01 \pm 3.41e+00$	+	$\mathbf{-6.32e+00 \pm 2.27e+00}$	-
f_{18}	$3.49e+01 \pm 1.09e+01$	$4.58e+01 \pm 1.62e+01$	+	$\mathbf{2.20e+01 \pm 7.05e+00}$	-
f_{19}	$\mathbf{-9.23e+01 \pm 2.87e+00}$	$-9.08e+01 \pm 3.25e+00$	+	$-8.99e+01 \pm 3.26e+00$	+
f_{20}	$-5.45e+02 \pm 1.26e-01$	$\mathbf{-5.46e+02 \pm 9.34e-02}$	=	$-5.43e+02 \pm 1.19e-01$	+
f_{21}	$5.32e+01 \pm 1.16e+01$	$5.38e+01 \pm 1.37e+01$	=	$\mathbf{4.82e+01 \pm 6.61e+00}$	-
f_{22}	$\mathbf{-9.84e+02 \pm 1.27e+01}$	$\mathbf{-9.84e+02 \pm 1.20e+01}$	=	$\mathbf{-9.84e+02 \pm 1.16e+01}$	=
f_{23}	$8.20e+00 \pm 4.53e-01$	$\mathbf{8.18e+00 \pm 4.44e-01}$	=	$8.86e+00 \pm 4.80e-01$	+
f_{24}	$2.10e+03 \pm 3.66e+02$	$2.71e+03 \pm 5.25e+02$	+	$\mathbf{1.96e+03 \pm 2.41e+02}$	=

gorithmic component to handle non-separability in fitness landscapes. The resulting algorithm is simple, processes only one solution, and is characterized by modest hardware requirements. These features make RI3SOME suitable for embedded implementations. Numerical results on a set composed of diverse optimization problems shows that RI3SOME outperforms in the majority of the cases the original 3SOME structure. This effect is particularly evident for the high dimensional problems taken into account in this work. Finally, RI3SOME appears to be a competitive option also with respect to complex modern algorithms.

Acknowledgments

This research is supported by the Academy of Finland, Akatemiatutkija 130600.

References

[1] N. Hansen, S. D. Müller, and P. Koumoutsakos. Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES). *Evol. Comput.*, 11(1):1–18, 2003.

[2] N. Hansen and A. Ostermeier. Adapting arbitrary normal mutation distributions in evolution strategies: The covariance matrix adaptation. In *Proc. IEEE International Conference on Evolutionary Computation*, pages 312–317, 1996.

[3] N. Hansen and A. Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evol. Comput.*, 9(2):159–195, 2001.

[4] G. Iacca, F. Neri, E. Mininno, Y. S. Ong, and M. H. Lim. Ockham's razor in memetic computing: Three stage optimal memetic exploration. *Inform. Sciences*, 188(1):17–43, 2012.

[5] C. Igel, T. Suttorp, and N. Hansen. A computational efficient covariance matrix update and a (1+1)-CMA for evolution strategies. In *Proc. Genetic and Evolutionary Computation Conference*, pages 453–460, 2006.

[6] N. Hansen, S. Finck, R. Ros, and A. Auger. Real-parameter black-box optimization benchmarking 2010: Noiseless functions definitions. Technical Report RR-6829, INRIA, 2010.

[7] F. Neri and C. Cotta. Memetic algorithms and memetic computing optimization: A literature review. *Swarm and Evolutionary Computation*, 2(1):1–14, 2012.

[8] F. Neri, C. Cotta, and P. Moscato. *Handbook of Memetic Algorithms*, volume 379 of *Studies in Computational Intelligence*. Springer, 2012.

[9] F. Neri, G. Iacca, and E. Mininno. Disturbed exploitation compact differential evolution for limited memory optimization problems. *Inform. Sciences*, 181(12):2469–2487, 2011.

[10] K. Price. An introduction to differential evolution. In David Corne, Marco Dorigo, Fred Glover, Dipankar Dasgupta, Pablo Moscato, Riccardo Poli, and Kenneth V. Price, editors, *New Ideas in Optimization*, pages 79–108. McGraw-Hill, 1999.

[11] T. Takahama and S. Sakai. Solving nonlinear optimization problems by differential evolution with a rotation-invariant crossover operation using gram-schmidt process. In *Proc. World Congress on Nature and Biologically Inspired Computing*, pages 533–540, 2010.

[12] T. Takahama and S. Sakai. Efficient nonlinear optimization by differential evolution with a rotation-invariant local sampling operation. In *Proc. IEEE Congress on Evolutionary Computation*, pages 2215–2222, 2011.

[13] L. Y. Tseng and C. Chen. Multiple trajectory search for large scale global optimization. In *Proc. IEEE Congress on Evolutionary Computation*, pages 3052–3059, 2008.

[14] D. Whitley, S. Rana, and R. B. Heckendorn. The island model genetic algorithm: On separability, population size and convergence. *Journal of Computing and Information Technology*, 6(3):33–47, 1998.

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

A MULTIDIRECTIONAL MODIFIED *PHYSARUM* SOLVER FOR OPTIMAL DISCRETE DECISION MAKING

Luca Masi, Massimiliano Vasile
Department of Mechanical & Aerospace Engineering, University of Strathclyde, Glasgow, UK
{luca.masi; massimiliano.vasile}@strath.ac.uk

Abstract In this paper, a bio-inspired algorithm able to incrementally grow decision graphs in multiple directions is presented. The heuristic draws inspiration from the behaviour of the slime mould *Physarum Polycephalum*. In its main vegetative state, the *plasmodium*, this large single-celled amoeboid organism extends and optimizes a net of veins looking for food. The algorithm is here used to solve classical problems in operations research (symmetric Traveling Salesman and Vehicle Routing Problems). Simulations on selected test cases demonstrate that a multidirectional modified *Physarum* solver performs better than a unidirectional one. The ability to evaluate decisions from multiple directions enhances the performance of the solver in the construction and selection of optimal decision sequences.

Keywords: Discrete Optimization, Multidirectional, *Physarum*.

1. Introduction

Over the past two decades, bio-inspired computation has become an appealing topic to solve NP-hard problems in combinatorial optimization using a reasonable computational time [3]. Ant Colony Systems, Genetic Algorithms and Particle Swarms are examples [6]. The method proposed in this paper takes inspiration from *Physarum Polycephalum*, an organism that was endowed by nature with heuristics that can be used to solve discrete decision making problems. It has been shown that *Physarum Polycephalum* is able to connect two food sources by solving a maze, using morphing properties [7]. In [1] and [9] it has been shown that a living *Physarum* is able to reproduce man-made transportation networks, i.e. Mexican highway network and Japan rail network, re-

spectively. *Physarum* based algorithms have been developed recently to solve multi-source problems with a simple geometry [5, 11], mazes [8] and transport network problems [8, 9]. In this paper a multidirectional modified *Physarum Polycephalum* algorithm able to solve NP-hard discrete decision making problems is presented. In the mathematical model proposed in Section 2, discrete decision making problems are modeled with decision graphs where nodes represent the possible decisions while arcs represent the cost associated with decisions. Decision graphs are incrementally grown and explored in multiple directions using the *Physarum*-based heuristic. This paper aims at proving that a multidirectional incremental *Physarum* solver is more efficient, in terms of success rate (see Section 3.2), than a unidirectional incremental *Physarum* solver when applied to the solution of decision problems that can be represented with directed symmetric decision graphs, i.e. graphs where the contribution of an arc to a complete path can be evaluated moving forward or backward along the graph. This thesis will be demonstrated in Section 4 solving a series of test cases. Symmetric Traveling Salesman and Vehicle Routing Problems (known with the acronyms TSP and VRP), introduced in Section 3, were chosen as representative examples of the above type of decision making problems, here called reversible decision-making problems, i.e. problems in which a decision can be taken either moving forward or backward along the graph, as explained in Section 2.

2. Biology and Mathematical Modeling

Physarum Polycephalum is a large, single-celled amoeboid organism that exhibits intelligent plant-like and animal-like characteristics. Its main vegetative state, the *plasmodium*, is formed of a network of veins (*pseudopodia*). The stream in these tubes is both a carrier of chemical and physical signals, and a supply network of nutrients throughout the organism [11]. *Physarum* searches for food by extending this net of veins, whose flux is incremented or decremented depending on the food position with reference to its centre.

2.1 Mathematical Modeling and Multiple Direction Growing Decision *Physarum* Graphs

Reversible decision making problems. Assuming that a discrete decision problem is reversible, i.e. for each decision that induces a change from state a to state b it is possible to evaluate a symmetric inverted decision that brings back from b to a, it can be modeled with a symmetric directed graph. Here the interest is for general graphs in which the cost $C(a, b)$ associated with the decision a to b may not be identical

to the cost $C(b, a)$, but the decision that brings from b to a exists and can be evaluated. The symmetric directed graph can be seen as the superposition of two directed graphs (direct-flow, *DF*, and back-flow, *BF*, graphs) whose nodes are coincident and edges have opposite orientation. In so doing, the decision between state a and b has a forward link a to b and a superposed backward link b to a. It is assumed that the first decision node is the heart of a growing *Physarum* in *DF*, and the end decision node the heart of a growing *Physarum* in *BF*. The two *Physarum* are supposed able to incrementally grow the decision graph in the two directions by extending their net of veins. The result is a graph where both nodes and links are incrementally built by two expanding *Physarum*.

Multidirectional incremental modified Physarum algorithmic. The flux through the net of *Physarum* veins can be modeled as a classical Hagen-Poiseuille flow in cylindrical ducts with diameter variable with time [5, 8, 9, 11]:

$$Q_{ij} = \frac{\pi r_{ij}^4}{8\mu} \frac{\Delta p_{ij}}{L_{ij}}, \tag{1}$$

where Q_{ij} is the flux between i and j, μ is the dynamic viscosity, Δp_{ij} the pressure gradient, L_{ij} the length and r_{ij} the radius. Following [5], these two last main parameters are taken into account in the algorithm. Diameter variations allow a change in the flux. Veins' dilation due to an increasing number of nutrients flowing can be modeled using a monotonic function of the flux. In the present work, a function linear with respect to the product between the radius r_{ij} of the veins traversed by nutrients, and the inverse of the total length L_{tot} traveled, will be used for the veins' dilation:

$$\left.\frac{d}{dt} r_{ij}\right|_{dilation} = m \frac{r_{ij}}{L_{tot}}, \tag{2}$$

where the coefficient m is here called linear dilation coefficient. Veins' contraction due to evaporative effect can be assumed to be linear with radius:

$$\left.\frac{d}{dt} r_{ij}\right|_{contraction} = -\rho r_{ij}, \tag{3}$$

where $\rho \in [0, 1]$ is defined evaporation coefficient. The probability associated with each vein connecting i and j is then computed using a simple adjacency probability matrix based on fluxes:

$$P_{ij} = \begin{cases} \frac{Q_{ij}}{\sum_{j \in N_i} Q_{ij}} & if\ j \in N_i \\ 0 & if\ j \notin N_i \end{cases} \tag{4}$$

Algorithm 1 Multidirectional incremental modified Physarum solver

```
initialize m, ρ, GF_ini, N_agents, p_ram, α, r_ini
generate a random route from start to destination in DF and BF
for each generation do
  for each virtual agent in all directions (DF and BF) do
    if current node ≠ end node then
      if rand ≤ p_ram then
        using Eq. (6) create a new path, building missing links and nodes
      else
        move on existing graph using Eq. (4).
      end if
    end if
  end for
  look for possible matchings
  contract and dilate veins using Eqs. (2), (3) for each agent, (5)
  update fluxes and probabilities using Eqs. (1), (4)
end for
```

where N_i is the set of neighbour for i.

A further social term in the dilation process was added in the algorithm and takes inspiration from the behavior of the *Dictyostelium Discoideum*'s pacemaker amoeba [6]:

$$\frac{d}{dt} r_{ij_{best}} \bigg|_{elasticity} = GF r_{ij_{best}}, \tag{5}$$

where GF is the growth factor of the best chain of veins and $r_{ij_{best}}$ the veins' radii. This is an additive term in the veins' dilation process, whose first main term is expressed in Eq. (2). The incremental growth of decision network in multiple directions is then based on a weighted roulette. Nutrients inside veins are interpreted as virtual agents that move in accord with adjacency probability matrix in Eq. (4). Once a node is selected, there is an *a priori* probability p_{ram} of ramification towards new nodes that are not yet connected with the actual node. The probability of a new link construction from the current node c to a new possible node $n_i \in N$, where N is the set of new possible decisions, is here assumed to be inversely proportional to the cost L_{cn_i} of the decision between c and n_i, i.e. the length:

$$p_{cn_i} \propto \frac{1}{L^{\alpha}_{cn_i}}, \tag{6}$$

where α is a weight. Once a new link is built, a complete decision path is constructed (creating other links if necessary). Assuming then one DF and one superposed BF *Physarum*, a matching condition can be

defined. If an arc connecting two nodes that belong to *DF* and *BF* *Physarum* respectively, exists or can be created, it is traversed by the agent and becomes part of both the *DF* and the *BF*. Several types of matching strategies are possible: *all-matching*, in which all joint paths are saved and evaluated, *random-matching*, in which some joint paths are selected among all the possible joint paths with a probabilistic rule and *selective-matching*, in which some joint paths are selected among all the possible joint paths according to an elitist criterion. At each generation, the elitist criterion selects a joint path if and only if it is better in terms of total cost than the previous joint paths selected during the same generation. The pseudocode of the multidirectional incremental modified *Physarum* solver is provided in Algorithm 1, where r_{ini} is the initial radius of the veins, always sets equals to 1 in this paper, and N_{agents} the number of virtual agents. A unidirectional algorithm is a special case of multidirectional algorithm, obtained by freezing the *BF*: flux and graph growth are allowed in only one direction.

Simulations on selected test cases (see Section 4) were carried out adding in the *Physarum* software a routine for the adaptive control of the growth factor GF. This control was introduced in order to incrementally boost the effect of GF during a simulation, driving exploring agents towards best veins. Simulations showed that the adaptive control of GF helps the convergence of the algorithm towards optimal solution. Given an initial value for the growth factor GF_{ini}, GF is incremented by a fixed percentage $\sigma = 0.01$ after every generation. If the probability p_{best} associated with the best path so far (see adjacency probability matrix, Eq. (4)) is higher than a fixed value $p_{lim}^{low} = 10^{-4}$, the increment is set to zero. Then, if p_{best} exceeds a value $p_{lim}^{high} = 0.85$, GF is set equal to GF_{ini} and veins are dilated and contracted to their initial value.

3. Application to Traveling Salesman and Vehicle Routing Problems and Benchmark

Traveling Salesman Problem (TSP) and Vehicle Routing Problem (VRP) are classical problems in combinatorial optimization. When the modified *Physarum* algorithm is applied to VRP, a probability skew factor $\psi = 0.5$ is included in the algorithm. If an agent is not obliged to go to depot, the probability to reach the depot is lowered of a factor $(1 - \psi)$. The skew factor is introduced in the model to avoid frequent returns to depot.

Algorithm 2 Testing procedure

set to N the max number of function evaluations for the algorithm P
apply P to p for n times and set $j_s = 0$
for all $i \in [1, ..., n]$ **do**
  if $f_p^{best} + tol \leq f_i^{best} \leq f_p^{best} - tol$ **then**
    $j_s = j_s + 1$
  end if
end for
evaluate $p_s = j_s/n$

3.1 Benchmark

TSPLIB [12] was used to benchmark the proposed *Physarum* algorithm, developed in Matlab® R2010b, on the TSP problem. In Section 4 are reported the results obtained by applying the multidirectional solver to test case *Ulysses16*, normalised with a factor equals to 10000, and to test case *Eil51*. The reference optimal solutions for this particular TSP instance can be found in the TSPLIB. The *Physarum* solver applied to VRP was tested on a map of 9 cities plus one depot. The map is built using 9 Italian cities (Firenze, Livorno, Montecatini, Pistoia, Prato, Montevarchi, Arezzo, Siena, San Gimignano), with a city considered the depot (Ponsacco). The Euclidean distance in kilometers was used. Optimal tour with constant demand for each city equals to 1 and one vehicle with capacity equals to 4, was found using an exhaustive search.

3.2 Testing Procedure

The testing procedure proposed in [13] was used in this paper and is reported in Algorithm 2. The success rate $p_s = j_s/n$, expressing the percentage of success after N function evaluations of algorithm P over a group of n simulations, will be used for the comparative assessment of the algorithm's performance. A function evaluation is defined as the call to the objective function, i.e. each arc selected by the virtual exploring agents (see Section 2.1) is considered a function evaluation. For example, each exploring agents during a generation would call the objective function a number of time proportional to the dimension of the problem, i.e. 16 times for *Ulisses16* test case and 51 times for *Eil51* test case (see Section 3.1). The number of calls for an agent during a generation have to be then multiplied by the total number of exploring agents and doubled if agents move in two directions, in order to have an estimate of total calls during a complete generation. In [13] one can find a full explanation on the setting of the value of n in order to have a reliable

estimate of algorithm's success probability. Using $n = 175$ one would obtain an error ≤ 0.05 with a 95% of confidence. A value equals to 200 for n is used in the simulations presented in this paper. Tolerance is set to 10^{-8}.

4. Results

The multidirectional modified *Physarum* solver, named D&B in the following, was compared against a unidirectional modified *Physarum* solver, named D. Simulations were carried out on a 64-bit OS Windows 7 Intel® Core™2 Duo CPU E8500 3.16. D&B was tested without and with matching ability (see Section 2.1), named D&B with xM. The prefix x indicates the type of matching selected: nM means *no-matching*, rM *random-matching*, aM *all-matching* and sM indicates *selective-matching*.

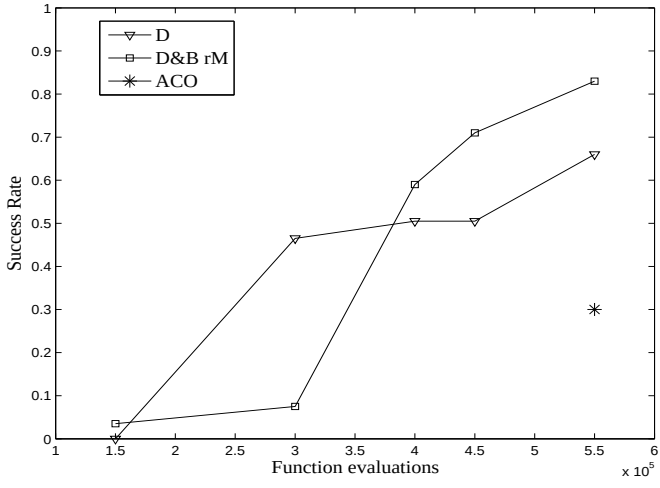


Figure 1. Variation of the success rate with the number of function evaluations - TSP test case *Ulysses16*.

The multidirectional and unidirectional modified *Physarum* algorithms were applied to the symmetric TSP test case *Ulysses16*, described in Section 3.1. The values used as input parameters in the simulations are $m = 5 \times 10^{-5}$, $\rho = 1 \times 10^{-5}$, $GF_{ini} = 5 \times 10^{-3}$, $N_{agents} = 100$, $p_{ram} = 0.8$, $\alpha = 0$, $r_{ini} = 1$, and were chosen after a series of trials. Results, as shown in Fig. 1, demonstrate that the multidirectional modified *Physarum* algorithm with matching ability (D&B with rM) provides higher success rate than the unidirectional modified *Physarum* algorithm (D). Although D

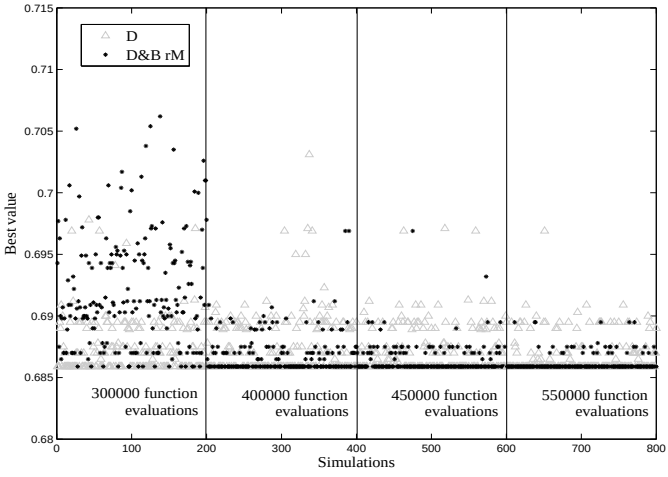


Figure 2. Best solution found by D and D&B with rM on groups of 200 runs for different function evaluation limits - TSP test case *Ulysses16* - best value = 0.6859.

performs better at 300000 function evaluations (success rate is near 0.5 whilst success rate of D&B with rM is under 0.1), increasing the number of function evaluations the performance of multidirectional exceeds the performance of direct only solver. At 550000 function evaluations the success rate of D&B with rM reaches 0.83, against the 0.66 of D. Figure 1 shows also the performance of a simple Ant Colony Optimization solver (ACO [10]) at 550000 function evaluations. This solver is the implementation of the algorithm provided in [4]. Results show that D&B and D&B with rM have a good performance (success rate are respectively 0.66 and 0.83) if compared with ACO solver (success rate is 0.30). This comparison was made in order to check if the required function evaluations for the *Physarum* solver were comparable, in terms of order of magnitude, with a simple ACO solver. This comparison was only a preliminary verification and did not absolutely aim at demonstrating that the modified *Physarum* solver performs better than a general ACO solver. Comparison with more advanced ACO solvers are currently under way. Following same logic, a preliminary comparison with genetic algorithm (GA) was made for the test case *Ulysses16*. The GA from Matlab® R2010b global optimization toolbox, customized to solve the TSP, showed a success rate equals to 0.27 at 550000 function evaluations.

Figure 2 shows a comparison among the best solutions for each of the 200 runs at 300000, 400000, 450000, 550000 function evaluations.

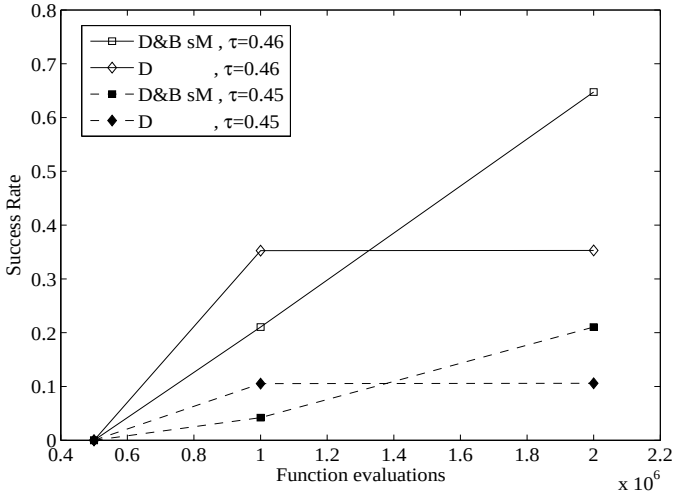


Figure 3. Variation of the success rate with the number of function evaluations - TSP test case *Eil51*.

The best solution is known to be 0.6859 (TSPLIB [12]). Analyzing the figure, the D&B with rM algorithm (black dots) has an output comprised in a narrower band of values if compared with D algorithm (grey dots) increasing the number of function evaluations. At 300000 function evaluations there is a 9% probability to reach the exact value, a 71% probability to reach a value in the band $(0.6859, 0.6978]$ and 20% probability to reach a value in the band $(0.6978, 0.7070]$ using D&B with rM, while, using D, there is a 48% probability to reach the exact value, a 52% to reach a value between $(0.6859, 0.6978]$ and a 0% probability to reach a value in the band $(0.6978, 0.7070]$. At 550000 function evaluations the probability of exact value outcome using D&B with rM rise up to 83% with a 17% probability to reach a value in the band $(0.6859, 0.6950]$, whilst there is only a 66% probability using D, with a 27% probability to reach a value in the band $(0.6859, 0.6950]$ and a 7% in the band $(0.6950, 0.6978]$. It is evident the fastest convergence towards exact solution of black dots increasing the number of function evaluations.

The multidirectional and unidirectional modified *Physarum* algorithms were also applied to the symmetric TSP test case *Eil51*, described in Section 3.1. The values used as input parameters in the simulations are $m = 5 \times 10^{-3}$, $\rho = 1 \times 10^{-6}$, $GF_{ini} = 5 \times 10^{-4}$, $N_{agents} = 100$, $p_{ram} = 0.9$, $\alpha = 1$, $r_{ini} = 1$, and were chosen after a series of trials.No GF adaptive routine was used. Results, as shown in Fig. 3 where

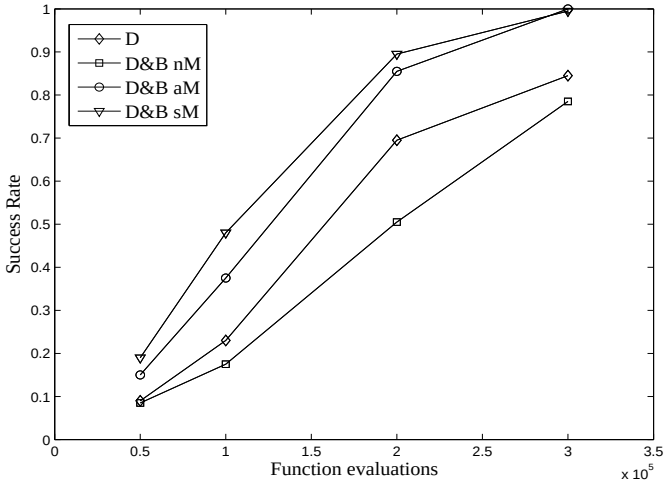


Figure 4. Variation of the success rate with the number of function evaluations - VRP test case.

$\tau = f_i^{best}$ (see Algorithm 2) represents the value of the threshold chosen for the success rate evaluation, demonstrate again that the multidirectional modified *Physarum* algorithm with matching ability (D&B with sM) provides higher success rate than the unidirectional modified *Physarum* algorithm (D). With $\tau = 0.46$ and $\tau = 0.45$, the gain in success rate using a multidirectional approach reaches up to 20% at 5500000 function evaluations.

Finally, the multidirectional and unidirectional modified *Physarum* algorithms were applied to the VRP test case described in Section 3.1. The values used as input parameters in the simulations are $m = 1 \times 10^{-5}$, $\rho = 1 \times 10^{-5}$, $GF_{ini} = 5 \times 10^{-3}$, $N_{agents} = 150$, $p_{ram} = 0.9$, $\alpha = 0$, $r_{ini} = 1$, and were chosen after a series of trials. The success rate is reported in Fig. 4. Results demonstrate that the multidirectional algorithm with matching ability (D&B with sM and D&B with aM) performs better than the unidirectional algorithm (D). After analysing more in depth the results and considering the mathematical modeling in Section 2.1, one could argue that:

I. At 300000 function evaluations the gain in success rate using D&B with sM or aM instead of D is around 20%.

II. The performance of D&B with sM and D&B with aM are almost comparable. The real gain using D&B with sM is the saving in computational time. The mean computational time in the simulations (200

runs) is reduced considerably (between 33% and 37%) using a selective matching instead of a non selective matching. This is due to the selection of best joint decision sequence at each generation: discarding worst joint decision sequences the updating of veins is done only for the best sequences.

III. The multidirectional algorithm without matching ability (D&B with nM) has the worst performance compared to others in all the cases. This is due to lack of communications between the two *Physarum*.

Simulations on bigger TSP and VRP problems (50 to more than 100 cities) are currently under way. Due to the fact that an higher number of generations are necessary to solve higher dimensional problems, the adaptive control of GF (see Section 2.1) may cause an uncontrolled growth of the flux inside few veins (it should be noted that flux is related to the fourth power of the radius, see Eq. (1)). A good solution is to set $\sigma = 0$ when dealing with high dimensional problems.

4.1 Conclusion

This paper proposed a multidirectional incremental modified *Physarum* solver for discrete decision making problems. The algorithm showed the ability to solve simple symmetric TSP and VRP problems, selected as representative examples of reversible decision making problems. Simulations on selected test cases proved that a multidirectional approach with matching ability performs better than a unidirectional one when applied to reversible discrete decision making problems. The possibility for the two *Physarum* to evaluate each step of the decision sequence from multiple directions and create joint paths enhances the performance of the solver: this forward and backward decision making process increased the gain in success rate up to 20% during selected simulations. Furthermore simulations on TSP selected test cases showed a good performance of multidirectional incremental modified *Physarum* solver if compared to simple ACO and GA solvers. This preliminary comparison was made in order to check if the required function evaluations for the *Physarum* solver were comparable, in terms of order of magnitude, with simple ACO and GA solver and did not absolutely aim at demonstrating that the modified *Physarum* solver performs better than general ACO and GA solvers. Comparison with more advanced solvers as well as simulations on bigger TSP and VRP problems (50 to more than 100 cities) are currently under way. After these last series of tests, the multidirectional incremental *Physarum* solver will be applied to other challenging decisional problems with a special focus on aerospace engineering.

References

[1] A. Adamatzky, G. J. Martinez, S. V. Chapa-Vergara, R. Asomoza-Palacio, and C. R. Stephens. Approximating Mexican highways with slime mould. *Nat. Comput.*, 10(3), 1195–1214, 2011.

[2] N. R. Burns, M. D. Lee, and D. Vickers. Are individual differences in performance on perceptual and cognitive optimization problems determined by general intelligence?. *The Journal of Problem Solving* 1(1):5–19, 2006.

[3] M. Dorigo and L.M. Gambardella. Ant colonies for the travelling salesman problem. *Biosystems*, 43(2):73–81, 1997.

[4] M. Dorigo, V. Maniezzo, and A. olorni. The ant system: Optimization by a colony of cooperating agents. *IEEE T. Syst. Man Cyb.*, 26(1):29–41,1996.

[5] D. S. Hickey and L. A. Noriega. Insights into information processing by the single cell slime mold *Physarum Polycephalum*. In *Proc. UKACC International Conference on Control*, 2008.

[6] D. R. Monismith Jr. and B. E. Mayfield. Slime mold as a model for numerical optimization. In *Proc. IEEE Swarm Intelligence Symposium*, 2008.

[7] T. Nakagaki, H. Yamada, and A. Toth. Maze-solving by an amoeboid organism. *Nature*, 407:470, 2000.

[8] A. Tero, R. Kobayashi, and T. Nakagaki. *Physarum* solver: a biologically inspired method of road-network navigation. *Physica A*, 363(1):115–119, 2006.

[9] A. Tero, S. Takagi, T. Saigusa, K. Ito, D. P. Bebber, M. D. Fricker, K. Yumiki, R. Kobayashi, and T. Nakagaki. Rules for biologically inspired adaptive network design, *Science*, 439:327, 2010.

[10] H. Wang. Solving Symmetrical and DisSymmetrical TSP based on Ant Colony Algorithm. Retrieved from MATLAB® CENTRAL, `http://www.mathworks.com/matlabcentral/fileexchange/14543`.

[11] A. Tero, K. Yumiki, R. Kobayashi, T. Saigusa, and T. Nakagaki. Flow-network adaptation in *Physarum* Amoebae. *Theor. Biosci.*, 127(2):89–94, 2008.

[12] TSPLIB, library of instances for travelling salesman and vehicle routing problems. Ruprecht Karls Universitaet Heidelberg, `http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/`.

[13] M. Vasile, E. Minisci, and M. Locatelli. An inflationary differential evolution algorithm for space trajectory optimization. *IEEE T. Evolut. Comput.*, 2(2):267–281, 2011.

III

APPLICATIONS

AN ANT COLONY ALGORITHM FOR RECYCLING WASTE COLLECTION

Roxanne T. Evering
Graduate Consultant
roxanne.t.evering@bath.edu

Martin B. Reed
Department of Mathematical Sciences, University of Bath, UK
m.b.reed@bath.ac.uk

Abstract We describe a strategy for solving the Capacitated Vehicle Routing Problem for domestic recycling waste collection, using an Ant Colony System heuristic. Improved results are obtained by incorporating monitoring of vehicle load within the ACS, to automatically insert unloading trips to the depot into the vehicle route.

Keywords: Ant colony optimisation, Travelling salesman, Vehicle routing.

1. Introduction

Almost every Local Authority (LA) in the UK provides a weekly or fortnightly kerbside collection service for domestic recycling waste. The LA or its contractor needs to determine the number of vehicles required in its fleet, and the collection route to be followed by each vehicle, which minimise the total cost while meeting the total customer demand within the working day.

In its simplest form, where a single vehicle is used to collect waste from N households, starting from the depot and returning at the end of the day, the problem reduces to the classic Travelling Salesman Problem (TSP). When the total customer demand exceeds the capacity of a single vehicle, our approach is to first determine the number of vehicles of each type required (the Fleet Size Problem). Then the task of allocating households to vehicles and planning the route taken by each vehicle in servicing its customers, comprises a Vehicle Routing Problem (VRP).

Because of the fixed maintenance and staffing costs per vehicle, it is preferable to find a solution involving the minimum number of vehicles consistent with all customer demand being met within the working day, and involving trips back to the depot/tip to unload when full, before resuming their route.

The present work is from an MSc project [8] performed for Eunomia Research and Consulting, an environmental consultancy company based in Bristol UK.

The Fleet Size Problem. To solve the fleet size problem we use an Integer Linear Programming (ILP) formulation. The model assumes a fixed number K of vehicles, and so the algorithm is run repeatedly with increasing values of K until a feasible solution is found. For the cost function we use the total time spent by vehicles on unloading trips to the depot. This model is based on the Linear Programming (LP) models of [9, 13]; further details including the constraints used are in [8].

In a solution the program will output the number of unloading trips made, and the number of households to be serviced, by each vehicle.

2. The Vehicle Routing Problem

The Vehicle Routing Problem (VRP) is a combinatorial optimisation and integer programming problem which consists of designing the optimal set of routes for a fleet of vehicles in order to serve a given set of customers, with vehicles travelling via minimum-cost routes, originating and terminating at a depot. The problem is defined on a graph, with a customer demand specified at each node and a travel cost or driving distance associated with each arc. There are many extensions of the classic VRP, in particular the Capacitated VRP (CVRP) where each vehicle has a limited capacity.

Sahoo et al. [13] describe a commercial software package they developed for the waste collection problem, treating it as a CVRP. The objective function to be minimised is the total travelling time of all vehicles. They implemented an iterative two-phase algorithm, first finding a feasible solution by partitioning the nodes into clusters, and then improving it using an insertion algorithm combined with a Simulated Annealing metaheuristic.

3. Ant Colony Optimisation

Another metaheuristic which has been used for the VRP is Ant Colony Optimisation (ACO). ACO uses artificial ants (agents) which mimic the behaviour of natural ants in choosing to travel along paths with a higher

pheromone level (i.e. those most often used). The two main phases of the ACO algorithm are the ants' route construction and the pheromone update. In the tour construction phase, M ants concurrently build tours beginning from randomly chosen nodes. At each construction step, ant k currently at node i applies a probabilistic random proportional rule to decide which node to go to next. It selects the move to expand its tour by taking into account the following two values:

- The heuristic function η_{ij} which represents the attractiveness of the move, usually calculated as the inverse of the distance/cost on the arc from node i to node j.
- The level of pheromone on the arc (i, j), denoted τ_{ij}, which indicates how useful it has been in the past to traverse this particular arc.

Given these parameters, the probability with which the ant chooses to go to node j next is

$$p_{ij}^k = \frac{(\tau_{ij})^\alpha (\eta_{ij})^\beta}{\sum_{l \in \mathcal{N}_i^k} (\tau_{il})^\alpha (\eta_{il})^\beta} \tag{1}$$

if node $j \in \mathcal{N}_i^k$, and $p_{ij}^k = 0$ otherwise. Here, $\mathcal{N}_i^k$ is the feasible neighbourhood (i.e. the nodes which are directly accessible from node i and not previously visited), and α and β are heuristic parameters. Each ant maintains a memory of the nodes already visited. Once all ants have completed a tour, the pheromone trails are updated. This is done by first lowering the pheromone levels on all arcs (to represent evaporation and in order to progressively forget bad solutions) and then adding pheromone to the arcs that have been traversed.

3.1 The Ant Colony System (ACS)

This method improves on the ACO in the following main aspects:

- **Route Construction:** During tour construction, ant k located at node i, moves to node j chosen according to the following pseudorandom proportional rule. A random variable q uniformly distributed over $[0, 1]$ is evaluated, and if $q > q_0$ the node j is chosen according to the standard ACO rule (1), using $\alpha = 1$. Otherwise, choose j by

  $$j = \arg\max_{j \in \mathcal{N}_i^k} \{(\tau_{ij})(\eta_{ij})^\beta\}. \tag{2}$$

  So, with probability q_0, the ant makes the best move described by the pheromone trails and heuristic information (exploiting the

learned knowledge), while with probability $(1 - q_0)$ it performs a biased exploration of the arcs.

- **Pheromone Update:** The ACS method uses two types of pheromone updates; global and local. The local update is performed every time an ant traverses an arc and the pheromone is modified as follows:

$$\tau_{ij} \leftarrow (1 - \xi)\tau_{ij} + \xi\tau_0, \tag{3}$$

where $0 < \xi < 1$ and τ_0 is the initial pheromone value defined as $\tau_0 = (nL^{nn})^{-1}$ where n is the number of nodes to be seen, and L^{nn} is the length of the nearest neighbour tour (a tour in which each move is to the nearest unvisited node; this is used as a baseline tour length). The global update, however, is only carried out by the ant that produced the best tour so far and is implemented by the following equation:

$$\tau_{ij} \leftarrow (1 - \rho)\tau_{ij} + \rho\Delta\tau_{ij}^{bs}, \quad \forall(i, j) \in T^{bs}, \tag{4}$$

where $\Delta\tau_{ij}^{bs} = (L^{bs})^{-1}$, ρ is a parameter governing decay and T^{bs} is the best found tour so far with L^{bs} its length. This enables the algorithm to converge faster by directly concentrating the search around the best tour.

The ACS method has proved to be very efficient in solving VRPs [12].

3.2 Previous Applications

An ACO algorithm was directly applied to the problem for best route identification for urban solid waste collection in [10], although the problem was transformed to a TSP for simplicity. Thus, the region was initially 'fragmented into a series of sub-areas which produced a quantity of solid waste, equal to or less than a fixed quantity', thereby seeking to ensure that vehicle capacity was not exceeded. A later paper [1] again directly applies ACO to the problem of urban waste collection, modelling the problem as a Capacitated Arc Routing Problem (CARP).

4. Proposed Solution Method

As in papers [4, 5] by Dorigo et al., we have chosen to implement an ACS since this method has shown promising results when applied to the TSP.

The parameter values used in our algorithm were taken from [3, 6] and have been used in [4, 5] which both report optimal solutions. These are: $\beta = 2$, $q_0 = 0.9$, $\xi = \rho = 0.1$ and $\alpha = 1$. The total number

of ants M used within an ACS is another important parameter. It is suggested in [3] that an ant should be placed at each node; however, in applications of ACS to TSPs M is usually set to be 10. The ACS was programmed in MATLAB, and was tested on a selection of the freely available TSP examples at TSPLIB (`http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95`) and in all cases produced the optimal solution, although in the larger examples (of over 500 nodes) required large numbers of iterations to do so.

Description. The suggested solution approach is a cluster first, route second method which implements an ACS to solve the VRP. This strategy is described in more detail in [7] and is used in [2, 11] to solve a routing problem, reporting successful results. The solution method has five stages:

i. Using the information on recycling demand within a Local Authority, the Fleet Size Problem is solved by Integer Linear Programming. This produces the minimum number of vehicles required, together with the number of households each vehicle can see in the working day, and the number of unloading trips which it will need to make, and hence the total number of tours needed to cover all households.

ii. Using geographic data, the households need to be grouped into 'macro-points', so that the problem can be modelled as a CVRP. The macro-points form the nodes of the graph, connected by arcs, with a driving distance associated with each arc. Each node has a recycling weight and volume demand (found by aggregating the individual household demands, factored by the setting-out rate) and a service time (summing the times to collect from and to travel between households). We have taken this data from the collection zones used by Eunomia in its current database.

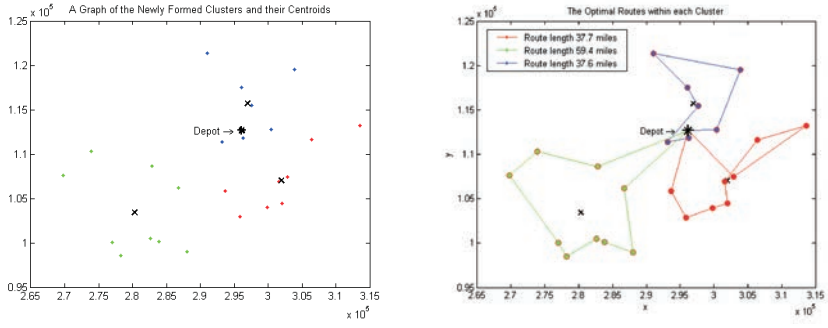
iii. The nodes are next grouped into the requisite number of clusters, each cluster forming a set of nodes to be visited on a tour. For the clustering algorithm we have used Euclidean coordinates, obtained from latitude/longitude of each zone. The depot node is added to each cluster. We employed the in-built k-means clustering algorithm within MATLAB.

iv. A minimum-length tour around each cluster is determined using the ACS algorithm.

v. The final stage of the solution is to assign each tour to a particular vehicle, checking that the total weight and volume demands of

the nodes in each cluster do not exceed the vehicle capacity, and to then schedule the tours such that each vehicle will be able to complete its tours within the working day.

Results. We will present results to illustrate the final two stages of the solution method, using data for one of the Local Authorities which Eunomia has worked with in the past. Unfortunately the database is too coarse-grained for a realistic test of the method – a minimum of 10 vehicles would be needed to meet all demand, but the LA is divided into 24 collection zones, so using this data each tour would contain only 2 or 3 nodes. Instead, we reduce the nodal demands so that the total can be collected in 3 tours, treating each zone as a node. Initially the ants are placed randomly at nodes to be included in the tour. When they have visited all the other nodes (including the depot node) they return to their starting-point to complete the tour. The tour is then considered as starting and finishing at the depot node.



(a) (b)

Figure 1. Graphs of (a) the 3 clusters and (b) their optimal tour.

The three clusters (with their centroids) and the optimal tours found using the ACS are displayed in Fig. 1. The optimal tours are of combined length 134.7 miles, and were found by running the ACS on each cluster in turn. The three locally-optimal tours were obtained after 52, 92 and 7 iterations respectively.

In real-life situations, the fleet size will contain ten or more vehicles, so with unloading trips there will need to be 30–40 clusters. At this scale, Stage (v) of the method becomes a significant load-balancing issue. Although the Fleet Size Problem solution demonstrates that a feasible solution exists, there is no guarantee that we will be able to achieve this from the clusters formed in Stage (iii). The underlying cause of

this problem is that the clustering/routing algorithm does not take any account of the demands at the individual nodes. In the next section we propose a way of extending the ACS algorithm to address this issue.

5. An ACS Algorithm incorporating Unloading Trips

Note that the Fleet Size Problem solution will produce not just the number of vehicles, but also the maximum number of households each vehicle can serve and the number of unloading trips (known as 'tips') it will make within a day; this information can be used in an improved algorithm. In this, the algorithm keeps track of the amount of waste collected along the tour, and includes a trip to the depot to unload whenever necessary. The algorithm can thus optimise the whole day's schedule for a single vehicle, so the number of clusters is equal to the number of vehicles in the fleet, rather than the number of tours.

This idea, which aims to exploit the ants' ability to store multiple sources of information regarding the current tour built, is relatively unexplored. Most reports in the literature tackle the CVRP by implementing an heuristic savings algorithm, which combines customers into tours following a greedy strategy. An ACO algorithm with a path scanning heuristic involving vehicle capacity was described in [1], solving a CARP for waste collection in an urban environment.

Our program uses an additional input vector recording the amount of waste to be collected at each node, and so within the ants' memory structures, we include a vector containing the total amount of recycling waste collected along the route.

Programming. Let the graph involve N nodes where waste is to be collected, together with the depot as node $N + 1$. The depot node has zero waste demand, and is not included in the path node selection algorithm in Sec. 3.1. Because of the increased length of the route to be constructed, compared to the previous cluster-based algorithm, we have used N ants, one placed at each node initially. The tour matrix is expanded to allow space in the tour for all the collection nodes plus the required number of unloading trips to the depot node. We have worked with volume demand and capacity, although weight collected could easily be included as a parallel data calculation. The tour length for each ant is initialised as the distance from the depot to its starting node. A vector storing the waste volume collected is initialised as the demand at the starting node.

As an ant moves from node to node, the volume of waste collected is updated. When the program selects a new node to move to, it checks that collection from that node will not exceed the vehicle capacity; if it will, the node is rejected and another selected. If no collection nodes can be chosen without exceeding capacity, the ant moves instead to the depot node to unload. The distance of this arc is added to the tour length, and the ant's waste volume is reset to zero.

To resume its tour from the depot node, the ant selects at random an unvisited node to move to, and the distance from depot to node is added to the tour length. When all nodes have been visited the ant makes a final move to the depot node, with the arc distance being added to the tour length.

Results. We will form a single tour of the 24 nodes of the LA data, and adjust the vehicle capacity so that two unloading trips will be needed (a vehicle capacity of 40 tonnes against a total nodal demand of 104 tonnes), i.e. three sub-tours will be constructed.

The code was run with the initial pheromone levels set as $\tau_0 = 1/(200n)$ where 200 was chosen as a starting guess for the optimal tour length. Initially we ran the code performing 10,000 iterations, with the best two resulting tours for two separate runs displayed in Fig. 2.

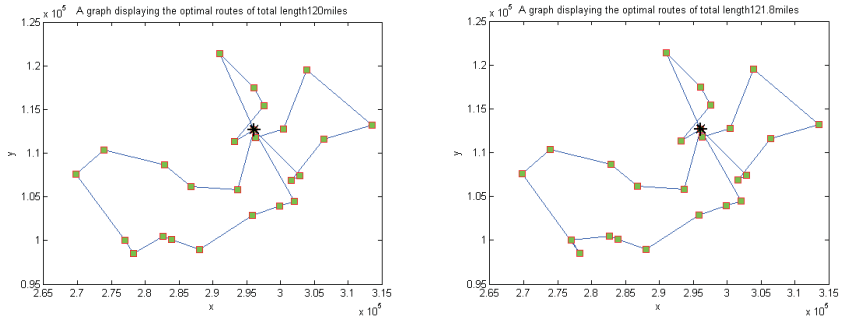
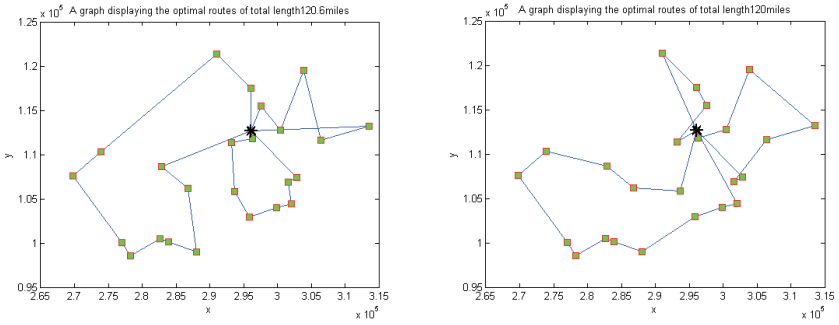


Figure 2. Graphs displaying (a) a tour of length 120 miles and (b) a tour of length 121.8 miles found in 10,000 iterations using the extended ACS.

The results, illustrated by the close similarity of Fig. 2.a and 2.b, suggest that the solution tours are converging to a local optimum, and the tour length is more than 10% shorter than that of the combined tours in Fig. 1.

We now update the initial pheromone level using these newly-found tour lengths so that $\tau_0 = 1/(120n)$, to determine the effect on the solu-

tion. Running the algorithm for 10,000 iterations produced the routes as displayed in Fig. 3. Overall, slightly shorter tours are found for the same number of iterations. The tour in Fig. 3.b matches the tours in Fig. 2, but that in Fig. 3.a is a new, equally good solution.



(a) (b)

Figure 3. The resulting graphs displaying (a) a tour of length 120.6 miles and (b) a tour of length 120 miles found in 10,000 iterations of the extended ACS with updated τ_0.

We did experiment with sending the ants to the unvisited node nearest to the depot, rather than to a random node, after a tip. Although the code ran faster, the sub-tours produced were not optimised and would need local improvement. The same occurred if ants were sent to the unvisited node furthest from the depot.

6. Conclusions

We have demonstrated on a small-scale problem a five-stage solution method for the recycling waste collection problem, and proposed an improved algorithm where the routing and scheduling stages are combined by the technique of monitoring the vehicle load within the ACO algorithm, and forcing an unloading trip when necessary. This technique could be further extended to model multi-compartment vehicles, which use kerbside sorting of recycling into paper, glass, plastics etc, with the load of each compartment being monitored independently in the same way as in Sec. 5. The k-means clustering could also be improved by using driving distances insted of Euclidean distances between nodes, with the centroid of each cluster being replaced by a central node (the node with minimum average distance to the other nodes in the cluster).

These improvements will be tackled in a follow-up project, using more finely-grained data for a proper experimental evaluation.

Acknowledgements

We are grateful to Maxine von Eye and her colleagues at Eunomia Research and Consulting for their involvement in the project, including provision of data and technical guidance on aspects of recycling collection.

References

[1] J. Bautista and J. Pereira. Ant algorithms for urban waste collection routing. *Lect. Notes Comput. Sc.*, 3172:386–403, 2004.

[2] J. E. Bell and P. R. McMullen. Ant colony optimization techniques for the vehicle routing problem. *Adv. Eng. Inform.*, 18(1):41–48, 2004.

[3] E. Bonabeau, M. Dorigo, and G. Theraulaz. *Swarm intelligence: from natural to artificial systems.* Oxford University Press, 1999.

[4] M. Dorigo and L. M. Gambardella. Ant colony system: A cooperative learning approach to the traveling salesman problem. *IEEE T. Evolut. Comput.*, 1(1):53–66, 1997.

[5] M. Dorigo and L. .M. Gambardella. Ant Colonies for the travelling salesman problem. *Biosystems*, 43(2):73–82, 1997.

[6] M. Dorigo and T. Stützle. *Ant colony optimization.* The MIT Press, 2004.

[7] H. A. Eiselt, M. Gendreau, and G. Laporte. Arc routing problems, part II: The rural postman problem. *Oper. Res.*, 43(3):399–414, 1995.

[8] R. T. Evering. *Optimising a Waste Collection Service.* MSc Project Report, Dept. of Mathematical Sciences, University of Bath, UK, 2011.

[9] P. Ji, H. Wu, and Y. Wu. Quadratic programming for the vehicle routing problem. In *Proc. 7th International Symposium on Operations Research and its Applications*, 2008.

[10] V. Karadimas, G. Kouzas, I. Anagnostopoulos, and V. Loumas. Urban solid waste collection and routing: The ant colony strategic approach. *International Journal of Simulation: Systems, Science & Technology*, 6(12-13):45–53, 2007.

[11] N. V. Karadimas, K. Papatzelou, and V. G. Loumos. Optimal solid waste collection routes identified by the ant colony system algorithm. *Waste Manage. Res.*, 25(2):139–147, 2007.

[12] A. E. Rizzoli, R. Montemanni, E. Lucibello, and L. M. Gambardella. Ant colony optimization for real-world vehicle routing problems. *Swarm Intelligence*, 1(2):135–151, 2007.

[13] S. Sahoo, S. Kim, B. I. Kim, B. Kraas, and A. Popov Jr. Routing optimization for waste management. *Interfaces*, 35(1):24–36, 2005.

GPU BASED PARALLEL GENETIC ALGORITHM LIBRARY

Péter Cserti, Szabolcs Szondi, Balázs Gaál, György Kozmann, István Vassányi

Department of Electrical Engineering and Information Systems, University of Pannonia, Veszprem, Hungary

cserti.peter@virt.uni-pannon.hu; szabolcs.szondi.hun@gmail.com; gaal.balazs@virt.uni-pannon.hu; {kozmann; vassanyi}@almos.vein.hu

Abstract The use of Genetic Algorithms (GAs) has grown to widespread acceptance by providing an efficient way to solve complex problems lacking deterministic solution method. GAs employ a special stochastic search method based on evolutionary theory, which gives them the ability to outperform most traditional search methods. Also their use of independent individuals makes them an ideal candidate for parallelization enhancing their inherently good performance even further. Their parallelizability on Graphical Processing Units (GPU) had been shown multiple times, but the implementations were either single objective GAs or just partially accelerated by GPUs, also every time they were experimental designs. The genetic algorithm library discussed in this article is the first that contains fully parallelized GPU implementations of multi-objective genetic algorithms besides the single objective ones. Furthermore, it is organized into a ready to use framework, which provides flexible and efficient GPU accelerated GAs. Thus enabling the user to solve complex problems faster than standard CPU based implementations would allow and with lower overall energy cost.

Keywords: C/C++, Genetic Algorithm, GPU, Many-threaded differential evolution, Multi-objective, Nvidia CUDA, Parallelization.

1. Introduction

Recently there has been a great change in the world of parallel computation with the appearance of the General Purpose Graphical Processing Units (GPGPU) which provide unprecedented computational power for desktop computers to solve general, not graphics related, problems by the use of massive parallelism. The ability to run computations parallel on hundreds of cores is no longer tied to expensive mainframes. A

high-end GPU provides computational power that was only reachable by supercomputers in a few years back. So by utilizing this power, parallelized problems can be solved faster and more economically on a simple desktop PC than ever before. However, parallelization if done correctly will not change the relative efficiency of two algorithms compared to each other, so even if parallelizing a simple but slow algorithm can yield better performance than any of its sequential competitors, it will never be able to compete with the parallelized version of an inherently quicker algorithm. As it has been proven numerous times, Genetic Algorithms (GAs) provide an efficient and fast alternative for solving complex problems where there is no deterministic algorithm to do it. Evolutionary search methods are able to outperform exhaustive- and random search algorithms, because they do not have to traverse the whole search space and their steps are not merely random either. Still their execution time can rise quite high as the complexity of the computation is rising, especially in multi-objective cases. This justifies the efforts to accelerate the computation by using GPUs. Fortunately GAs perform most of their computational tasks independently for each individual which enables them to be parallelized efficiently. Parallel execution on graphic cards can speed up the computation while retaining the flexibility and robustness of GAs. There are implementations of GPU based GAs from which the most noteworthy are ParadisEO-GPU and the work of Pospichal and Jaros [10] but these are single objective realizations. Also there are efforts to parallelize the computation of Pareto frontiers [16] on GPU but not the whole multi-objective algorithm. So the main novelty of the library described here is a fully parallelized multi-objective genetic algorithm running on GPU which to the best of the authors knowledge have never been published before. Also it is not an experimental design, it was built as a modular framework which hides the difficulties inherent with parallelization as well as the interactions between the graphic card and the host computer thereby, allowing the user to easily harness the computational power of the GPU. The GAME library (GPU Accelerated Multi-objective Evolutionary algorithm library) was designed according to the structure of GAlib [15], and like its ancestor, it contains base classes for basic GA archetypes as well as a set of derived classes, which are able to solve problems handled by traditional single- and multi-objective GAs. Furthermore the library readily contains multiple predefined fitness functions and both single- and multi-criterion knapsack problem solvers. Also the modular design of the framework allows for easy expandability to handle special situations. The usage of the predefined algorithms within the framework does not require GPU programming, or any kind of parallel programming experience, the fit-

ness functions can be written in C/C++ language in standard sequential manner, and are managed through a so called Host Genome. It is a standard C++ class, which is responsible for providing an interface for the user as well as for managing the parameters of the parallel genetic algorithm. The computations on the GPU side are handled by Device Genomes which are structured like GAlib's *GAGenome*. They can have integer, single/double precision real or binary data representation, but are not limited to the use of single objective computation. Furthermore every device genome can redefine its own genetic operators by this allowing for even more flexibility when creating new algorithms. This way the framework provides an efficient and easy way to reduce computational time as well as energy cost by the use of parallel computation capabilities of GPUs.

2. GPU Accelerated Multi-Objective Evolutionary Algorithm (GAME)

2.1 Background

The GAME library was written in C++ and the parallel computation on the GPU is handled by Nvidia CUDA [12], for this reason it requires a device with CUDA compute capability 2.0 or greater to function. To understand the design decisions and the working mechanisms of the library some introduction to the concept of GPU based parallelization is required.

The basic principles of parallelization is common for every GPGPU and programming language, a number of threads are organized into a group or block, then these blocks are organized into a grid. A device function (kernel) is executed on a grid, every block of the given grid is then distributed between a number of multiprocessors on the GPU, if there is not enough processing element available the remaining blocks are scheduled for execution. Threads within the same block are executed on the same multiprocessor, so they are guaranteed to be run in parallel and can be synchronized without considerable impact on performance, but there is no efficient way to synchronize between different blocks.

The memory on graphics cards is divided into multiple levels, the main two are global memory and on-chip memory. Global memory is the graphics card's built in DRAM memory which is usually very large in capacity, and also serves as an input/output buffer between the device and the host machine. Unfortunately it has relatively high latency which makes it less desirable for frequent use. On the other hand, the on-chip memory types, as their name suggests, are built into the multiprocessors thus having low latency (an order of magnitude better than

global memory), but their capacity is very limited. There are two types of them, register memory and shared memory. The former is usually limited to 8 KB, from which every thread gets dedicated share the exact amount based on the numerosity of the threads. The size of the latter varies from 16 to 48 KB and can be accessed by all threads within the same block by this enabling inter-thread communication. Because inter-block communication is expensive and efficient synchronization is only available within the boundaries of a block, the GPU architecture [12] is most suited for the implementation of island model genetic algorithms particularly for those with small populations. Each of these small populations can run on a single block utilizing regular migrations between them to keep overall diversity. It has also been shown that GAs using island model with systematic migrations can better exploit the search space while converging to the global optima, than the ones using a large singular population [1]. This way every block manages its own separate population, where every individual is controlled by a single thread, operations on blocks are separated by CUDA block barriers with zero overhead [12]. The populations are stored in the corresponding multiprocessors shared memory which enables the different threads to work with any data belonging to their block, without hindering the algorithms performance with high memory latency.

2.2 Structure

The library has an object oriented class design consisting of three main layers: Host genome, Game and the Device genomes. The first layer acts as an interface between the library and the user, where the user can be an actual person or another process. It is a standard C++ class running exclusively on the CPU, its task is to check, order and store the input data then present it for the second layer. Through it, every parameter related to the computation can be controlled. It also contains a special array called UserData which if set, will be moved to the GPU unmodified to allow for greater flexibility. The definition of fitness functions, while wrapped into a special kernel invocation, are and can be designed in a sequential manner utilizing the built in genotypes. Host genome also stores the results of the computation and presents them for the user in an organized manner.

The second layer is the main class of GAME bearing the same name, it is a transitional C++ class, running on the CPU and handling the GPU, it is responsible for every interaction happening between the host computer and the graphics card. It moves data between the two, calculates the amount of GPU resources to be allocated and translates every control

and computational parameter stored in the Host Genome for the GPU. Each instance of class GAME can be configured separately to set, the problem to be solved (by supplying a Host Genome) the GPU to be used and the scheduling of the computation can be configured independently. Thus allowing for the concurrent use of multiple CUDA 2.0 capable devices. It is the second layer's responsibility to supply launch parameters and start the proper kernel corresponding to the given archetype of GA (Single- or Multi-objective) as well as to define and initialize the random number generator(RNG) and the device genome corresponding to the genotype. As both needs to be instantiated on the GPU and the RNG require seed from the CPU while the genome's starting population can either be predefined using the Host Genome or will be random generated on the GPU. Also it is the second layer's task to collect statistical data from each computation and present it for the user.

The third and most important layer is on the device level, here multiple kernels and sets of CUDA classes work in conjunction to handle the computational tasks efficiently while allowing for flexibility within the algorithm. The first part is the kernel definition itself, it is responsible for controlling the sequence in which the genetic operators are invoked, by this ultimately differentiating between the single- and the multi-objective GAs. The flowchart on Fig. 1 shows the steps taken by the multi-objective genetic algorithm implemented in GAME library, each operator is parallelized and working on a whole population. Dashed lines represent points of local synchronization. As most of the operators are virtually identical between single- and multi objective GAs except, that MOGAs employ multiple fitness functions which in turn have to be summarized, in this case by the *Pareto dominance check* and *Crowding distance assignment* operators, so the evolutionary process could be guided unequivocally. The post processing step of *Pareto filtering* is required because of the use of multiple, separated populations where the global dominance of a locally Pareto dominant individual cannot be guaranteed when the results from different populations are merged. Consequently the flowchart of GAME library's single-objective GA implementation would look like Fig. 1, but without the three operators mentioned above.

The behavior of the genetic operators are controlled by the device genomes, these are sets of CUDA classes which exist only on the device, and control every aspect of the parallel computation, from efficiently distributing and moving data between memory levels of the GPU to the exact computational steps and data representation. Which genome to use during the computation is controlled by the problem definition in the fitness function so new genomes are easily integrated. Before the

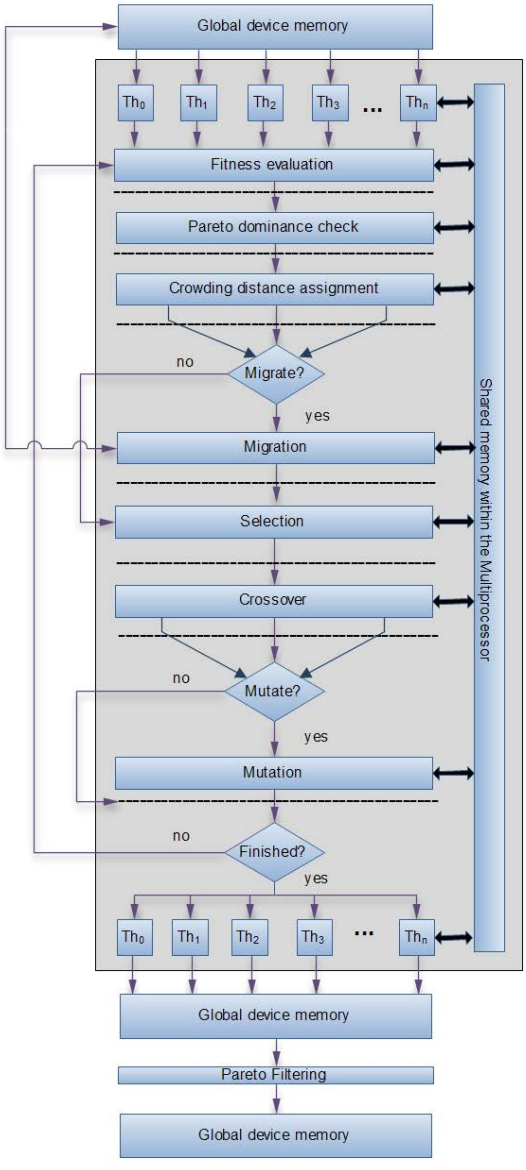


Figure 1. The flowchart of GAME's multi-objective genetic algorithm.

main kernel launch the chosen genome is instantiated in the device's global memory by the second layer. Upon the launch of the main kernel the genome will deploy itself to the shared memory space of every

block, along with the individuals and other parameters belonging to that population. When designing the genomes the goal was to maximize performance while keeping flexibility and the memory requirements as low as possible. Each method and operator are designed so they can be accessed by any number of threads concurrently without conflicts, even though there is only a single instance of the genome is present within every block. Optimization methods are incorporated into the genomes like addressing the individuals through redirection which eliminates the need for actually moving them within shared memory when sorting is needed. Furthermore genomes are structured in such way that they will coalesce global memory access patterns and eliminate shared memory bank conflicts. This is very important because bank conflicts and uncoalesced memory accesses are the two main performance reducing factors in GPU programming as the former will lead to the serialization of the computation and the latter will burden the GPU with excessive and unnecessary global memory loads which is by far the slowest operation on the graphics card. The genome's deploy function is also a quite unique design, allowing the class to copy itself to a specific shared memory address from its original place in global memory. Also to incorporate the use of UserData array a special memcopy operation had to be created that can move data between global and shared memory in a type independent manner, because CUDA does not support this kind of operation on the device by default. The GPU works as a completely closed, independent device, all computations are handled by the operators defined within the genomes. When the processing is finished the results are moved back to global memory from where the second layer will copy them to the Host Genome before freeing up resources and shutting down the kernel.

In summary the first layer acts as an interface while also stores data related to the problem. The second layer task to interpret the data collected by the first layer and set up the GPU for the computation. While the third layer is responsible for controlling every calculation and data movement happening on the GPU.

2.3 Operators

Independently from GA archetype every genome uses parallel tournament selection and 1 bit random mutation by default while the type of the crossover operator is dependent only upon the genotype. As the main novelty of this article is the parallel, GPU based implementation of a MOGA its main operators will be discussed in more detail. The

MOGA implementation featured in this library is modeled upon the widely accepted, sequential NSGA2 [2].

Parallel Pareto dominance checking. This operator employs dedicated threads to every pair of individual that need to be checked and a matrix sized according to the number of threads for storing information about domination. It is able to find all Pareto frontiers from 10 up to 50 times faster than the sequential approach. To achieve optimal performance, the dominance checking also have to work within the shared memory, but there are limitations for how much shared memory and how many threads can be used in a block. To reach the best possible efficiency on the GPU a specialized Pareto frontier computation method was created which uses $\frac{N^2-N}{2}$ threads to compute the Pareto frontiers of N individuals in parallel. First a domination matrix is created, by calculating dominance relation of every pair of individuals according to all the problem dimensions. Then this matrix is processed by checking it's rows in parallel, if no domination is indicated than the individual represented by that row is dominant and part of the first Pareto front. Then the rows and columns belonging to the individuals of the first front are omitted from the matrix. Thus allowing the individuals of the second front to appear as dominant. This process repeats until all individuals are organized into fronts. The dominance checking also happens independently for each block, so it can run in parallel for every population.

Pareto filtering. After the algorithm finished evolving, there is also a global Pareto dominance checking, because the fact that an individual is dominant in its own population, doesn't necessarily means that it is dominant in the every population evolving concurrently. So this last global dominance checking actually works like a filter, dismissing dominated individuals from the first front. This algorithm works in the global memory, and from a different kernel launch, so it isn't restricted by the block limitations, and it only have to find the first Pareto front, which means there is no need for a matrix to store domination information. It finds the first frontier in one step, using the fitness data left in the devices global memory by the main kernel. The separate kernel launch is needed as it is the only way to synchronize between blocks to actually stop the main kernel. Leaving the data in the global memory means, that there is no unnecessary data movement between device and host, and the GPU is already initialized, so there is practically no overhead from the second kernel launch.

3. Results

Multi-objective GA. GAME was compared to three state of the art metaheuristic algorithms: PAES [8], NSGAII [2] and SPEA2 [17] which are all using pareto terminology to solve multi-objective problems and were uniformly running 250 generations with the population size being 100. All three algorithms are implemented in Java and part of jMetal [4] metaheuristic algorithm library and testing framework. GAME was compared to them by goodness of solution and execution time on six different multi-objective test problems: Fonseca [6], Kursawe [11], Viennet2&3 [14] and DTLZ1&2 [3] which are chosen to represent a variety of problem types. The problems were two dimensional with the exception of the Viennet problems. To get a comprehensive picture of the performance of GAME's MOGA implementation multiple performance metrics were used: Hypervolume [5], Inverted Generation Distance [13], Spread [2], and Additive Epsilon [9]. The following tables are averaged from 25 trials, standard deviation values are supplied in subscript. The darkest shading represents the best results.

Table 1. Hypervolume: Mean and standard deviation.

	GAME	NSGAII	SPEA2	PAES
Fonseca	$3.13e-01_{2.6e-04}$	$3.09e-01_{3.3e-04}$	$3.11e-01_{1.5e-04}$	$3.07e-01_{6.7e-04}$
Kursawe	$3.59e-01_{9.0e-03}$	$3.99e-01_{2.4e-04}$	$4.01e-01_{1.8e-04}$	$3.94e-01_{1.7e-03}$
Viennet2	$9.28e-01_{1.3e-03}$	$9.20e-01_{1.4e-03}$	$9.25e-01_{2.9e-04}$	$9.12e-01_{3.2e-03}$
Viennet3	$8.37e-01_{3.7e-04}$	$8.32e-01_{6.0e-04}$	$8.27e-01_{1.3e-03}$	$8.30e-01_{3.0e-03}$
DTLZ1	$4.27e-01_{1.2e-02}$	$4.93e-01_{3.4e-04}$	$4.95e-01_{7.0e-05}$	$4.90e-01_{1.1e-03}$
DTLZ2	$2.13e-01_{4.7e-04}$	$2.11e-01_{3.5e-04}$	$2.12e-01_{1.0e-04}$	$2.10e-01_{3.8e-04}$

Table 2. Inverted Generation Distance: Mean and standard deviation.

	GAME	NSGAII	SPEA2	PAES
Fonseca	$2.14e-04_{4.2e-05}$	$3.11e-04_{1.7e-05}$	$2.27e-04_{4.5e-06}$	$4.50e-04_{7.4e-05}$
Kursawe	$1.57e-03_{6.1e-04}$	$1.87e-04_{1.0e-05}$	$1.36e-04_{3.6e-06}$	$3.37e-04_{4.6e-05}$
Viennet2	$1.59e-04_{2.6e-05}$	$3.25e-04_{3.5e-05}$	$1.72e-04_{4.5e-06}$	$4.07e-04_{5.9e-05}$
Viennet3	$8.78e-04_{4.1e-04}$	$1.72e-04_{1.3e-05}$	$1.46e-04_{2.1e-05}$	$5.20e-04_{9.2e-04}$
DTLZ1	$5.20e-03_{1.7e-03}$	$4.60e-04_{2.8e-05}$	$3.09e-04_{3.7e-06}$	$6.68e-04_{1.1e-04}$
DTLZ2	$1.15e-03_{4.2e-04}$	$4.77e-04_{3.8e-05}$	$3.50e-04_{5.3e-06}$	$6.19e-04_{6.2e-05}$

Table 3. Spread: Mean and standard deviation.

	GAME	NSGAII	SPEA2	PAES
Fonseca	$6.22e-01_{2.3e-02}$	$4.39e-01_{4.3e-02}$	$1.43e-01_{1.1e-02}$	$6.34e-01_{5.3e-02}$
Kursawe	$6.81e-01_{5.8e-02}$	$5.99e-01_{2.5e-02}$	$4.41e-01_{8.7e-03}$	$9.07e-01_{4.3e-02}$
Viennet2	$9.58e-01_{3.6e-02}$	$8.13e-01_{6.5e-02}$	$7.08e-01_{3.2e-02}$	$1.04e+00_{6.7e-02}$
Viennet3	$1.37e+00_{4.4e-02}$	$7.23e-01_{5.0e-02}$	$7.78e-01_{5.5e-02}$	$7.37e-01_{8.5e-02}$
DTLZ1	$1.23e+00_{1.5e-01}$	$5.16e-01_{4.3e-02}$	$3.13e-01_{3.5e-01}$	$9.74e-01_{2.2e-01}$
DTLZ2	$8.55e-01_{3.6e-02}$	$4.23e-01_{3.1e-02}$	$1.69e-01_{9.1e-03}$	$6.40e-01_{3.4e-02}$

Table 4. Additive Epsilon: Mean and standard deviation.

	GAME	NSGAII	SPEA2	PAES
Fonseca	$1.09e-02_{3.5e-03}$	$1.33e-02_{2.3e-03}$	$8.01e-03_{8.3e-04}$	$2.16e-02_{7.5e-03}$
Kursawe	$1.11e+00_{4.8e-01}$	$8.19e-02_{9.9e-03}$	$7.01e-02_{8.1e-03}$	$2.43e-01_{7.8e-02}$
Viennet2	$9.47e-03_{1.6e-03}$	$3.49e-02_{1.0e-02}$	$1.97e-02_{1.9e-03}$	$3.63e-02_{7.3e-03}$
Viennet3	$2.36e-02_{3.0e-03}$	$5.03e-02_{1.0e-02}$	$5.09e-02_{6.6e-03}$	$4.69e-02_{1.1e-02}$
DTLZ1	$7.95e-02_{2.4e-02}$	$7.36e-03_{1.4e-03}$	$3.52e-03_{2.8e-04}$	$1.18e-02_{4.6e-03}$
DTLZ2	$7.15e-02_{3.4e-02}$	$1.33e-02_{3.0e-03}$	$7.51e-03_{7.5e-04}$	$2.44e-02_{1.4e-02}$

Table 5. Execution time (in milliseconds). Mean and standard deviation.

	GAME	NSGAII	SPEA2	PAES
Fonseca	$87.64_{22.53}$	$511.72_{186.99}$	$4531.96_{453.22}$	$581.04_{89.84}$
Kursawe	$101.48_{44.84}$	$616.56_{188.93}$	$4003.32_{362.10}$	$426.60_{80.37}$
Viennet2	$98.04_{7.83}$	$897.60_{184.71}$	$11150.04_{2301.31}$	$2514.84_{309.30}$
Viennet3	$105.32_{4.72}$	$1020.68_{96.36}$	$10546.12_{1581.52}$	$3232.60_{1022.46}$
DTLZ1	$107.96_{5.71}$	$918.52_{115.73}$	$9752.36_{1520.88}$	$882.40_{127.36}$
DTLZ2	$107.68_{3.12}$	$844.20_{9.62}$	$14298.68_{2167.74}$	$1572.04_{23.71}$

Figure 2 is a graphical representation of Table 5 for easier comparison. Figure 3.b shows how GAME scales up when the problem dimension and the number of individuals is increased.The vertical scale shows execution time in milliseconds on both graphs. While Fig. 3.a shows GAME's speed advantage compared to NSGAII with the same parameters, solving

the DTLZ2 problem. Please note that the vertical scale on this graph actually shows how many times GAME is quicker than NSGAII.

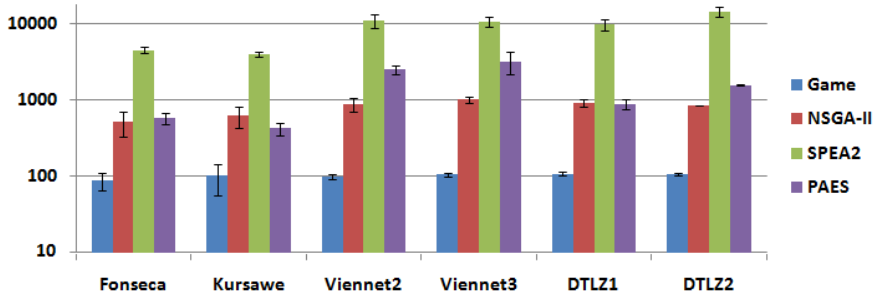
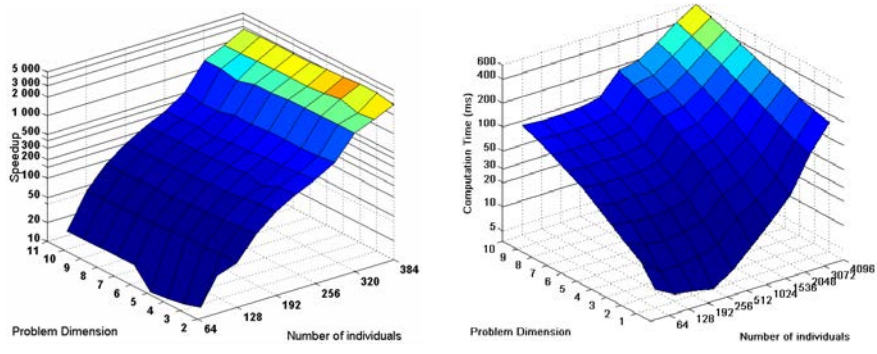


Figure 2. Execution times graph (in milliseconds).



(a) Speedup compared to NSGAII (b) GAME's scale-up

Figure 3. Other properties of GAME.

Single-objective GA. Table 6 assesses the performance of the single-objective GA implemented in GAME compared to GAlib [15], using numerical problems. Table 7 compares GAME's single-objective knapsack solver's performance against GPUGA [7], which is a parallel GA especially designed to solve knapsack problems using GPUs. Both algorithms were run for 300 generations, please note that for the sake of comparability Table 7's Time columns only show the running time of the GPU kernels. The results shown in the tables are averaged from 25 trials.

Table 6. Single-objective GA results, lower is better.

	GAME-SO		GAlib	
Problems	Best value	Time [ms]	Best value	Time [ms]
Ackley's Path	$6.00e-06_{9.00e-06}$	$19.548_{0.182}$	$1.04e-02_{9.01e-03}$	$1104.5_{95.295}$
Griewangk's	$8.00e-06_{2.50e-05}$	$20.478_{0.217}$	$2.09e-01_{1.90e-01}$	$1761.3_{126.86}$
Hyper-ellipsoid	$0.00e+00_{0.00e+00}$	$12.138_{0.221}$	$8.00e-06_{9.00e-06}$	$934.4_{44.421}$
Rastrigin's	$0.00e+00_{1.00e-06}$	$15.485_{0.302}$	$5.56e-03_{6.21e-03}$	$1160.6_{126.42}$
Rosenbrock	$0.00e+00_{0.00e+00}$	$10.961_{0.247}$	$1.31e-03_{2.08e-03}$	$1096.7_{70.794}$

Table 7. Knapsack benchmark, higher best value is better.

	GAME-Knapsack		GPUGA	
Problem size	Best value	Time [ms]	Best value	Time [ms]
Number of items: 10	$798.0_{0.00}$	$1.26_{0.044}$	$788.8_{20.57}$	$23.4_{0.548}$
Number of items: 20	$905.8_{13.27}$	$9.40_{0.141}$	$913.8_{20.81}$	$31.8_{1.789}$
Number of items: 30	$1679.6_{15.79}$	$18.58_{0.130}$	$1638.8_{26.56}$	$30.8_{3.701}$
Number of items: 40	$4100.6_{76.61}$	$22.0_{0.00}$	$4058.2_{17.33}$	$118.8_{1.79}$

4. Conclusions

The main focus of this article was the Multi-objective GA implementation for GPUs as it is to the best knowledge of the authors the first of its kind. Being a new design on a relatively new platform it suffers from some limitations. Mainly from the maximal size of a population is being limited to 32, which makes it hard to find well spread out solutions as it is shown by the *SPREAD* metric. On the other hand GAME mostly manages to keep up in the quality of solutions while being 8–140 times faster than its state of the art competitors. Also this speed advantage will grow as the problems get larger, because thanks to its multiple small populations GAME scales up quite well. Also a good portion of the running time is taken up by the GPU initialization, thus remaining constant. While GAME's single-objective GA is not the first of its kind, and received less attention in the course of this paper, its result clearly show that it can outperform even its GPU accelerated competitors.

Acknowledgements

The work presented was partly funded by the National Innovation Office, Hungary (project No TÁMOP-4.2.2/B-10/1-2010-0025) and Inomed Inc.

References

[1] E. Cantu-Paz. Designing Efficient Master-Slave Parallel Genetic Algorithms. In *Proc. 3rd Annual Conference on Genetic Programming*, 1998.

[2] K. Deb, A. Pratap, S. Agrawal, and T. Meyarivan. A fast elitist multi-objective genetic algorithm: NSGA-II. *IEEE T. Evolut. Comput.*, 6(2):182–197,2000.

[3] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler. Scalable test problems for evolutionary multiobjective optimization. In *Evolutionary Multiobjective Optimization. Theoretical Advances and Applications*, pages 105-145. Springer, 2005.

[4] J. J. Durillo and A. J. Nebro. jMetal: A Java framework for multi-objective optimization. *Adv. Eng. Softw.*, 42:760–771, 2011.

[5] M. Emmerich, N. Beume, and B. Naujoks. An emo algorithm using the hypervolume measure as selection criterion. *Lect. Notes Comput. Sc.*, 3410:62–75, 2005.

[6] C. M. Fonseca and P. J. Flemming. Multiobjective optimization and multiple constraint handling with evolutionary algorithms - part ii: Application example. *IEEE T. Syst. Man Cy. A*, 28(1):38–47, 1998.

[7] J. Jaros and P. Pospichal. A fair compariosn of modern CPUs and GPUs running the genetic algorithm under the Knapsack benchmark. *Lect. Notes Comput. Sc.*, 7248:426–435, 2012.

[8] J. Knowles and D. Corne. The Pareto archived evolution strategy: a new baseline algorithm for Pareto multiobjective optimisation. In *Proc. Congress on Evolutionary Computation*, pages 98–105, 1999.

[9] J. Knowles, L. Thiele, and E. Zitzler. A Tutorial on the Performance Assessment of Stochastic Multiobjective Optimizers. Technical Report 214, Computer Engineering and Networks Laboratory, ETH Zurich, 2006.

[10] P. Kromer, J. Platos,V. Snasel, and A. Abraham. A comparison of many-threaded differential evolution and genetic algorithms on CUDA. In *Proc. 3rd World Congress on Nature and Biologically Inspired Computing*, pages 509–670, 2011.

[11] F. Kursawe. A variant of evolution strategies for vector optimization. *Lect. Notes Comput. Sc.*, 496:193–197, 1991.

[12] NVIDIA: Compute Unified Device Architecture programming Guide. 2011.

[13] D. A. Van Veldhuizen and G. B. Lamont. Multiobjective Evolutionary Algorithm Research: A History and Analysis. Technical Report TR-98-03, Department of Electrical and Computer Engineering, Air Force Institute of Technology, Wright-Patterson AFB, OH, 1998.

[14] R. Viennet. Multicriteria optimization using a genetic algorithm for determining the Pareto set. *Int. J. Syst. Sci.*, 27(2):255–260. 1996.

[15] M. Wall. GAlib: A C++ Library of Genetic Algorithm Components. Mechanical Engineering Department, Massachusetts Institute of Technology, 1996.

[16] M. L. Wong and T. T. Wong. Implementation of Parallel Genetic Algorithms on Graphics Processing Units. In *Intelligent and Evolutionary Systems*, volume 187, pages 197–216. Springer-Verlang, 2009.

[17] E. Zitzler, M. Laumanns and L. Thiele. SPEA2: Improving the Strength Pareto Evolutionary Algorithm. *Evolutionary Methods for Design, Optimization and Control with Applications to Industrial Problems*, pages 95-100, 2001.

BI-OBJECTIVE RESOURCE ALLOCATION IN SPATIALLY DISTRIBUTED COMMUNICATION NETWORKS

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

Risto Vesanen, Erkki Laitinen
Department of Mathematical Sciences, University of Oulu, Finland
rvesane@gmail.com; erkki.laitinen@oulu.fi

Abstract We deal with a bi-objective problem of optimal resource allocation in communication networks which consists of maximizing the total network utility, i.e., the fee paid by the consumers, and minimizing the costs of implementing these resources. We compare two techniques for solving this problem, a traditional approach of first transforming the problem into a single-objective form and then solving it using a suitable method, and a population based Pareto optimization approach utilized by an evolutionary algorithm which provides a more informative result in the form of a set of tradeoff solutions.

Keywords: Differential evolution, Evolutionary multiobjective optimization, Resource allocation, Spatially distributed network, Subgradient method.

1. Introduction

Allocation of limited resources among competing entities according to predefined criteria is an optimization problem met in practice in numerous forms. In the quickly developing area of wireless communications, it finds its application in emerging technologies, such as mobile ad-hoc networks and sensor networks, which enhance the availability and quality of information and support a wide range of services. Due to the presence of a great number of nodes whose locations may not be fixed and due to strict limitations on communication resources, such as the transmission power and/or the bandwidth, effective resource allocation

is needed for these networks to provide acceptable levels of services (for more information see, for example, [3, 18]).

Resource allocation in wireless networks has been addressed using various techniques, including scalar and vector optimization [9, 18], equilibrium and game theory [1, 22]. Moreover, dealing with resources in complex networks usually reveals it is necessary to utilize a proper decomposition or clustering approach, which can be based on zonal, time, frequency and other attributes of the network nodes (see, for example, [2, 16]).

In this paper, we consider a bi-objective optimization problem, which consists of maximizing the total network utility, i.e., the fee paid by the consumers, and minimizing the total network implementation costs by means of the optimal resource allocation. To solve this problem, we apply two different decision making procedures, a preference-based one that exploits available decision-making information to transform the problem into a single-objective form and then solves it, and the ideal one that utilizes the multi-objective optimization approach to find multiple tradeoff solutions that are then available for further analysis and final decision-making.

First, we follow the method from [13], which applies the dual decomposition approach for solution finding. Using this approach, we obtain a set of single objective lower-level optimization problems and a bi-objective upper-level problem. The latter is transformed into a scalar form with the difference between the total utility and the total costs as the optimization objective. Under certain assumptions for the involved utility and cost functions, the problem can then be solved using the subgradient optimization methods. The decomposition approach and a suitable subgradient algorithm were presented in [13]. We use the results of preference-based method to verify the solutions of an alternative approach.

Second, we exercise an alternative approach which is to consider the task in its original form and search for Pareto optimal solutions that represent trade-offs between the conflicting objectives. This provides a network manager with a better insight into the problem objective space and makes it possible to choose the most suitable trade-off solution according to additional preferences. Moreover, supported by a population-based search algorithm, the approach is capable of finding an approximation for the Pareto optimal set in a single run. Specifically, we use the evolutionary algorithm called DEMO (Differential Evolution for Multiobjective Optimization) [15, 20].

The text is further organized as follows. We present the problem and provide the optimization formulations, outline the two applied optimiza-

tion methodologies, and report on the experimental setup, numerical experiments and results.

2. Optimal Network Resource Allocation

2.1 Problem Statement

Let us consider a network with nodes which is divided into n zones (see Fig. 1). The problem of the network manager is to allocate at most C units of network resources, such as bandwidth, to the network zones so that the cost of providing the resources $a_k, (k = 1, ..., n)$, to the zones is minimized, and the total network utility is maximized.

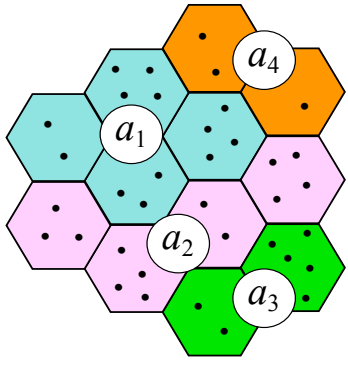


Figure 1. Illustration of the zonal structure of the network.

In this scenario, each network zone k involves $|I_k|$ nodes/consumers (I_k denotes the index set of nodes in zone k), with known individual utility functions $U_j(x_j)$, depending on the resource x_j allocated to the node $j, (j \in I_k)$. For each zone k we denote by $f_k(a_k)$ the optimal value of the following (lower-level) optimization problem:

$$\text{maximize} \quad \sum_{j \in I_k} U_j(x_j), \tag{1}$$

$$\text{subject to} \quad \sum_{j \in I_k} x_j \leq a_k, \tag{2}$$

$$\alpha_j \leq x_j \leq \beta_j, \ j \in I_k, \ \alpha_j \geq 0, \ \beta_j < a_k. \tag{3}$$

Hence, $f_k(a_k)$ determines the total utility of consumers of zone k if the resource value for this zone equals a_k. However, the implementation costs $h_k(a_k)$ for providing this quantity of the resource for zone k must be taken into account as well. As a result, the upper-level optimization problem consists of maximizing the total network utility and minimizing

the total network implementation costs by means of the optimal resource allocation:

$$\text{maximize} \quad \sum_{k=1}^{n} f_k(a_k), \tag{4}$$

$$\text{minimize} \quad \sum_{k=1}^{n} h_k(a_k), \tag{5}$$

$$\text{subject to} \quad \sum_{k=1}^{n} a_k \le C, \tag{6}$$

$$a_k \ge 0, \ k = 1, ..., n. \tag{7}$$

Criteria in (4) and (5) need not be homogeneous in general. The problem is both bi-objective and bi-level, and the presented problem statement allows to simultaneously take into account the income and the expenses from the two levels.

2.2 Optimization Formulations

The following approaches to formulation of the optimality concept for problem (4)–(7) can be applied.

- One can replace the bi-objective problem (4)–(7) by its scalarization via assigning weights, say, $\gamma_1 > 0$ and $\gamma_2 > 0$ to the criteria. This approach leads to the concave maximization problem:

$$\text{maximize} \quad \sum_{k=1}^{n} (\gamma_1 f_k(a_k) - \gamma_2 h_k(a_k)), \tag{8}$$

  subject to (6) and (7). Clearly, the determination of these weights may be difficult in the case the criteria in (4) and (5) are not homogeneous.

- Alternatively, one can consider the bi-objective problem (4)–(7) as the Pareto optimization problem:

$$\text{maximize}_{\succ} \quad \mathbf{F}(\mathbf{a}) = \sum_{k=1}^{n} \mathbf{F}^{(k)}(a_k), \tag{9}$$

  subject to (6) and (7). In this formulation, $\mathbf{a} = (a_1, ..., a_n)^T$, $\mathbf{F}^{(k)}(a_k) = (f_k(a_k), -h_k(a_k))^T$, $k = 1, \ldots, n$, and $\succ$ denotes the Pareto dominance relation

$$x \succ y \iff x_i \ge y_i \ \forall i \text{ and } x \ne y.$$

3. Problem Solving Methodology

3.1 Preference-Based Approach Using a Subgradient Method

Defining certain additional, natural properties of cost functions, the problem (1)–(3) has always a solution and it is easy to see that f_k is concave on $\mathbb{R}_+$, and then (4), (6), and (7) becomes a scalar concave maximization problem (see [13]). Also, if we suppose that each cost function h_k is convex, then (5), (6), and (7) becomes a scalar convex minimization problem.

The scalar optimization problems (4) and (5) are separable, however, function f_k can be non-differentiable and its values computed algorithmically without any explicit formula. In this case, we can apply suitable non-differentiable optimization methods for convex problems [12, 17] to find the solution of (8), (6), (7). From the family of subgradient techniques suitable for solving this problem we applied an iterative *relaxation subgradient method* for convex minimization [11, 12].

3.2 Evolutionary Multiobjective Optimization

A suitable algorithmic platform to support ideal multiobjective optimization [4, 10] is evolutionary computation [6]. Most of the initially proposed evolutionary algorithms for multiobjective optimization, including the well-known NSGA (Nondominated Sorting Genetic Algorithm) [4, 5], employ genetic algorithms to explore the variable space. However, as the differential evolution (DE) algorithm [14, 19] has proved to be a powerful numerical optimizer, several extensions of DE were later proposed for solving numerical multiobjective optimization problems.

In this work, we use a DE-based multiobjective optimization algorithm called DEMO [15, 20]. In DEMO, candidate solutions are represented as n-dimensional vectors. New solutions are constructed from the existing ones using vector addition and scalar multiplication.The candidate and its parent are compared using the Pareto dominance relation. If the candidate dominates the parent, it replaces the parent in the current population. If the parent dominates the candidate, the candidate is discarded. Otherwise, when the candidate and its parent are incomparable, the candidate is added to the population. After constructing candidates for each parent individual in the population, the population size possibly exceeds the predefined value. In this case, the population needs is truncated to the original size using one of the several implemented environmental selection mechanisms. The algorithm was, for example, applied in multiobjective optimization of the industrial steel

casting [8]. In addition to the serial version, a parallel implementation of the algorithm was developed and successfully applied to engineering multiobjective optimization problems with computationally demanding solution evaluation [7].

4. Experimental Evaluation

4.1 Test Problems

The abovementioned optimization methodology was numerically evaluated on three of the test problems (labeled 1a, 2a and 3a) from [21], where the scalar problem formulation (8) was considered. To make our initial exploration easier, the selected problems were special cases designed in a way that uniform allocation of the resource to the nodes within zones is optimal. Assuming this property, these problems need not be treated as bi-level, however, this fact does not change their bi-objective nature. The selected test problems share the following characteristics:

- number of zones $n = 5$,
- the same resource constraints for all nodes: $1 \leq x_j \leq 5$, $j \in I_k$, $k = 1, ..., n$,
- the same utility function for all nodes: $U_j(x_j) = -x_j^2 + 15x_j$, $j \in I_k$, $k = 1, ..., n$,
- linear resource implementation costs per zone: $h_k(a_k) = \omega_k a_k$, $k = 1, ..., n$.

Problem 1a is further characterized by:

- toatal amount of resource $C = 150$,
- number of nodes per zone $|I_k| = 5$, $k = 1, ..., n$,
- resource implementation costs coefficients: $\omega_k = 7$, $k = 1, ..., n$.

Problem 2a differs from problem 1a in that the costs of implementing the resources vary across the zones as indicated by the cost function coefficients: $\omega_1 = 3$, $\omega_2 = 7$, $\omega_3 = 9$, $\omega_4 = 11$, and $\omega_5 = 13$. Problem 3a differs from problem 1a in the amount of resource, $C = 200$, and varying number of nodes across the zones: $|I_1| = 2$, $|I_2| = 4$, $|I_3| = 6$, $|I_4| = 8$, and $|I_5| = 10$.

4.2 Experimental Setup

The aim of the experiments was to compare the single- and multi-objective optimization approaches on the resource allocation task. The single-objective optimization assuming the problem formulation (8) with $\gamma_1 = \gamma_2 = 1$ was performed and the results reported in [21]. Three optimization methods were tested there and in the comparison we will refer to the results of the relaxation subgradient method [11, 12] that on average produced the best results.

In this study we performed bi-objective optimization using the DEMO algorithm. To assess candidate resource allocations, an external evaluator of the optimization objectives was implemented that, given the network configuration and the amounts of resources allocated to the zones, calculated the total utility and the implementation costs. Note that the uniform distribution of resources within each zone was assumed.

The DEMO algorithm was integrated with the external evaluator and found well-performing under the following algorithm parameter settings: population size 20, number of evaluated candidate solutions 1600, scaling factor value 0.5, and crossover probability c = 0.5.

Tuned this way, the algorithm operated effectively and its results were repeatable, both in terms of the hypervolume indicator and the produced fronts of nondominated solutions. Fig. 2 shows the progress of hypervolume over five runs of the optimization algorithm, confirming negligible differences towards the end of the runs.

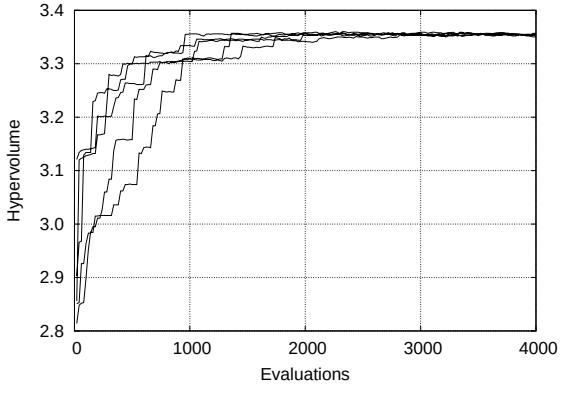


Figure 2. Hypervolume improvement over five runs of the optimization algorithm.

The integrated optimization environment was run on a Windows platform with a 2.8 GHz i7 CPU and 4 GB RAM.

4.3 Results

Figure 3 shows examples of nondominated fronts found for test problems in the Pareto optimization approach. Their cardinalities are equal to the population size of the optimization algorithm.

Results of the Pareto optimization approach are suitable for comparisons and analyses of tradeoffs between the criteria. Let us look, for example, at the results for problem 1a. Scalar optimization with the relaxation subgradient method allocates the resource to the zones as $a_k = 20$, $k = 1, ..., 5$, meaning that 100 out of available 150 units of the resource are utilized. In this case, the total utility of the network is 1100, and the costs of implementing the resources 700, yielding the difference (the actual optimization criterion considered by this method) of 400. The closest solution from the nondominated front produced by DEMO gives resource allocation $\mathbf{a} = (21.85, 21.01, 19.69, 15.37, 21.01)^T$ which amounts to 98.93 units. Here the total utility equals 1087.11, the implementation costs are 692.52, and the difference is 394.59. In fact, the two solutions are incomparable in the Pareto sense, since the former offers higher utility and the latter lower costs. Naturally, the difference is due to the fact that the two optimization methods pursue different goals and operate differently. Yet, a detailed inspection shows that the single solution from scalar optimization is neither dominated by nor it dominates any solution from the obtained front. This was confirmed in multiple runs on all three test problems.

However, the nondominated fronts provide additional information about similar (in terms of the difference between the utility and the costs) solutions and revel to what degree one can be traded for the other.

5. Conclusion

We presented an empirical study in optimal resource allocation in spatially distributed communication networks. In its general form, this is a bi-level problem and its upper part includes two objectives, maximizing the network utility and minimizing the implementation costs. We compared a single- and a multiobjective approach to the problem, solving three test instances. At the level of a single solution, the matching was satisfactory, however, the population based multiobjective optimization is clearly advantageous as it provides more informative results and imposes no limitations on the involved utility and cost functions. This work will continued by demonstrating these potentials in solving more complex general-type problem instances with particular interest in studying the interaction between the two levels during the optimization procedure.

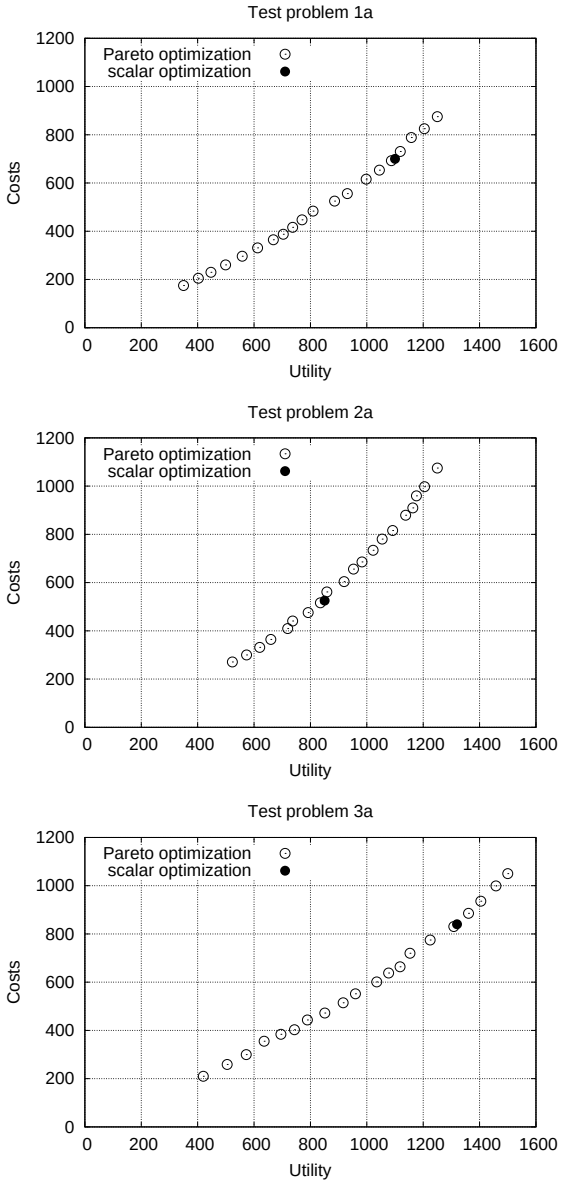


Figure 3. Fronts of nondominated solutions to the test problems found in the Pareto optimization approach, compared with the results of scalar optimization.

Acknowledgment

The work presented in this paper was supported by the Slovenian Research Agency under the Slovenian-Finnish project BI-FI/11-12-018

Constrained Multiobjective Optimization Based on Simulation Models and the Research Programme P2-0209 *Artificial Intelligence and Intelligent Systems*.

References

[1] E. Altman and L. Wynter. Equilibrium, games, and pricing in transportation and telecommunication networks. *Netw. Spat. Econ.*, 4(1):7–21, 2004.

[2] Y. P. Chen and A. L. Liestman. A zonal algorithm for clustering ad hoc networks. *Int. J. Found. Comput. Sci.*, 14(2):305–322, 2003.

[3] X. Cheng, X. Huang, and D.-Z. Du, editors. *Ad Hoc Wireless Networking*. Kluwer, Boston, 2004.

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

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

[6] Á. E. Eiben and J. E. Smith. *Introduction to Evolutionary Computing*. Springer, Berlin, 2003.

[7] B. Filipič and M. Depolli. Parallel evolutionary computation framework for single- and multiobjective optimization. In R. Trobec, M. Vajteršic, and P. Zinterhof (Eds.), *Parallel Computing – Numerics, Applications, and Trends*, pages 217–240, Springer, Dordrecht, 2009.

[8] B. Filipič, T. Tušar, and E. Laitinen. Preliminary numerical experiments in multiobjective optimization of a metallurgical production process. *Informatica*, 31(2):233–240, 2007.

[9] S. Jordan. Resource allocation in wireless networks. *J. High Speed Netw.*, 5(1):23–34, 1996.

[10] J. Knowles, D. Corne, and K. Deb, editors. *Multibjective Problem Solving from Nature – From Concepts to Applications*. Springer, Berlin, 2008.

[11] I. V. Konnov. A method of the conjugate subgradient type for minimization of functionals. *Issled. Prikl. Mat.*, 12:59–62, 1984 (in Russian). English translation in *J. Soviet Math.*, 45(2):1026–1029, 1989.

[12] I. V. Konnov. *Methods of Nondifferentiable Optimization*. Kazan University Press, Kazan, 1993 (in Russian).

[13] I. V. Konnov, O. Kashina, and E. Laitinen. Optimization problems for control of distributed resources. *Int. J. Model. Ident. Control*, 14(1/2):65–72, 2011.

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

[15] T. Robič and B. Filipič. DEMO: Differential evolution for multiobjective optimization. *Lect. Notes Comput. Sc.*, 3410:520–533, 2005.

[16] K. Rohloff, J. Ye, J. Loyall, and R. Schantz. A hierarchical control system for dynamic resource management. In *Proc. Work-In-Progress Session of the 12th Real-Time and Embedded Technology and Applications Symposium*, pages 37–40, San Jose, USA, 2006.

[17] N. Z. Shor. *Minimization Methods for Non-differentiable Functions*. Naukova Dumka, Kiev, 1979 (in Russian). English translation by Springer-Verlag, Berlin, 1985.

[18] S. Stańczak, M. Wiczanowski, and H. Boche. *Resource Allocation in Wireless Networks: Theory and Algorithms*. Springer-Verlag, New York, 2006.

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

[20] T. Tušar and B. Filipič. Differential evolution versus genetic algorithms in multiobjective optimization. *Lect. Notes Comput. Sc.*, 4403:257–271, 2007.

[21] R. Vesanen. *Resource Allocation in Spatially Distributed Communication Networks Using Subgradient Methods*. M.Sc. Thesis, University of Oulu, Department of Mathematical Sciences, Oulu, 2010 (in Finnish).

[22] H. Yaïche, R. R. Mazumdar, and C. Rosenberg. A game theoretic framework for bandwidth allocation and pricing in broadband networks. *IEEE/ACM T. Network.*, 8(5):667–678, 2000.

OPTIMIZATION IN ORGANIZATIONS: THINGS WE TEND TO FORGET

Uroš Bole
Pikarp d.o.o., Nova Gorica, Slovenia
uros.bole@gmail.com

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

Abstract The implications of bringing stochastic optimization of complex systems (SOCS) into an organization are still poorly understood. There appears to be a gap between technical and business people that causes a problem when introducing SOCS in an organization. As one of the ways to combat the gap we propose identification of critical success factors. Importantly we also identify, as the root cause of the gap, the fact that any SOCS initiative is interdisciplinary. If not managed correctly this potential source of creativity, may produce many failures to bridging the gap. In addition we put forth the need to think of SOCS in terms of its integration in organizations as opposed to the currently prevailing thinking in terms of individual projects.

Keywords: Management, Organizations, Stochastic optimization.

1. Introduction

Stochastic optimization has a significant role in the analysis, design, and operation of modern complex systems. Stochastic methods are able to cope with systems that are highly nonlinear, high dimensional, or otherwise inappropriate for classical deterministic methods of optimization [32]. Stochastic optimization algorithms have broad application to problems in statistics, science, engineering, and business. Algorithms that employ some form of stochastic optimization have become widely available. For example, many modern data mining packages include methods such as simulated annealing and genetic algorithms as tools for extracting patterns in data.

As mentioned in [10] the gap between data-mining and business is a real problem that has yet to be addressed [12, 26, 29, 38]. Similarly, stochastic optimization of complex systems (SOCS) process has many obstacles when entering the world of business [18, 20, 59]. There are currently no attempts of closing this gap to be found in SOCS literature.

Our previous research [10] shows that the interdisciplinary collaboration of technical experts and business people is at the root of this problem. Such initiatives should be a source of creativity [9, 42, 56]. However, when inappropriately managed, they often lead to failure [60].

The implications of the interdisciplinary nature of SOCS in organizations have not been researched yet. Hence a new methodology for integration of SOCS in organizations is needed. To design it, researchers will need a comprehensive set of criteria to guide them. We therefore identify the critical success factors (CSFs) of SOCS integration management. Moreover, organizations, and management in particular, may use the CSFs to focus on the areas where things must go right to successfully integrate it.

In Section 2 we define the problem of the gap in light of interdisciplinary collaboration. We describe the methodology for information gathering and analysis in Section 3. In Section 4 we identify candidate CSFs and discuss the implications of interdisciplinary collaboration, while in Section 5 we evaluate the CSFs.

2. Problem Definition

The gap between SOCS and business [18, 20, 59] has yet to be addressed. In data mining the gap is manifested in the accentuated focus of the technical community on algorithms and theory [26, 28]. SOCS presents a similar challenge. Technical experts and business people tend to see the same reality differently [29, 31]. The gap originates from the fact that business goals are based in reality, whereas SOCS is concerned with a simulation expressed in the evaluation function, which is an imperfect representation of that reality. Due to this difference, SOCS and business knowledge is required at every step of the SOCS integration process. Therefore, collaboration of experts with business knowledge and experts with SOCS knowledge is required to successfully manage SOCS integration in organizations.

Interdisciplinary collaboration requires more than the acquisition of explicit knowledge from the "other" discipline. People from different disciplines have different mental models that are at the root of the differences in the way we see, comprehend, express, approach, and solve the same problems [56]. Mental models or modes of cognition are sets

of basic assumptions about the world and how it works that facilitate dealing with information overload [2, 58]. Mental models depend on an individual's cultural, educational, and professional background [9, 24]. As these become more diverse, communication and collaboration between representatives of different disciplines becomes more difficult and often results in break-downs [60]. We therefore propose to look at the problem of the gap from the perspective of its root cause - collision of two different disciplines (optimization and business).

3. Research Methodology

CSF methodology is widely used for research and management. It can be used for most organizational initiatives that need to be managed for success [39, 48, 57]. CSFs are the few key areas in which things must go right for an initiative to succeed in attaining its goals [11].

The method for CSF data collection and analysis should fit the peculiarities of the case being addressed [8]. We chose the grounded theory research strategy as it is particularly helpful for research in business and management to predict and explain behavior [33]. Below we explain the strategies chosen for data collection and the data analysis procedure.

SOCS literature search. Observing the criterion of completeness, we draw from organizational in addition to SOCS research. In the SOCS literature we observe scarcity of management-oriented view of the SOCS integration in organizations. However, it serves as the source of organizational topics relevant to SOCS integration (as shown in Table 1).

Action research. To widen the number of resources we also used the action research (AR) strategy for data collection. AR allows the investigator to be a part of the organization and the change process [15]. As such, it is suitable for researchers concerned with the development of theory [55]. The goal was to observe (collect data) and reflect on the SOCS integration process while helping client organizations resolve their SOCS problems.

We chose several industrial projects to satisfy the criterion of generality. We selected five client organizations: Gorenje (AR1), Domel (AR2), Eta (AR3), Hydria AET (AR4), UKC (AR5). One organization is large (household appliances), three are medium-sized industrial producers (electrical motors, household appliances parts, automotive parts), and one is a large hospital (non-for-profit organization). SOCS was applied to solve different problems (product design for efficiency, quality control, product improvement, manufacturing process improvement, a recommendation system, and diagnosis support system development).

Organization literature search. Based on managerial and organizational issues which emerged in SOCS literature we reviewed organizational research sources as shown in Table 1.

Table 1. The organization science topics as suggested in the literature.

Topic	SOCS literature and AR projects	Organization science literature
Leadership, General Management	[18, 20]	[21, 22, 23, 30, 53]
Information Systems	[16, 25, 40], AR3	[34, 44, 48, 54, 58, 62]
Change Management and BPR	[18, 45, 59, 65]	[1, 4, 13, 47, 61]
Project Management	[16, 63]	[50]
Decision Making	[18, 20, 59], AR5	[7, 19]
Innovation, R&D	[20], AR2, AR4	[14, 27, 34, 42, 43, 52]
Human Resources Management	[20], AR3	[5, 24, 49, 51]
Learning, Knowledge Management	[18], AR1, AR3	[3, 6, 11, 37, 46, 56, 64]

Analysis. Grounded theory involves the following procedures of analysis: (i) open coding, to disaggregate data into units; (ii) axial coding, to recognize relationships between categories; and (iii) selective coding, to integrate the categories and produce a theory [17]. We began following this procedure iteratively in 2008 to unify topics, further explore the promising ones, and incorporate new and relevant findings.

4. Identification of Candidate CSF

We identified eight candidate CSFs in SOCS literature (upper part of Table 2). The analysis of organizational literature confirmed their validity (lower part of Table 2). However, we found evidence for reducing the number of candidate CSFs to six (see CSFs description in Section 5).

Common language and **time perception management** become **interdisciplinary learning**. Given the discussion of the implications of interdisciplinary collaboration, we merged common language and time perception management into interdisciplinary learning. Our review of the organizational literature suggests that the two candidate CSFs address the same problem – different mental models. Interdisciplinary learning implies awareness of the risks inherent in collaborative initiatives. Moreover, it suggests the need for the management of SOCS in this respect. Organizations may develop their own methods of coordination of SOCS as an interdisciplinary initiative. Alternatively, it might prove beneficial for the future penetration and integration of SOCS in organizations to design a general method with the goal of leveraging the

Table 2. Identified candidate CSFs (header) in the SOCS literature and action research (upper part), and confirmation of the CSFs in the organizational literature (lower part).

		business champion	relevance to business	clearly articulated problem	time perception management	interpersonal skills	common language	stakeholder support	change management
SOCS research	[18]	*	*	*	*	*	*	*	*
	[20]	*	*	*	*	*	*	*	*
	[59]	*	*	*	*	*	*	*	*
	[63]	*	*	*	*	*	*		*
	[45]	*	*	*				*	*
	[65]	*	*	*				*	
	AR1	*	*	*	*				
	AR2	*	*	*					
	AR3	*	*	*	*		*		
	AR4	*	*	*	*	*			
	AR5	*	*			*	*		
organizational research	[13]	*	*	*	*	*	*	*	*
	[34]	*	*	*	*	*	*	*	*
	[1]	*	*	*		*	*	*	*
	[24]	*	*	*	*	*	*	*	
	[56]	*	*	*	*	*	*	*	
	[47]	*	*	*		*	*	*	*
	[36]		*	*		*	*	*	*
	[50]		*	*		*	*	*	*
	[21]	*	*	*				*	*
	[44]		*	*		*		*	*
	[49]		*	*		*	*	*	
	[27]	*			*	*	*	*	
	[43]	*	*		*			*	
	[61]	*	*					*	*
	[30]		*			*	*	*	
	[9]			*	*	*	*		
	[3]	*		*		*		*	
	[23]		*	*				*	
	[19]		*	*		*			
	[7]		*	*		*			
	[42]				*	*		*	
	[41]		*	*		*			
	[46]		*	*		*			

upsides of the cognitive gap between SOCS experts and business people while minimizing its downsides.

Clearly articulated problem and **relevance to business** become **focus on problem-solving action**. It is difficult to achieve clearly articulated problem as expected by SOCS experts. Problems may be classified as structured, or not [46]. Unstructured, or "wicked" [36], problems are those in which it is not possible to determine a priori what the elements of a satisfactory solution are. A decision maker may only perceive the suitability of a solution (relevance to business) once it is stumbled upon. In such cases the definition of the problem evolves. It is itself a process. Similarly, a solution is found as a result of an innovative and creative discovery process [46], i.e. an emerging knowledge process (EKP) [35, 44].

EKP exhibits three characteristics: there is no best structure or sequence; there are complex requirements for knowledge, which is distributed across people and evolves dynamically; and it requires an actor set that is unpredictable in terms of job roles or prior knowledge [44]. SOCS in organizations exhibits these characteristics to a large extent. Hence, in SOCS initiatives, clearly articulated problem and relevance to business are interrelated and likely to evolve over time. The principles of EKP should be observed. Both business people and SOCS experts need to be aware that they will have to actively participate in an uncertain, recursive, and evolutionary process, starting with problem definition [9, 44, 46].

The purpose of information systems is not information, new knowledge, or a decision in itself, but to solve real business problems [23]. Like any problem-solving activity, SOCS only becomes relevant to a business when it produces a problem-solving action [34, 47]. Therefore, a key component of problem definition is that it is carried out with deployment in mind, i.e. how the end-user will act based on the information from the model. We therefore merged candidate CSFs *relevance to business* and *clearly articulated problem* into one: a *focus on problem-solving action.*

5. The CSFs Overview

The final six candidate CSFs identified in the previous section are further described below.

Business champion. A business champion is a strong and engaged business person who is confident in the business potential of SOCS. The presence of a business champion is a basic condition for starting and completing a SOCS initiative. A business champion should be viewed

as the primary generator of business demand for SOCS and the first person to be interested in the success of a SOCS initiative. He is instrumental in guaranteeing access to data and to subject matter experts. In addition, the sponsor helps manage organizational politics, stakeholder insecurities, power games, inflated expectations, etc.

Focus on problem-solving action. Having the problem clearly identified and understood in the same way by everyone presents difficulties. People from different disciplines interpret things differently. It is helpful to apply methods of interdisciplinary learning and to keep deployment in mind from the start, which may be done by assessing potential costs and benefits early and frequently. This helps secure relevance to business and helps to establish priorities between SOCS and other business endeavors. The SOCS process, from problem definition to solution is uncertain, iterative, and evolutionary. It requires active participation of the stakeholders.

Interdisciplinary learning. SOCS in organizations is an interdisciplinary initiative. The stakeholders need to be aware of its implications beyond explicit knowledge, otherwise the problem solving process can be fraught with difficulties and conflicts that often result in failure. Different disciplines imply different mental models. To leverage them as a source of creativity, organizations must develop methods of coordination of interdisciplinary collaboration. With time the constituents develop shared cognition that enables progressively more efficient and effective problem solving.

Interpersonal skills. SOCS involves people. Therefore, the interpersonal skills of all involved are the foundation on which the problem solving process depends. SOCS experts need to be able to work in a team, i.e. be humble, communicate well, possess educational and sales skills, etc. They should expect to encounter and deal with organizational politics.

Stakeholder support. It is a key enabler of SOCS in organizations. Like any problem-solving activity, SOCS generates changes that some stakeholders will likely resist unless they share the need to solve the problem. Stakeholder support reflects their awareness and confidence in SOCS, and their understanding of how SOCS is relevant to their business. This often requires a change in mindset facilitated by interdisciplinary learning and results in internal commitment, rather than mere compliance. Besides SOCS experts, the stakeholders may be management, other business people, subject matter experts, IT, and final users.

Change management. Like any problem-solving activity SOCS likely causes changes in the way of thinking or in the way of doing

things. Because changes may be significant, effective change management is needed. Without it the SOCS initiative is likely to run in different kinds of resistance by different stakeholders. They all have the power to kill a project. One of the aspects of change management is embedding of the optimization model into the information system of the organization. It greatly facilitates change if the implementation does not cause major disruptions in operations and avoids excessive additional work to stakeholders.

6. Conclusion

We reviewed relevant literature to propose candidate CSFs of SOCS. They are intended to support the management of SOCS integration in organizations. They are based on the fact that the gap between SOCS and business is fundamentally a problem of interdisciplinary collaboration between SOCS experts and business people. The CSFs may be used for SOCS management by helping stakeholders focus on the right issues.

SOCS requires specific methods to facilitate the management of its integration. Such methods are currently unavailable. Future research on SOCS in organizations should propose new methodologies to leverage the upsides of interdisciplinary collaboration and avoid its downsides. The CSFs may also be applied as the criteria against which to build and evaluate such methodologies.

The final contribution for SOCS research and practice is the emphasis on the distinction between a SOCS project and SOCS integration. The latter is the process of closing the gap between SOCS and business. It is composed of shorter SOCS projects and has the fundamental aim of building organizational capacity for future results. Hence, the integration needs to take a longer-term view compared to that of a project.

In the future we plan to carry out research to validate the CSFs, possibly through interviews with SOCS practitioners and/or through additional action research. In addition we plan to design a methodology to aid business people and SOCS practitioners alike in the effort to integrate SOCS in organization.

Acknowledgements

Operation part financed by the European Union, European Social Fund.

References

[1] M. Al-Mashari and M. Zairi. BPR Implementation Process: An Analysis of Key Success and Failure Factors. *Business Process Management Journal*, 5(1):87–

112, 1999.

[2] C. Argyris. Organizational learning and management information systems. *Account. Org. Soc.*, 2(2):113–123, 1977.

[3] C. Argyris. *Reasoning, Learning, and Action: Individual and Organizational.* Jossey-Bass, 1982.

[4] C. Argyris. *Knowledge for Action: A Guide to Overcoming Barriers to Organizational Change.* Jossey-Bass Publishers, 1993.

[5] C. Argyris and D. A. Schön. *Theory in practice: Increasing professional effectiveness.* Jossey-Bass, 1974.

[6] C. Argyris and D. A. Schön. *Organizational Learning II: Theory, Method, and Practice.* Addison-Wesley, 1996.

[7] M. Á. Ariño and P. Maella. *Iceberg a la vista: Principios para tomar decisiones sin hundirse.* Ediciones Urano, 2009.

[8] F. Bergeron and C. Bégin. The Use of Critical Success Factors in Evaluation of Information Systems: A Case Study. *J. Manage. Inform. Syst.*, 5(4):111–124, 1989.

[9] R. J. Boland Jr. and R. V. Tenkasi. Perspective Making and Perspective Taking in Communities of Knowing. *Organ. Sci.*, 6(4):350–372, 1995.

[10] U. Bole, J. Jaklič, J. Žabkar, and G. Papa. Identification of important factors to success of organizational data mining. In *Proc. 15th Portuguese Conference on Artificial Intelligence*, pages 535–549, 2011.

[11] C. Bullen. Reexamining Productivity CSFs: The Knowledge Worker Challenge. *Inform. Syst. Manage.*, 12(3):13–18, 1995.

[12] L. Cao. Domain-Driven Data Mining: Challenges and Prospects. *IEEE T. Knowl. Data En.*, 22(6):755–769, 2010.

[13] J. Chamberlin. Business Process Reengineering - a Retrospective Look. *Management Services*, 54(1):13–21, 2010.

[14] R. L. Chapman and M. Corso. From continuous improvement to collaborative innovation: the next challenge in supply chain management. *Prod. Plan. Control*, 16(4):339–344, 2005.

[15] D. Coghlan and D. T. Brannick. *Doing Action Research in Your Own Organization*, Sage Publications, 2004.

[16] G. Consigli and M. A. H. Dempster. Dynamic stochastic programming for asset-liability management. *Ann. Oper. Res.*, 81:131–161, 1998.

[17] J. M. Corbin and J. Corbin. *Basics of Qualitative Research: Techniques and Procedures for Developing Grounded Theory*, Sage Publications, 2008.

[18] T. H. Davenport and J. G. Harris. *Competing on Analytics: The New Science of Winning*, 1st edition. Harvard Business School Press, 2007.

[19] T. H. Davenport. Make Better Decisions. *Harvard Bus. Rev.*, 117–123, Nov. 2009.

[20] T. H. Davenport, J. G. Harris, and R. Morison. *Analytics at Work: Smarter Decisions, Better Results.* Harvard Business Press, 2010.

[21] W. E. Deming. *Out of the Crisis.* MIT Press, 2000.

[22] W. E. Deming. *The New Economics for Industry, Government, Education*, 2nd edition. MIT Press, 2000.

[23] P. F. Drucker. *Management Challenges for the 21st Century*. Butterworth-Heinemann, 1999.

[24] E. Du Chatenier, J. A. A. M. Verstegen, H. J. A. Biemans, M. Mulder, and O. Onno. The Challenges of Collaborative Knowledge Creation in Open Innovation Teams. *Human Resource Development Review*, 8(3):350–381, 2009.

[25] W. W. Eckerson. *Predictive Analytics, Extending the Value of Your Data Warehousing Investment*. TDWI Best Practices Report, 2007.

[26] J. Elder IV. *Successes, Failures and Learning from Them*. 2007. Available: `http://videolectures.net/kdd07\_elder\_sfle/`. (Accessed: 07-Mar-2011).

[27] J. Fagerberg. Innovation: A Guide to the Literature. In J. Fagerberg, D. Mowery, and R. Nelson (Eds). *The Oxford Handbook of Innovation*, Oxford University Press, 2005.

[28] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. From Data Mining to Knowledge Discovery in Databases. *AI Magazine*, 17:37–54, 1996.

[29] U. Fayyad. The Data Miner's Story - Getting to Know the Grand Challenges. In *Proc. KDD Conference*, 2007. Available: `http://videolectures.net/kdd07\_fayyad\_dms/`. (Accessed: 17-Jun-2010).

[30] P. Ferreiro and M. Alcázar. *Gobierno de personas en la empresa*, 1st edition. Editorial Ariel, 2002.

[31] F. Fogelman Soulié. Industrial Data Mining, Challenges and Perspectives. In *Proc. ECML PKDD Conference*, 2008. Available: `http://videolectures.net/ecmlpkdd08\_soulie\_idmc/`. (Accessed: 28-Jun-2010).

[32] J. E. Gentle, W. Härdle, and Y. Mori. *Handbook of Computational Statistics: Concepts and Methods*. Springer, 2004.

[33] C. Goulding. *Grounded Theory: A Practical Guide for Management, Business and Market Researchers*. Sage Publications, 2002.

[34] W. L. Harkness, W. J. Kettinger, and A. H. Segars. Sustaining process improvement and innovation in the information services function: lessons learned at the Bose corporation. *MIS Quart.*, 20(3):349–368, 1996.

[35] H. Hasan. Information Systems Development as a Research Method. *Australasian Journal of Information Systems*, 11(1):4–13, 2003.

[36] A. Hevner, S. March, J. Park, and S. Ram. Design Science in Information Systems Research. *Management Information Systems Quarterly*, 28(1):75–106, 2004.

[37] W. N. Isaacs. Taking flight: Dialogue, collective thinking, and organizational learning. *Organ. Dynam.*, 24–39, Winter 1993.

[38] T. Khabaza. *9 Laws of Data Mining*. 2010. Available: `http://khabaza.codimension.net/index\_files/9laws.htm`. (Accessed: 10-Mar-2011).

[39] C. Kimble and I. Bourdon. Some success factors for the communal management of knowledge. *Int. J. Inform. Manage.*, 28(6):461–467, 2008.

[40] E. A. King. How to Buy Data Mining, A Framework for Avoiding Costly Project Pitfalls in Predictive Analytics. *Information Management Magazine*, Oct. 2005.

[41] S. Lichtenstein, D. Bendall, and S. Adam. Marketing Research and Customer Analytics: Interfunctional Knowledge Integration. *International Journal of Technology Marketing*, 3(1):81–96, 2008.

[42] J. W. Lorsch and P. R. Lawrence. Organizing for Product Innovation. *Harvard Bus. Rev.*, 109–120, Jan.-Feb. 1965.

[43] E. Mansfield. How Economists See R&D. *Harvard Bus. Rev.*, November-December:98–106, 1981.

[44] M. L. Markus, A. Majchrzak, and L. Gasser. A Design Theory for Systems That Support Emergent Knowledge Processes. *MIS Quarterly*, 26(3):179–212, 2002.

[45] L. T. Moss and S. Atre. *Business Intelligence Roadmap: The Complete Project Lifecycle for Decision-Support Applications.* Addison Wesley, 2003.

[46] B. Muñoz-Seca and J. Riverola. *Gestion del Conocimiento.* Ediciones Folio, 1997.

[47] D. J. Paper, J. A. Rodger, and P. C. Pendharkar. A BPR Case Study at Honeywell. *Business Process Management Journal*, 7(2):85–99, 2001.

[48] K. Peffers, C. E. Gengler, and T. Tuunanen. Extending Critical Success Factors Methodology to Facilitate Broadly Participative Information Systems Planning. *J. Manage. Inform. Syst.*, 20(1):51–85, Jul. 2003.

[49] J. A. Pérez López, *Teoría de la acción humana en las organizaciones: la acción personal.* Ediciones Rialp, 1991.

[50] J. K. Pinto and J. G. Covin. Critical Factors in Project Implementation: A Comparison Of Construction and R&D Projects. *Technovation*, 9(1):49–62, 1989.

[51] B. Renzl. Trust in management and knowledge sharing: The mediating effects of fear and knowledge documentation. *Omega*, 36(2):206–220, 2008.

[52] T. Ritter and H. G. Gemünden. The Impact of a Company's Business Strategy on Its Technological Competence, Network Competence and Innovation Success. *J. Bus. Res.*, 57(5):548–556, 2004.

[53] J. F. Rockart. Chief executives define their own data needs. *Harvard Bus. Rev.*, 52(2):81–93, 1979.

[54] J. F. Rockart and A. D. Crescenzi. Engaging top management in information technology. *MIT Sloan Manage. Rev.*, 24(1):3–13, 1982.

[55] M. Saunders, P. Lewis, and A. Thornhill. *Research Methods for Business Students*, 5th edition. Financial Times/Prentice Hall, 2009.

[56] P. Senge. *The Fifth Discipline: The Art and Practice of the Learning Organization*, 2nd edition. Doubleday Business, 2006.

[57] M. H. Shah and F. A. Siddiqui. Organisational critical success factors in adoption of e-banking at the Woolwich bank. *Int. J. Inform. Manage.*, 26(6):442–456, 2006.

[58] H. A. Simon. *The Sciences of the Artificial.* MIT Press, 1996.

[59] J. Taylor and N. Raden. *Smart Enough Systems: How to Deliver Competitive Advantage by Automating Hidden Decisions.* Prentice Hall, 2007.

[60] J. Tidd, J. Bessant, and K. Pavitt. *Managing Innovation: Integrating Technological, Market and Organizational Change*, 3rd edition. Wiley, 2005.

[61] P. Trkman. The critical success factors of business process management. *Int. J. Inform. Manage.*, 30(2):125–134, 2010.

[62] E. Turban, D. Leidner, E. McLean, and J. Wetherbe. *Information Technology for Management: Transforming Organizations in the Digital Economy.* Wiley, 2005.

[63] S. Viaene and A. Van Den Bunder. The Secrets to Managing Business Analytics Projects. *MIT Sloan Manage. Rev.*, 53(1):64–70, 2011.

[64] H. Wang and S. Wang. A knowledge management approach to data mining process for business intelligence. *Ind. Manage. Data Syst.*, 108(5):622–634, 2008.

[65] S. Williams and N. Williams. *The Profit Impact of Business Intelligence.* Morgan Kaufmann, 2006.

NMPC AND GENETIC ALGORITHM BASED APPROACH FOR TRAJECTORY TRACKING OF UAVS

Luca De Filippis, Giorgio Guglieri
Department of Mechanics and Aerospace, Politecnico di Torino, Italy
{luca.defilippis; giorgio.guglieri}@polito.it

Abstract Research on unmanned aircraft is improving constantly the autonomous flight capabilities of these vehicles in order to provide performance needed to employ them in even more complex tasks. UAV Path Planning (PP) system plans the best path to perform the mission and then it uploads this path on the Flight Management System (FMS) providing reference to the aircraft navigation. Tracking the path is the way to link kinematic references related to the desired aircraft positions with its dynamic behaviors, to generate the right command sequence. This paper presents a Nonlinear Model Predictive Control (NMPC) system that tracks the reference path provided by PP, solving on-line (i.e. at each sampling time) a finite horizon (state horizon) open loop optimal control problem with Genetic Algorithm (GA). The GA finds the command sequence that minimizes the tracking error with respect to the reference path, driving the aircraft toward the desired trajectory.

Keywords: Genetic algorithms, Model predictive control, Trajectory tracking, UAV.

1. Introduction

Unmanned Aerial Vehicles (UAV) represent one of the most studied field of research in aeronautics and robotics. Characteristics and performance of these aircraft excited wide industrial and academic engagement to improve their autonomy in order to cope with even more complex missions. Planning the path and control the UAV in order to follow the desired trajectory accomplishing with its mission is a challenging task. Different PP and tracking systems have been developed, that exploit wide range of techniques providing encouraging results. However UAV dynamics is nonlinear and control systems able to optimize aircraft performance are desirable in order to plan complex trajectories and in turn solve challenging tracking problems.

In the last decades wide research has been done on Receding Horizon Control (RHC) techniques [5] to cope with: a) intrinsically nonlinear dynamic systems, b) high quality requirements, c) growing use of robotic systems in any working division. Linear models are not sufficient to describe adequately any dynamic system particularly when performance close to constraint boundaries are desirable. NMPC allows the use of accurate models and more complex problem formulations that provide better prediction and optimization [1]. Deep investigation is carried on to consolidate theoretical background and to proof fundamental properties of NMPC (i.e. feasibility, stability and robustness). In ground and flight robotics many applications to tracking and collision avoidance problems can be found in literature. Sprinkle et al. [11] presented a NMPC system applied to trajectory tracking for persuit/evasion games between two fixed wing UAVs. This control technique works on the "planning" level. In other words an optimal trajectory is provided to the autopilot in order to perform mission tasks. Our interest is on the other hand in a control technique able to work at lower level generating optimal commands so that the desired path is tracked with the UAV. Kang et al. [8] implemented an interesting NMPC system to cope with trajectory tracking problems. They designed a high-level tracking controller for a small fixed-wing UAV and they studied close-loop stability extracting some performance properties of the control strategy adding an outer loop to the inner control loop. Whether theoretical and mathematical support is fundamental to convert a simple problem solving approach in a deeper investigation able to provide general concepts on this control technique, on the other hand this approach to the problem requires simplifications that could mismatch with a real implementation problem.

Genetic algorithms (GA) are optimization techniques based on biological principles of natural selection. They are able to cope with any kind of problem even when its features are not completely understood. Another important merit of evolutionary optimization is the wide range of potential solutions that the algorithm is able to evaluate with respect to classic formulations [7]. These features make GA particularly useful when optimization problems like the one met in NMPC formulations with large state and solution spaces need to be solved. Tian et al. [12] combine MPC with GA to implement an algorithm for UAV cooperative search. The authors subdivide the environment with hexagonal cells and use MPC to predict future states of the UAV swarm in order to minimize searching area through the best aircraft distribution over it.

This paper describes a novel approach to trajectory tracking that matches GA features with NMPC. To the authors knowledge applications of NMPC tested on real problems exploit model-predictive to tackle

collision avoidance and formation flight. In these works predictive features are needed to generate optimized trajectories to avoid unpredicted obstacles or coordinate the path of UAV swarms preventing collisions. On the other hand studies where MPC is applied to trajectory tracking are oriented to investigate the predictive-control properties in terms of stability and robustness, simplifying the reference trajectory and the tracking strategy in favor of theoretical contributions. Then the tracking system described here aims to cope with real applications where the PP system cooperates with the lower level navigation system in order to steer the UAV over the best trajectory in complex and performance demanding conditions.

The non-deterministic approach of evolutionary optimization together with critical time constraints for trajectory tracking of UAVs discouraged applications of NMPC with GA. However new theoretical contributions and wide experimental results obtained in these years encouraged the authors to fuse these two techniques. As a matter of fact gradient-based optimization, commonly used on MPC, is robust and theoretically solid but it forces to simplify the problem formulation, preventing the application to more complex and critical tracking tasks. SBX crossover and Polynomial mutation are chosen as genetic operators for the GA here described. SBX is a well known crossover method developed by Deb et al. [2] in 1995, more advanced solutions were developed in the following years. Ono and Kobayashi [9] implemented the Unimodal Normally Distributed Crossover operator (UNDX) in 1997, where three parent solutions are used to create two or more children solutions. Then Herrera and Lozano [6] introduced real coded cross-over based on fuzzy connectives in 1999 and in 2002 Deb et al. [3] proposed another real coded crossover called Parent Centric Crossover operator (PCX). Even if more advanced crossover operators are described in literature, providing comparison about their performance. SBX is still used in specific applications linked to path planning of UAV [10] and it is chosen here as a preliminary approach that will be improved with further investigations.

2. Reference Path

The higher-level PP system that provides the path the aircraft must follow exploits Kinetamic A* algorithm [4]. The output of the PP system is a sequence of waypoints used as a reference in order to steer the aircraft toward the path. Kinematic A* (KA*) implements the graph search logic to generate feasible paths and introducing basic vehicle characteristics to drive the search. It includes a simple kinematic model of the vehicle to evaluate the moving cost between waypoints in a three-dimensional

environment. Movements are constrained with minimum turn radius and the maximum rate of climb.

The aircraft model used to generate the waypoints sequence is given by equations:

$$\begin{cases} \dot{X} = V \cdot \cos\chi \cdot \cos\gamma_{max} \cdot w \\ \dot{Y} = V \cdot \sin\chi \cdot \cos\gamma_{max} \cdot w \\ \dot{Z} = V \cdot \sin\gamma_{max} \cdot w \\ \dot{\chi} = \dfrac{V}{R} \cdot u \end{cases} \tag{1}$$

where (X, Y, Z) is the position vector and V is the constant ground speed of the aircraft. Command variable w $(-1 \leq w \leq 1)$ modules the climb angle between its minimum and maximum values coincident with γ_{max}. The second command u $(-1 \leq u \leq 1)$ on the other hand modules the turn speed with respect to the minimum turn radius R.

KA* output is a waypoint sequence with each point represented by the state vector $(X, Y, Z, \gamma, \chi, V,)$. The NMPC system predicts future aircraft positions over the prediction horizon, then the tracking task is performed trying to reduce the error between predicted and reference positions. For each time step a receding fraction of the reference path is extracted by the full path and provided to the NMPC system as a reference. NMPC finds the optimal command which reduces the tracking error.

3. NMPC Formulation

NMPC acts solving a finite horizon open-loop optimal control problem in real-time. The cost function is the function minimized with the optimal commands. This function characterizes the problem containing variables that represent the optimization task. A classical quadratic function has been selected here made of two terms:

- **State error**: this term evaluates the error between the reference states and the predicted one and it is the term where predicted positions are compared with the desired one provided by KA*.
- **Command**: this term evaluates the amount of command needed to perform the predicted maneuver.

The cost function depends from initial states measured with sensors at each time-step and from a predicted control sequence. Indicating with $(^*)$ the predicted variables over the prediction horizon (T_p) the task is to find:

$$\min_{\bar{U}^*(\cdot)} J(\bar{X}_0, \bar{U}^*(\cdot)) = \int_t^{t+T_p} \hat{\bar{X}}^*(\tau)^T \boldsymbol{Q} \hat{\bar{X}}^*(\tau) + \bar{U}^*(\tau)^T \boldsymbol{R} \bar{U}^*(\tau) d\tau \tag{2}$$

with:

$$\hat{\bar{X}}^*(\tau) = \bar{X}^*(\tau) - \bar{X}_{ref}(\tau), \tag{3}$$

where $\bar{X}_0 = \bar{X}(t) \in \mathbb{R}^n$ is the initial-state vector and $\bar{U}^*(\cdot) \in \mathbb{R}^m$ is the predicted-command vector. Then $\bar{X}^*_\tau \in \mathbb{R}^n$ is the predicted-state vector and $\bar{X}_{ref}(\tau) \in \mathbb{R}^n$ is the reference-state vector. Finally $\boldsymbol{Q}$ and $\boldsymbol{R}$ are diagonal matrices of gains weighting state-variables effects over the cost function.

The command horizon T_c defines the horizon of optimal commands generated for each optimization loop and it commonly differs from T_p. The command strategy is then the other fundamental element to formulate NMPC. The classic command vector of an aircraft contains:

$$\bar{U} = \begin{cases} \delta_e, & \delta_{e_{min}} \leq \delta_e \leq \delta_{e_{max}} \\ \delta_a, & \delta_{a_{min}} \leq \delta_a \leq \delta_{a_{max}} \\ \delta_r, & \delta_{r_{min}} \leq \delta_r \leq \delta_{r_{max}} \\ Th, & 0 \leq Th \leq 1 \end{cases} \tag{4}$$

The command strategy is just the strategy to build the command signal over the command horizon and in general over the prediction horizon. A linear commands variation has been chosen here over the command horizon. Particularly a peacewise linear function is built over the prediction horizon based on functions:

$$\bar{U}^*_i(\tau) = \bar{U}^*_{i_0} + \bar{A}_i * \tau, \quad 1 \leq i \leq n_c, \tag{5}$$

where $\bar{U}^*_i(\tau)$ is the i^{th} linear function, $\bar{U}^*_{i_0}$ is the i^{th} initial command value and $\bar{A}_i$ is the i^{th} function slope.

The horizon of commands is then arranged subdividing the time interval in a number of steps (n_c) according with the command horizon and the command frequency(hz_c). As an example with $T_p = 2$ [s], $T_c = 1$ [s] and $hz_c = 2$ [1/s] two linear functions ($n_c = 2$) are built over the command horizon:

$$\begin{array}{ll} \bar{U}^*_1(\tau) = \bar{U}_0 + \bar{A}_1 * \tau, & t - 1 \leq \tau \leq T_c/2, \\ \bar{U}^*_2(\tau) = \bar{U}_{T_c/2} + \bar{A}_2 * \tau, & T_c/2 \leq \tau \leq T_c \end{array} \tag{6}$$

and the command over the time step $T_p - T_c = 1[s]$ given by:

$$\bar{U}^*(\tau) = \bar{U}^*_{n_c}(T_c), \quad T_c \leq \tau \leq T_p - T_c. \tag{7}$$

This command sequence has been chosen to guarantee continuity of command functions and in turn of external forces and couples acting on

aircraft. Commands generated with Eq. (5) are bounded with disequalities in (4) but $\bar{A}_i$ vector must be bounded too. Maximum command variation over unitary time-step is chosen such that:

$$\Delta\bar{U}^*_{min} \leq \bar{A}_i \leq \Delta\bar{U}^*_{max}, \tag{8}$$

$$\Delta\bar{U} = \begin{cases} \Delta\delta_{e_{min}} \leq \Delta\delta_e \leq \Delta\delta_{e_{max}} \\ \Delta\delta_{a_{min}} \leq \Delta\delta_a \leq \Delta\delta_{r_{max}} \\ \Delta\delta_{r_{min}} \leq \Delta\delta_r \leq \Delta\delta_{r_{max}} \\ \Delta Th_{min} \leq \Delta Th \leq \Delta Th_{max} \end{cases} \tag{9}$$

Solving online the optimization problem the linear-function slopes in Eq. (5) are chosen in order to minimize the cost function.

Defining the state vector as:

$$\bar{X} = \{u\ v\ w\ p\ q\ r\ \phi\ \theta\ \psi\ x_V\ y_V\ z_V\ \omega_p\} \tag{10}$$

the complete optimization problem prescribes to find:

$$\bar{U}^*_{opt} = f(\tau, \bar{U}_0, \bar{A}_{opt}, T_c, T_p, n_c), \tag{11}$$

so that Eq. (2) is satisfied according with system of Eqs. (4) and (9) but also with the aircraft nonlinear equations of motion:

$$\dot{\bar{X}} = f(\bar{X}, \bar{U}). \tag{12}$$

4. GA Algorithm

Referring to the problem formulation already presented the individuals representing the solutions are the aircraft command slopes and particularly the chromosomes of each individual are:

$$chromosome = [\Delta\delta e, \Delta\delta a, \Delta\delta r, \Delta Th], \tag{13}$$

while fitness function J described in Eq. (2).

Classic GA structure implemented to cope with this problem collects:

- **Population initialization**:

$$chromosome = \Delta\bar{U}_{min} + (\Delta\bar{U}_{max} - \Delta\bar{U}_{min}) \cdot \bar{R}d, \tag{14}$$

where $0 \leq \bar{R}d \leq 1$ is a random vector of 4 numbers. For each individual Eq. (14) provides the four genes composing the chromosome bounded between its minimum and maximum value. After initial population is build Eq. (2) provides the fitness value of each individual.

- **Selection**: to perform this task half of the total population is randomly selected. Two individuals are randomly chosen and the one with the lower cost is selected for reproduction. Then other two solutions are taken and the selection is repeated up to obtain a number of individuals equal to half of the total population. This technique is called *tournament selection* and it is preferable to select individuals sorted with respect to their fitness value because it introduces a random component into the selection logic.

- **Crossover**: Simulated Binary Crossover (SBX) [2] is chosen to exchange the genetic pool. This mechanism uses a probability distribution around two parents to create two children. Unlike other methods SBX uses a probability distribution similar to the probability typical of crossover operators used in binary-coded algorithms. The fundamental merit of SBX is its self-adaptive power that guarantees to the offspring to do not narrow near the previous optimal solution (typical phenomenon due to week diversity inside population). Then with SBX children closer to their parents are more likely to be created and the diversity inside the offspring is proportional to the one inside the previous generation. This is guaranteed fixing a distribution index η_c that can be any positive real number. Large values of η_c increase probability to have children close to their parents and vice-versa. Procedure presented with Algorithm 1 is implemented to create children from their parents with SBX.

Algorithm 1 Crossover

Chose a distribution index η_c
Chose a random number $u \in [0, 1)$
if $u \leq 0.5$ **then**
  $\beta_q = (2 \cdot u)^{\frac{1}{\eta_c+1}}$
else
  $\beta_q = (\frac{1}{2 \cdot (1-u)})^{\frac{1}{\eta_c+1}}$
end if
for i =1:4 **do**
  $g^i_{c_1} = 0.5 \cdot [(1+\beta q) \cdot g^i_{p_1} + (1-\beta q) \cdot g^i_{p_2}]$
  $g^i_{c_2} = 0.5 \cdot [(1-\beta q) \cdot g^i_{p_1} + (1+\beta q) \cdot g^i_{p_2}]$
end for

where $g^i_{c_j}$ is the i^{th} gene of the j^{th} child and $g^i_{p_j}$ is the i^{th} gene of the j^{th} parent.

- **Mutation**: a polynomial mutation technique based on probability distribution similar to the SBX one is implemented. Then reasons to chose this mutation operator are the same as for the crossover

one. Fixing a distribution index η_m from one individual one child is obtained with Algorithm 2.

Algorithm 2 Mutation

for i =1:4 **do**
  Chose a random number $u_i \in [0, 1)$
  if $u_i \leq 0.5$ **then**
    $\Delta g_c^i = (2 \cdot u)^{\frac{1}{\eta_m+1}} - 1$
  else
    $\Delta g_c^i = 1 - (2 \cdot (1 - u))^{\frac{1}{\eta_m+1}}$
  end if
  $g_c^i = g_p^i + \Delta g_c^i$
end for

where g_c^i and Δg_c^i are the i^{th} child gene and mutation. While g_p^i is the i^{th} parent gene.

- **Evaluation**: the fitness function to evaluate individuals is the cost function provided in Eq. (2).
- **Update**: the update scheme is performed joining old and offspring populations and sorting them with respect to the fitness value. Then a set of individuals equal to the population size is chosen and used for the next algorithm cycle.

The offspring population size is half of the total population and its composition is made with a fixed percentage of mutated individuals. As a matter of fact when the selection phase is complete mutation is performed up to obtain the prescribed percentage of individuals over the total amount then crossover is performed up to complete the offspring population.

Convergence condition is satisfied when the algorithm converges to the same solution for a prescribed number of times. In more details each time the population is updated the cost value of each individual is introduced inside a vector:

$$\bar{F} = [f_1, f_2, f_3, ..., f_{N-1}, f_N] \tag{15}$$

and the following equation inequality is evaluated:

$$f_{best} - \frac{\sum_{i=1}^{N-1} f_i}{N} \leq Toll, \tag{16}$$

where f_i is the cost linked to the i_{th} individual, f_{best} is the cost linked to the best individual, N is the population size and $Toll$ is a fixed tolerance, set to 10^{-15}.

When inequality (16) is verified, algorithm convergence to a local minimum is assumed (i.e. the whole population has the same average cost value). However the population is further updated to explore, with mutation operator, solution-space regions far from the one where the local minimum was found. If the algorithm converges to the same local minimum for a given number of times (i.e. five times for simulations presented in the results) full algorithm convergence is assumed.

5. Results

To implement this test Kinematic A* algorithm is exploited to generate the reference path on the DEM of a mountainous area. The area in the North of Italy is inside region "Valle d'Aosta" and it includes wide orographic obstacles. The aircraft is forced to climb and turn all along the path to maintain distance from ground because of continuous-obstacle distribution.

Genetic algorithm has 48 individuals that compose population and convergence tolerance fixed to 10^{-3}. NMPC has 40 Hz integration frequency, 1 Hz command frequency, 1 s integration horizon and 1 s command horizon. Simulation starts with the aircraft in trim condition and command bounds are ± 0.5 rad for elevator, aileron and rudder, $0 - 1$ for throttle. Command slopes are then ± 1 rad/s for each aerodynamic surface and ± 1 for throttle.

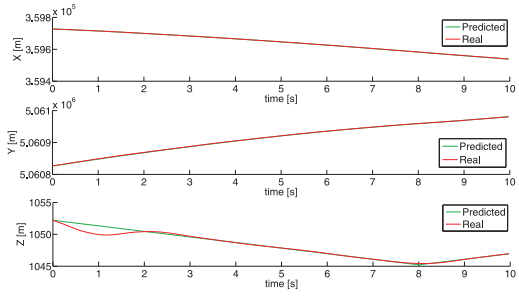


Figure 1. Comparison between reference and real path on each Ground axis.

Figure 1 represents the tracking error on the three axis that evidences altitude loss during first 2 seconds and that shows the system is able to track the reference path with high accuracy. The real trajectory in red completely overlaps the reference one in green. The error on the Z-axis is high when the simulation begins because of small trim-condition inaccuracies. However the altitude mismatch is just 2 meters and it is quickly reduced by the control system.

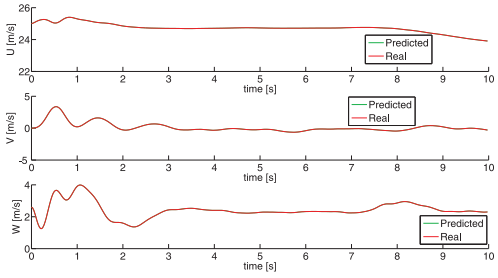


Figure 2. Comparison between predicted and real speed.

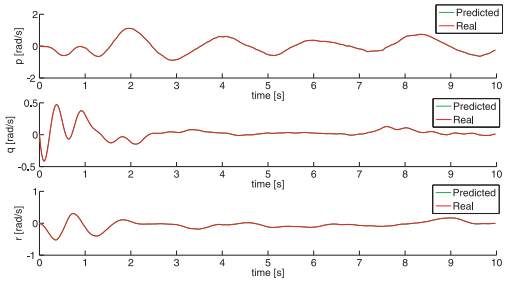


Figure 3. Comparison between predicted and real angular rate.

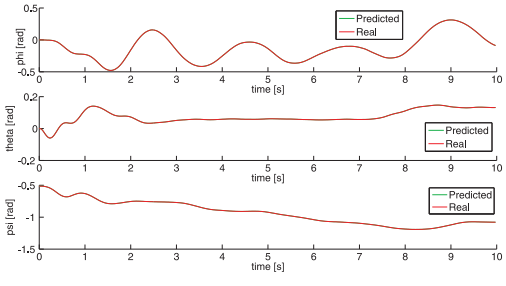


Figure 4. Comparison between predicted and real attitude.

Figures 2 and 3 collect time-histories for each B-axis of speeds and angular rates. Tuning cost-function gains, constraints on these state variables are imposed. Relative wind speed is compared with cruise speed introduced into the KA* model. The control system tries to keep this speed constant and equal to the reference. Coordinated turns are then imposed keeping to zero the later speed (Y-axis component) and

in turn the sideslip angle. Angular rates on the other hand are bounded in order to avoid aggressive maneuvers.

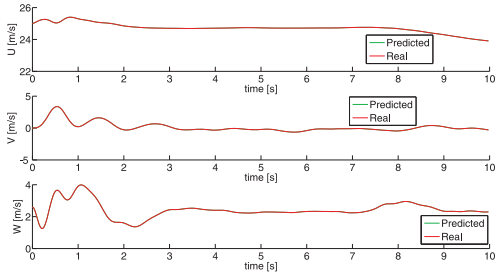


Figure 5. Comparison between predicted and real relative-wind speed, angle of bank and attack.

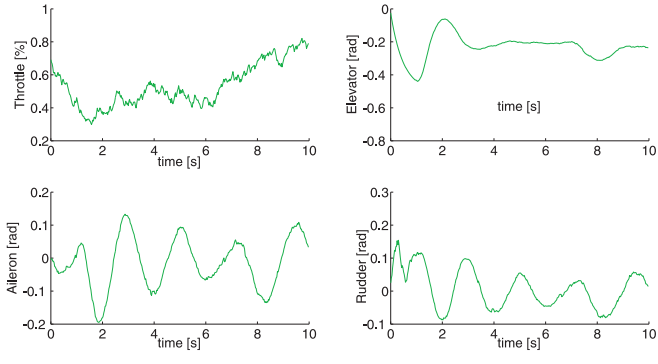


Figure 6. History of the optimal commands.

Aircraft attitude angles are represented in Fig. 4. Any reference is provided for these state variables and wind components plotted in Fig. 5 confirm performance previously described. Finally the command time-histories are shown in Fig. 6. Aileron deflections are bounded and linked to the rudder one providing coordinated turns. Throttle on the other hand is decreased to lose altitude and follow the descent. Then it is kept almost constant and increased to climb in the last 4 seconds. The elevator is quickly deflected up to the limit in order to compensate the altitude tracking error.

6. Conclusions

The tracking system proposed in this paper seems to reflect merits of NMPC and to accomplish with the task. As a matter of fact good tracking performance is evidenced with the results and effective control actions seem to provide smooth and safe paths. It must be stressed though that this is just the first implementation of this method and further improvements have been planned. Particularly the GA algorithm will be improved introducing modern genetic operators in order to optimize its convergence. On the other hand present results are sufficient to motivate further investigations and to confirm that model prediction is a powerful and robust technique particularly useful in these field of research.

References

[1] F. Allgöwer, R. Findeisen, and Z. K. Nagy. Nonlinear model predictive control: From theory to application. *J. Chin. Inst. Chem. Engrs.*, 35(3):299–315, 2004.

[2] K. Deb, and R.B. Agarwal. Simulated binary crossover for continuous search space. *Complex Syst.*, 9(2):115–148, 1995.

[3] K. Deb, A. Anand, D. Joshi. A computationally efficient evolutionary algorithm for real-parameter evolution. *Evol. Comput.*, 10(4):371–395, 2002.

[4] L. De Filippis and G. Guglieri. Advanced Graph Search Algorithms for Path Planning of Flight Vehicles. *Recent Advances in Aircraft Technology*, InTech - Open Access Publisher, 1:1–36, 2012.

[5] C. E. Garcia, D. M. Prett, and M. Morari. Model predictive control: theory and practice. *Automatica*, 25(3):335–348, 1989.

[6] F. Herrera, M. Lozano, J. L. Verdegay. Dynamic and heuristic fuzzy connectives based crossover operators for controlling the diversity and convergence of real-coded genetic algorithms. *Int. J. Intell. Syst.*, 11(12):1013–1040, 1996.

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

[8] Y. Kang and J. K. Hedrick. Linear tracking for a fixed-wing UAV using nonlinear model predictive control. *IEEE T. Contr. Syst. T.*, 17(5):1202–1210, 2009.

[9] I. Ono and S. Kobayashi. A real-coded genetic algorithm for function optimization using unimodal normal distribution crossover. In *Proc. Seventh International Conference on Genetic Algorithms*, pages 246–253, 1997.

[10] Y. V. Pehlivanoglu, O. Baysal, A. Hacioglu. Path planning for autonomous UAV via VGA. *Aircr. Eng. Aerosp. Tec.*, 79(4):352–359, 2007.

[11] J. Sprinkle, J. M. Eklund, H. J. Kim, and S. Sastry. Encoding aerial pursuit/evasion games with fixed wing aircraft into a nonlinear model predictive tracking controller. In *Proc. 43rd IEEE Conference on Decision and Control*, volume 3, pages 2609–2614, 2004.

[12] J. Tian, Y. Zheng, H. Zhu, and L. Shen. A MPC and genetic algorithm based approach for multiple UAVs cooperative search. *Lect. Notes Comput. Sc.*, 3801:399–404, 2005.

ARTIFICIAL NEURAL NETWORKS FOR DETECTION COVER AND STEGO IMAGES

Zuzana Oplatková, Jiří Hološka, Roman Šenkeřik
Faculty of Applied Informatics, Tomas Bata University in Zlín, Czech Republic
{oplatkova; holoska; senkerik}@fai.utb.cz

Abstract This paper is aimed on the technique for detection of cover and stego images by means of artificial neural networks. The work is connected with cryptography for information encoding and additional method for information hiding, which is called steganography. The core of the paper explains the principle of hidden information revealing in multimedia files like jpeg by means of artificial neural networks. The results show successful detection of four stego techniques by means of feed forward neural networks with one hidden layer.

Keywords: Artificial neural networks, Detection, Steganalysis.

1. Introduction

Within spreading of computers into human lives the need of security has arisen. One field, which covers developing of impossibility of secret message decoding, is called cryptography [3]. The other method connected with security hides information because of distraction from important messages. This method is called steganography [1, 16, 12]. Hiding of information is both useful and dangerous. Therefore it is important to develop tools and methods for forensic analysis to prevent from abusing hiding methods for criminal aims.

Steganography is an art of hiding communication by embedding of secret messages into innocent file content, mainly into multimedia files. Carrier files in steganography are called cover images, while files with hidden information embedded by some steganography technique are called stego files.

Steganography can be misused. Unwanted leaking of "know-how" or other confidential content is on the first place. The example of such a process can be described as follows.

Imagine a company with employees and secret information – e.g. a database with secret data is located on a database server accessible from the employee's terminal. If the employee decides to steal confidential data and uses a regular email (as seen in Fig. 1.) This action is revealed almost immediately because he has a monitored services email account. When the internal data is saved into a regular email and sent it by the department email account to a home computer then such email is checked.

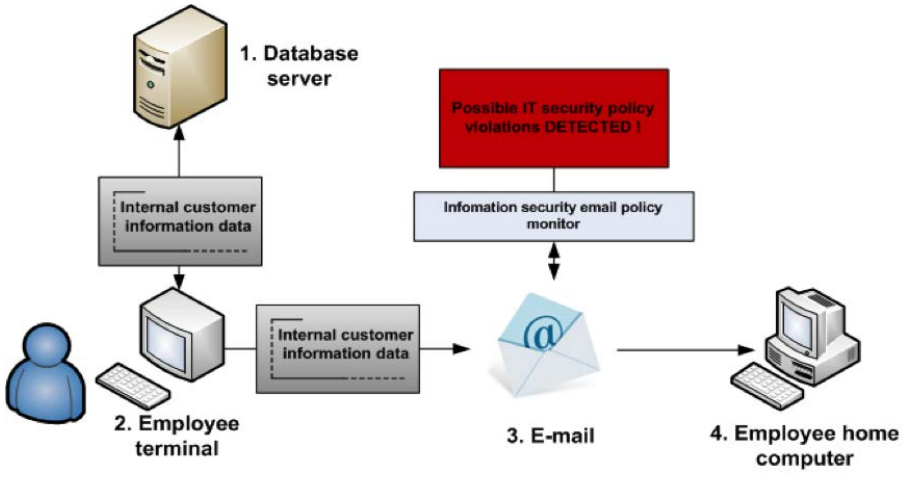


Figure 1. Message transport through plain text email.

There is an email monitor between the terminal and the email gateway which checks all-outgoing emails for viruses as well as its body plus attachments for internal business information. In the described case, the security monitor detects that an email attachment contains sensitive data. The security department is immediately informed about this incident and the employee is charged for the information fraud.

In Fig. 2 a similar scenario is shown. If the employee from this case decides to steal some confidential data from the employer's database it is not too difficult task with usage of a steganography tool. The steganography helps with the secure transfer of secret messages compared to cryptography, which is strong in the usage of the key and the message is coded. Steganography codes a message inside the images, video file or data stream. If a human eye sees a picture with steganographic content, he would not recognize the secret message inside. This is the main aim – to hide information itself.

The whole scenario of the second case with steganography tool is very easy. The employee can use e.g. a Java application downloaded

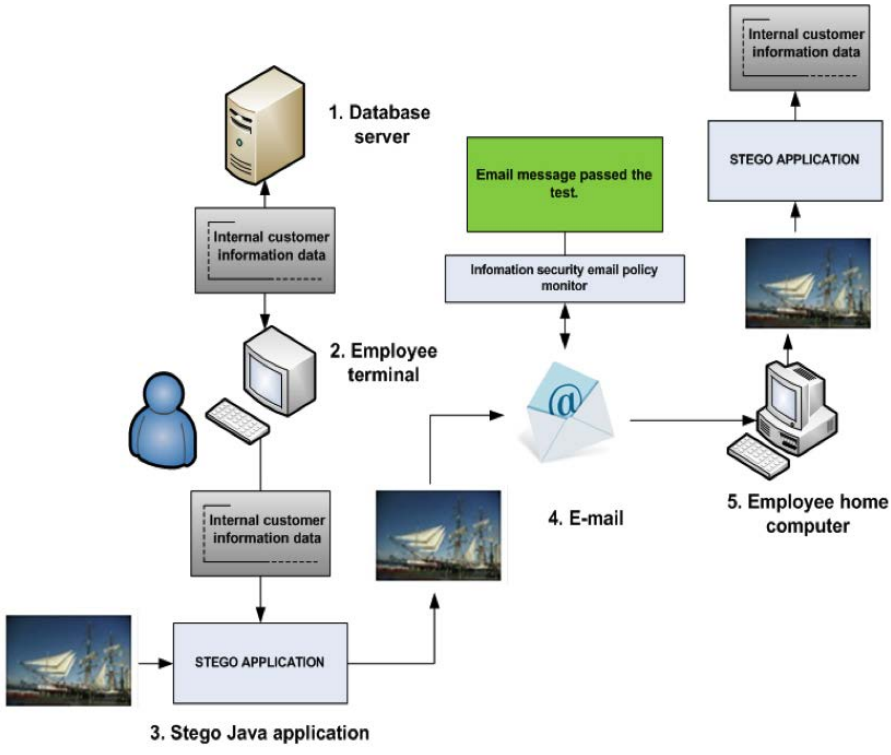


Figure 2. Message transport with steganography.

from the Internet because it is not possible to install any application on employee's terminal and Java is consider to be used for multi platforms. Then he has to prepare images in JPEG format and uses a steganography Java application, which embeds a text file containing internal business information into them. After that those images are sent to his home email account. The outgoing server does not point out any dangerous of information leaking.

Not to attract attention is the main goal of steganography. Therefore it is necessary to have a method for its detection because it is vulnerable to criminals. This research deals with such a case – method of detection by means of Artificial Neural Networks (ANN) [4].

The steganalysis techniques employ different ways of detection like statistical methods, searching specific steganography method signature [12]. Also methods of artificial intelligence (AI) were used. The research field within AI is connected with support vector machine (SVM) [8, 9]. Some researches use ANN as this paper [4]. SVM and ANN are similar tools but SVM is usually used for linear separable problems.

Our approach is different in the design of training sets. This paper does not use pixel differences or joint features of discrete wavelet transform and polynomial fitting errors or reversible data [8, 9, 16]. Our approach uses Huffman coding of bit word lengths extracted from discrete cosine transformation coefficients.

The paper consists of the parts with Huffman coding and ANN description. Results of four steganographic algorithms and conclusion follow.

2. Huffman Coding

Huffman coding was designed by David Huffman in 1952. This method takes symbols represented e.g. by values of discrete cosine transformation (which is one of methods how to present information in pictures like colour, brightness etc.) and codes it into changeable length code so that according to statistics the shortest bit representation is assigned to the symbols with the most often appearance [2]. It has two very important properties – it is a code with minimal length and prefix code that means that it can be decoded uniquely. On the other hand, the disadvantage is that we must know the appearance of each symbol a priori.

The images use Huffman coding for a lossless compression of data. It converts input data (coefficients of discrete cosine transformation) into bit streams of the different lengths. In the standard JPEG file the length is restricted to 16 bits. All bits after the cosine transformation can be divided into DC and AC coefficients. DC is the only one in the left upper corner in the 8x8 matrix (the standard size). The standard form is 4 tables for Huffman coding for JPEG file – AC and DC for brightness and AC and DC for colour components. Figures 3 and 4, show the differences between cover and stego images in DC or AC (direct or alternating part) class. The pictures show number of each bit word in the image (for all 4 classes together). On both images x-axis is for the length of bit word and the y-axis is for the absolute number of bit words of the actual length. The same picture is depicted in Huffman coding on both graphs. The difference is in some bit words; some of the absolute numbers are higher and some lower in cover and stego images.

3. Artificial Neural Networks

Artificial neural networks (ANN) are inspired in the biological neural nets and are used for complex and difficult tasks [4]. The most often usage is classification of objects. ANNs are capable of generalization and hence the classification is natural for them. Some other possibili-

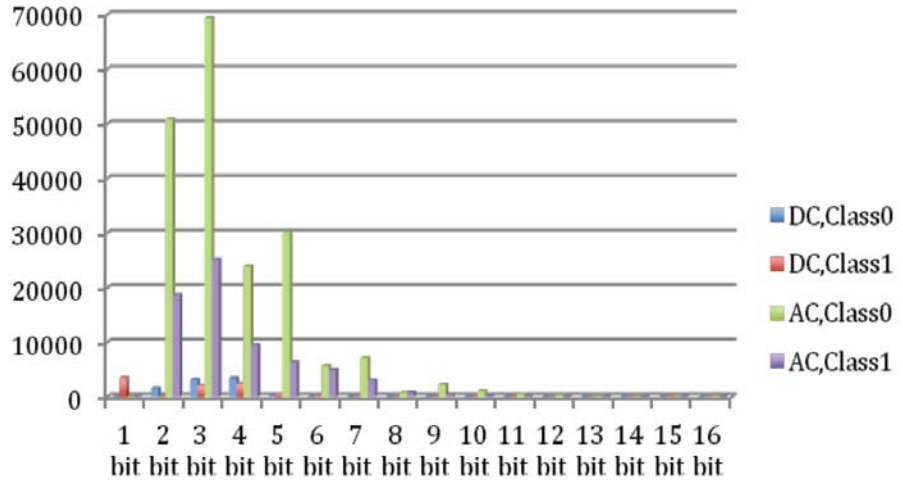


Figure 3. Huffman coding histogram – cover image (clear pictures).

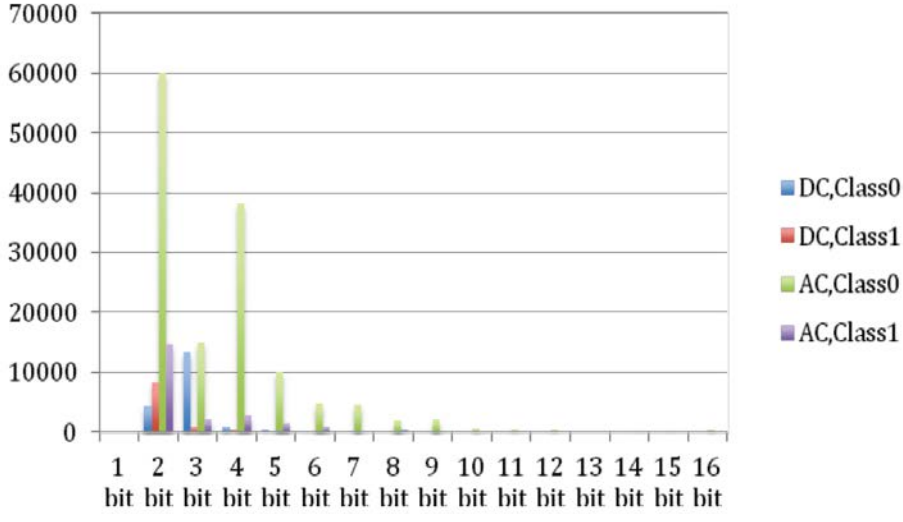


Figure 4. Huffman coding histogram – stego image (coded picture).

ties are in pattern recognition, control, filtering of signals and also data approximation and others.

Experiments were performed with feed forward net with supervision. ANN needs a training set of known solutions to be learned on them. Supervised ANN has to have input and also required output. ANNs with

unsupervised learning exist too and there a capability of self-organization is applied.

The neural network works so that suitable inputs in numbers have to be given on the input vector. These inputs are multiplied by weights which are adjusted during the training. In the neuron the sum of inputs multiplied by weights are transferred through mathematical function like sigmoid, linear, hyperbolic tangent etc. Therefore ANN can be used for data approximation [4].

These single neuron units (Fig. 5) are connected to different structures to obtain ANN (e.g. Fig. 6). These networks were design for different tasks. The notation in images is following

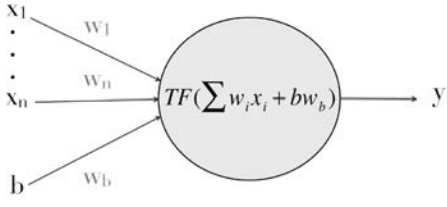


Figure 5. Model of artificial neuron, where TF means transfer function like sigmoid, x_1-x_n are inputs into neural network, b is bias (usually equal to 1), w_1 – w_n, w_b are weights, y is an output.

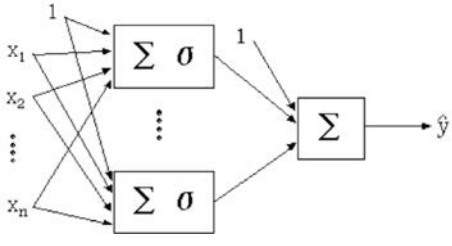


Figure 6. ANN models with one hidden layer with transfer function sigmoid inside neurons.

4. Steganographic Tools Used for Detection

For the purpose of the proposed technique four steganographic tools were selected – OutGuess, Steghide, F5 algorithm (Cipher AWT software) and PQ algorithm. All algorithms have its implementation available at Internet [5, 17, 20].

5. Training Sets

Training sets are necessary for the correct running of artificial neural nets. Within all experiments supervised ANN were used. In this case each item of a training set has an output value which says if the image is with (stego) or without a hidden content (cover).

The used training set consists of numbers obtained from Huffman coding [2]. Huffman coding was applied on adjustments and modifications of basic 2183 images, which have been acquired from three digital cameras (Sony DSC-P93, Olympus SP550UZ, Pentax K10D) in fine or superfine quality for testing purposes. The lowest image resolution for this test group was more then 2560×1600, the average picture resolution is 3529×2458 pixels and maximum picture resolution is 3872×2592 pixels with average file size 2616.6 kB and maximum file size 4403.2 kB.

5.1 Cover Images

Cover samples have been created by resizing original digital images by Linux tool ImageMagick [7] into different file resolution estimated by common appearance on internet. Full image pool has almost 22000 images for training.

List of all picture resolution used for test group: 800×600, 1024×768, 1280×1024, 1440×900, 1680×1050, 1920×1440, 2560×1600 and one special group containing original files with resolution higher then 2560×1600 pixels.

5.2 Stego Images

All samples from cover image pool were used for Outguess, Steghide and PQ algorithm. Due to the problems with F5 java implementation, input cover file pool was reduced only to images up to maximum resolution of 1680×1050 pixels in this case. The number of stego files for training was over 120000.

Secret message and encryption password were generated by Linux random number generator, which collects environmental noise from device drivers and other sources into entropy pool. The amount of hidden information was set up by measurement of common length of short messages.

List of all message lengths used for stego test samples: 5, 10, 15, 30, 75, 150, 300 and 600 Bytes.

6. Results

All experiments were performed with supervised feed forward net, which uses a Levenberg-Marquardt training algorithm. Currently, sim-

ulations are performed also with a usage of different kinds of neural networks, e.g. RBF nets with implementation of GAHC algorithm (modification of HC12) [11]. Other simulations will use also evolutionary designed ANN [18, 19] or other soft computing algorithms for suitable parameter or a structure optimization called evolutionary algorithms, e.g. Self-Organizing Migrating Algorithm [21], Differential Evolution [15] or HC12 [10].

Table 1. Results of testing success for four steganographic tools.

Type of algorithm	OutGuss	Steghide	F5 algorithm	PQ algorithm
Nr. of hidden neurons	12	1	15	1
Type of function in inner layer	logistic sigmoid	logistic sigmoid	saturated linear	logistic sigmoid
Type of function in output layer	saturated linear	saturated linear	hyperbolic tangent	saturated linear
Cover total	5246	5246	9746	22711
Cover errors	2	32	8	1
Cover % error	0.0381	0.61	0.0821	0.0044
Cover % success	99.9619	99.39	99.9179	99.9956
Stego total	135 891	142 772	77 314	126 599
Stego errors	0	3248	24	733
Stego % error	0	2.275	0.031	0.6106
Stego % success	100	97.725	99.969	99.3894
Total % error	**0.0014**	**2.216**	**0.0368**	**0.5184**
Total % success	**99.9986**	**97.784**	**99.9632**	**99.4816**

The testing of the proposed approach was performed with different settings of neurons in one hidden layer net (number from 1 to 20) and 9 combinations of transfer functions (logistic sigmoid, saturated linear and hyperbolic tangent for hidden layer and output neuron). The tests were carried out for each stego algorithm individually. Experiments with only one general neural network for all stego algorithms and even their detection have not been successful yet.

The whole data set were divided into training and testing sets. For training 14400 cover items and 135000 stego images were used. The exact number of testing items is written in a Table 1 for all testing stego algorithms.

All simulations used sixty four input neurons (obtained from Huffman coding) and one output neuron, which classifies the training item into class of cover or stego images.

Following table (Table 1 shows the best results from performed experiments. These tables contain information about number of cover and stego images and misclassified items (should be stego and ANN output was cover and viceversa). The last two rows represent a total error in a whole set of cover and stego images.

From the presented results, it follows that except for Steghide algorithm, for all other algorithms the total error in experiments was under 1%. Compared to the previous research [6, 13, 14], where the total error was almost zero, this training and testing sets consist of more items with more message payload etc. This probably caused the worsening of the results. As ANN output is a mathematical function – mathematical dependency of inputs, it is possible to extract it into the equation. This kind of equation can be used in stego detector software without knowledge of ANN structures. Such mathematical equation needs only suitable inputs, which are extracted from Huffman coding, and the output is obtained. The notations of these equations are very complex therefore they are not presented here.

7. Conclusion

In this paper steganalysis by means of artificial neural networks is presented. The novelty is in the training set design. The training set consists of 64 inputs obtained from Huffman coding extracted from discrete cosine transformation coefficients and counting of bit words of the same lengths. The paper tested files with embedded message by means of 4 steganographic algorithms – OutGuess, Steghide, PQ and F5 algorithm. ANNs were able to detect the cover and stego groups with less than 1% error. The exception was the case of Steghide where the error was around 2%. According to presented results, the proposed technique was successful. Future steps will be in preparation of a universal detector for more steganographic tools.

Acknowledgement

This work was supported by European Regional Development Fund under the project CEBIA-Tech No. CZ.1.05/2.1.00/03.0089.

References

[1] E. Cole and D. R. Krutz. *Hiding Sight*. Wiley Publishing, Inc., 2003.

[2] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*, 2nd edition, Sect. 16.3, pp. 385–392, MIT Press and McGraw-Hill, 2001. ISBN 0-262-03293-7. .

[3] S. Goldwasser and M. Bellare. *Lecture Notes on Cryptography*. Cambridge, Massachusetts, 2001. Online `http://cseweb.ucsd.edu/\~{}mihir/papers/gb.pdf`.

[4] J. Hertz, A. Kogh, and R. G. Palmer. *Introduction to the Theory of Neural Computation*. Addison-Wesley, 1991.

[5] S. Hetzl. Steghide - Linux man page. Available: `http://steghide.sourceforge.net/documentation/manpage.php`. (Accessed: 21-May-2008).

[6] J. Holoska, Z. Oplatkova, R. Senkerik, and I. Zelinka. Comparison Between Neural Network Steganalysis and Linear Classification Method Stegdetect. CIMSim 2010, Bali, Indonesie, IEEE, ISBN: 978-0-7695-4262-1.

[7] ImageMagick, `http://www.imagemagick.org/script/index.php`.

[8] Q. Liu, A. H. Sung, M. Qiao, Z. Chen, and B. Ribeiro. An improved approach to steganalysis of JPEG images. *Inform. Sciences*, 180(9):1643–1655, 2010.

[9] D.-C. Lou, C.-H. Hu, C.-L. Chou, and C.-C. Chiu. Steganalysis of HMPD reversible data hiding scheme. *Opt. Commun.*, 284(23):5406–5414, 2011.

[10] R. Matoušek. HC12: The principle of CUDA implementation. In *Proc. 16th International Conference on Soft Computing*, pages 303–308, 2010.

[11] R. Matoušek. Using AI methods to find a non-linear regression model with a coupling condition. *Engineering Mechanics*, 17(5/6):419–431, 2011.

[12] A. Nissar and A. H. Mir. Classification of steganalysis techniques: A study. *Digit. Signal Process.*, 20(6):1758–1770, 2010.

[13] Z. Oplatková, J. Hološka, I. Zelinka, and R. Šenkeřík. Steganography detection by means of neural networks. In *Proc. 9th International Workshop on Database and Expert Systems Applications*, pages 571–576, 2008.

[14] Z. Oplatková, J. Hološka, I. Zelinka, and R. Šenkeřík. Detection of Steganography Inserted by OutGuess and Steghide by means of Neural Networks. In *Proc. Asia Modelling Symposium*, 2009.

[15] K. Price. An Introduction to Differential Evolution. In D. Corne, M. Dorigo, and F. Glover, Editors. *New Ideas in Optimization*, pages 79–108, McGraw-Hill, 1999.

[16] V. Sabeti, S. Samavi, M. Mahdavi, and S. Shirani. Steganalysis and payload estimation of embedding in pixel differences using neural networks. *Pattern Recogn.*, 43(1):405–415, 2010.

[17] Software OutGuess, `www.outguess.org`.

[18] E. Volná. Design of neural network trees. In *Proc. 14th International Conference on Soft Computing*, pages 161–165, 2008.

[19] E. Volná. Learning and evolution in artificial neural networks: Comparison study. In *Proc. 4th International Workshop on Artificial Neural Networks and Intelligent Information Processing*, pages 10–17, 2008.

[20] A. Westfeld. High Capacity Despite Better Steganalysis (F5–A Steganographic Algorithm). *Lect. Notes in Comput. Sc.*, 2137:289–302, 2001.

[21] I. Zelinka. SOMA – Self Organizing Migrating Algorithm. In B. V. Babu and G. Onwubolu, editors. *New Optimization Techniques in Engineering*, Chapters 7 & 33, Springer-Verlag, 2004.

ARTIFICIAL NEURAL NETWORKS DESIGN WITH SELF-CONFIGURING GENETIC PROGRAMMING ALGORITHM

Eugene Semenkin, Maria Semenkina
Siberian State Aerospace University, Krasnoyarsk, Russia
eugenesemenkin@yandex.ru; semenkina88@mail.ru

Abstract Self-configuring genetic programming algorithm that does not require extra efforts for the choice of appropriate settings is suggested and results of testing are given. Its application to the automated ANN design is described. Approach usefulness is demonstrated through comparative performance analysis on classification problems.

Keywords: Automated design, Classification problems, Genetic programming, Neural networks, Self-configuration.

1. Introduction

Artificial neural networks (ANNs) are data processing techniques that have solved great range of complex and important problems in many different areas such as approximation, clustering, classification, control, forecasting, etc. [4]. They were successfully used in many real-world applications [9]. At the same time, the use of ANNs by end users who are not ANN experts is a difficult process because of complexity of the ANN design process. This process consists of three parts: ANN structure determination (number of neurons and their layers, connections, activation functions, etc.), connection weights tuning (ANN training) and validation. Application of ANN is problem-specific; the ANN design and training process is used to be manually fulfilled by human expert that has to test several architectures until funding the most appropriate one for problem in hand. This results in slow performance of ANN-based modeling although the current use of specific ANNs design techniques allows achieving a more automatic procedure. The most interesting here are evolutionary approaches to the ANNs design automation. There are many interesting and promising ideas in this field. We will not discuss them here, interested reader can find it in [14]. However, we have to

stress that an end user again has troubles as evolutionary algorithm application is not easier for non-experts than an ANN design and tuning.

In our study we consider the specific evolutionary tool for the ANN automated design that does not need efforts for adjustment. It is the self-configuring genetic programming (GP) algorithm. We have called it so following to organizers of the workshop "Self-tuning, self-configuring and self-generating evolutionary algorithms" (Self* EAs) within PPSN XI [10]. According to this definition, configuring is the process of selecting and using existing algorithmic components (such as variation operators) and it has to be distinguished from other adaptive behavior such as tuning, generating, etc. The configuring is exactly what we do trying to make the self-adaptive GP algorithm.

The rest of the paper is organized in following way. Section 2 explains our way to model ANN with GP, Section 3 describes the self-configuring GP (SelfCGP) algorithm idea, Section 4 describes results of numerical experiments comparing the performance of the proposed approach with alternative techniques on classification problems, in Conclusion section we discuss the obtained results.

2. ANN Design with GP

We have to describe our way to model and optimize an ANN structure with GP before an employment of our SelfCGP algorithm.

Usually, GP algorithm works with tree representation, defined by functional and terminal sets, and exploit the specific solution transformation operators (selection, crossover, mutation, etc.) until termination condition will be met [8].

The terminal set of our GP includes input neurons and 15 activation functions such as bipolar sigmoid, unipolar sigmoid, Gaussian, threshold function, linear function, etc. The functional set includes specific operations for neuron placement and connections. The first operation is the placing a neuron or a group of neurons in one layer. There will be no additional connections appeared in this case. The second operation is the placing a neuron or a group of neurons in sequential layers in such a way that the neuron (group of neurons) from the left branch of tree preceded by the neuron (group of neurons) from the right branch of tree. In this case, new connections will be added that connect the neurons from the left trees branch with the neurons from the right trees branch. Input neurons cannot receive any signal but have to send a signal to at least one hidden neuron. It might be so that our GP algorithm does not include some input neurons in resulting tree, i.e., high performance ANN structure can be found that does not need all inputs of problem.

This feature of our approach admits using our GP for the selection of the most informative problem inputs combination.

The GP algorithm forms the tree from which the ANN structure is derived. The ANN training is executed to evaluate its fitness that depends on its performance in solving problem in hand, e.g., approximation precision or number of misclassified instances. For training this ANN, connection weights are optimized with special self-configuring genetic algorithm (SelfCGA) that does not need any end user efforts to be the problem adjusted doing it automatically. We have no place here to go into the details of SelfCGA, referring to our other paper [11] but have to say that it is based on the same ideas as the self-configuring genetic programming algorithm described in the next section. When GP finishes giving the best found ANN structure as the result, this ANN is additionally trained with again SelfCGA hybridized with local search.

3. Operator Rates Based Self-Configuration of GP Algorithms

We apply operators probabilistic rates dynamic adaptation on the level of population with centralized control techniques [3, 6]. To avoid the issues of precision caused while using real parameters, we used setting variants, namely types of selection, crossover, population control and level of mutation (medium, low, high). Each of these has its own probability distribution, e.g., there are 5 settings of selection: fitness proportional, rank-based, and tournament-based with three tournament sizes. During initialization all probabilities are equal to 0.2 and they will be changed according to a special rule through the algorithm execution in such a way that the sum of probabilities should be equal to 1 and no probability could be less than predetermined minimum balance. The "idle crossover" is included in the list of crossover operators to make crossover probabilities less than 1 as it is used in conventional algorithms to model a "childless couple".

When the algorithm creates the next off-spring from the current population it first has to configure settings, i.e. to form the list of operators with using the probability operator distributions. The algorithm then selects parents with the chosen selection operator, produces an off-spring with the chosen crossover operator, mutates off-spring with the chosen mutation probability and puts off-spring into the intermediate population. When the intermediate population is filled, the fitness evaluation is computed and the operator rates (probabilities to be chosen) are updated according to operator productivities. Then the next parent population is formed with the chosen survival selection operator. The algorithm

stops after a given number of generations or if the termination criterion (e.g., given error minimum) is met.

The productivity of an operator is the ratio of the average off-spring fitness obtained with this operator and the average fitness of the overall off-spring population. Successful operators having productivity more than 1 increase their rates obtaining portions from unsuccessful operators. There is no necessity to consume extra computer memory to remember past events and updates are more dynamic (that can be both a plus and a minus).

As a commonly accepted benchmark for GP algorithms is still an "open issue" [7], we used the symbolic regression problem with 17 test functions borrowed from [1] for preliminary evaluation. Experiments settings were 100 individuals, 300 generations and 100 algorithm runs for each test function. Having no place here for detailed description of experiments and results, we summarize briefly and refer to our other paper [12]. We evaluated the algorithm reliability, i.e. proportion of 100 runs when approximation with sufficient precision was found.

The worse reliability (for the most hard problem) averaged over 100 runs was equal to 0.42. The best reliability was equal to 1.00. SelfCGP reliability averaged over 17 test function is better than averaged best reliability of conventional GP. Computation consumption is also better. It gives us a possibility to recommend SelfCGP for solving symbolic regression problems as better alternative to conventional GP. Main advantage of SelfCGP is no need of algorithmic details adjustment with no losses in performance that makes this algorithm useful for many applications where terminal users being no experts in evolutionary modeling nevertheless intend to apply GP for solving these problems. In the next section we apply SelfCGP to the automated design of ANN for classification problems.

4. Performance Comparison of ANN-Based Classifiers Designed with Self-Configuring GP and Alternative Classification Methods

In this section, we compare the performance of the ANNs designed with our SelfCGP algorithms with alternative methods on two sets of problems from [2]. Materials for the first comparison we have taken from [15] where together with results of authors' algorithm (CROANN) the results of 15 other approaches are presented on the same three classification problems from [2].

The problems are:

Iris Dataset: The Iris dataset is the most widely-used benchmark for machine learning and pattern recognition. The whole dataset can be divided into three different classes of iris species: *Setosa*, *Versicolour* and *Verginica*. The species of iris can be determined by four attributes of the plants: *sepal length*, *sepal width*, *petal length* and *petal width*. The dataset is divided in [15] into three parts: 75 training samples, 37 validation samples and 38 testing samples.

Wisconsin Breast Cancer Dataset: The Wisconsin Breast Cancer dataset contains 699 samples, each of which has real-valued attributes and can be classified into two classes: 458 *benign* and 241 *malignant*. To test the performance evaluation, all samples are divided into three parts by simple random sampling method: 349 training samples, 175 validation samples and 175 testing samples.

Pima Indian Diabetes Dataset: The Pima Indian Diabetes dataset contains 768 samples, 500 of which are indicated with sign of diabetes and 268 are without such sign. There are eight real-valued attributes that can be used to determine whether a patient has the sign of diabetes or not. This dataset is known as a difficult problem for machine learning for its scarcity of samples and heavy noise pollution. This dataset is partitioned into 384 training samples, 192 validation samples and the 192 testing samples.

The numbers in table 1 bellow were taken from [15] except the two upper rows that contain evaluations for our approaches (GP+ANN from Section 2 and SelfCGP+ANN from Section 3). Evaluations were fulfilled in the same manner as in [15], i.e. indicator is equal to weighted sum of mean squared error and quality of classification. Experiments conducted in the same way as it is described in [15] with one exception. The function evaluation limit for our algorithms was twice as many as alternative algorithms had. Alternative algorithms worked with one hidden layer perceptron having one activation function and, certainly, needed no computational expenditures for determining the ANN structure and activation function for each neuron.

Analyzing Table 1, we can conclude that performance of the approach suggested in this paper is high enough comparing alternative algorithms (1^{st}, 3^{rd} and 4^{th} places). However, the main benefit from our SelfCGP algorithm is the possibility to be used by end user without expert knowledge ANN modeling and evolutionary algorithm application. Additional dividend is the size of designed ANNs (see Table 2).

The ANNs designed with SelfCGP contain few hidden neurons and connections and use not all given inputs although perform well. It can be used for discrimination of unimportant inputs that could also give additional information for experts.

Table 1. Classifiers performance comparison - 1.

Classifier	Iris	Cancer	Diabetes
SelfCGP+ANN	**1.30**	1.05	19.70
GP+ANN	1.52	1.24	21.56
CROANN	1.31	1.06	**19.67**
GANet-best	6.40	1.06	24.70
SVM-best	**1.40**	3.10	22.70
CCSS	4.40	2.72	24.02
COOP	-	1.23	**19.69**
CNNE	-	1.20	**19.60**
EPNet	-	1.38	22.38
EDTs	-	2.63	-
SGAANN	14.20	1.50	24.46
EPANN	12.56	1.54	25.75
ESANN	7.08	**0.95**	20.93
PSOANN	10.38	1.24	20.99
GSOANN	3.52	**0.65**	19.79
MGNN	4.68	3.05	-
EENCL	-	-	22.1

Table 2. Sizes of ANNs designed with SelfCGP.

	Number of	Iris	Cancer	Diabetes
Average	*Inputs*	2.3	13.4	4.6
	Hidden neurons	9	34.7	23.2
	Connections	21.3	111.4	59.8
Minimal ANN	*Inputs*	2	7	3
	Hidden neurons	6	21	16
	Connections	15	34	34
Maximal ANN	*Inputs*	3	19	9
	Hidden neurons	13	58	31
	Connections	40	194	73

In [12] we compared classifiers generated with our SelfCGP in the form of symbolic expressions giving separating surfaces through symbolic regression method, with alternative approaches on two harder classification

problems containing not only real numbers as inputs but also qualitative data. These classifiers demonstrated competitive results but were not so close to winning position. That is why we considered as a good idea to continue that comparison adding in the competitors list the new ANN-based classifiers designed by SelfCGP.

The first data set, called the German Credit Data Set, includes customer credit scoring data with 20 features, such as age, gender, marital status, credit history records, job, account, loan purpose, other personal information, etc. There are 700 records judged to be creditworthy and 300 records judged to be non-creditworthy. The second data set includes Australian credit scoring data with 307 examples of the creditworthy customers and 383 examples for the non-creditworthy customers. It contains 14 attributes, where six are continuous attributes and eight are categorical attributes. Both data sets are made public from the UCI Repository of Machine Learning Databases [2], and are often used to compare the accuracy with various classification models.

Results for alternative approaches have been taken from scientific literature. In [5] the performance evaluation results for these two data sets are given for authors' two-stage genetic programming algorithm (2SGP) as well as for the following approaches taken from other papers: conventional genetic programming (GP), classification and regression tree (CART), C4.5 decision trees, k nearest neighbors (k-NN), linear regression (LR). Additional material for comparison we have taken from [13] where are evaluations data for authors' automatically designed fuzzy rule based classifier as well as for other approaches found in literature: Bayesian approach, boosting, bagging, random subspace method (RSM), cooperative coevolution ensemble learning (CCEL).

The first two rows of Table 3 contain results for algorithms proposed in this paper. Next row is related to our previous study (self-configuring GP and modified GP for symbolic regression).

Numbers highlighted in bold in table 3 are the best results in our comparison. The winner is 2SGP from [5], the algorithm specially designed for credit scoring problems solving. ANN-based classifier designed with SelfCGP is the second best on the Australian credit problem and the third best on the German credit problem.

We understand conventionality of this comparison, e.g. there is no information on indicators' variation or on computational efforts of compared methods. Some results are the best for the corresponding method, others, including our results, are averaged. Moreover, it is clear that 2SGP and fuzzy classifier are much more useful for decision makers as they give not only computational expression but also human expert understandable production rules. Our intention was to make it clear

Table 3. Classifiers performance comparison - 2.

Classifier	Australian	German
SelfCGP+ANN	0.9022	0.7940
GP+ANN	0.8969	0.7863
SelfCGP+SR	0.8930	0.7850
2SGP	**0.9027**	**0.8015**
GP+SR	0.8889	0.7834
Fuzzy classifier	0,8910	0,7940
C4.5	0.8986	0.7773
LR	0.8696	0.7837
Bayesian appr.	0,8470	0,6790
Boosting	0,8470	0,6840
Bagging	0,8520	0,6770
RSM	0,8660	0,7460
CCEL	0.7150	0.7151
k-NN	0.8744	0.7565
CART	0.8986	0.7618

whether our approach could give results competitive to alternative techniques without suggesting the best tool for banking credit scoring. That is why we did not adapt our algorithms to these problems making no problem-specific modifications and given computational resource in 500 generations (and not 1000 as in [5]), etc.

Now we can conclude that self-configuring genetic programming algorithm is the suitable tool for ANN automated design. It can produce competitive performance results at least in ANN-based approximation and classification problems, makes end user free from the ANN structure design and GP settings determination.

5. Conclusions

In this paper, the modified approach to probability based operators rates assignment for the GP algorithm automated configuration is briefly described. These modifications turned to be useful that was demonstrated with test problems. Main result is the new approach to automated design of artificial neural networks based on the specific form of the genetic programming algorithm. This method allows to design arbitrary feed-forward ANN with range of activation functions. Application of the self-configuring GP algorithm technique allows ANN designing by

non-expert end users as they do not need deep knowledge of ANN-based modeling and GP settings determination. The usefulness of suggested tools was proved with the range of real world problems.

We developed self-configuring GP algorithm that can be used for solving different problems by means of different tools. It seems not so hard work to adopt our SelfCGP for automated design and tuning of the fuzzy logic based systems, decision trees and other computational intelligence tools including problem-specific ones.

References

[1] S. Finck, N. Hansen, R. Ros, and A. Auger. Real-parameter black-box optimization benchmarking 2009: Presentation of the noiseless functions. Technical Report, Researh Center PPE, 2009.

[2] A. Frank and A. Asuncion. UCI Machine Learning Repository. School of Information and Computer Science, University of California, Irvine, 2010.

[3] J. Gomez. Self adaptation of operator rates in evolutionary elgorithms. *Lect. Notes Comput. Sc.*, 3102:1162–1173, 2004.

[4] S. Haykin. *Neural Networks*, 2nd edition. Prentice Hall, 1999.

[5] J.-J. Huang, G.-H. Tzeng, and C.-S. Ong. Two-stage genetic programming (2SGP) for the credit scoring model. *Appl. Math. Comput.*, 174(2):1039–1053, 2006.

[6] S. Meyer-Nieberg, H.-G. Beyer. Self-Adaptation in Evolutionary Algorithms. In F. Lobo, C. Lima, and Z. Michalewicz, editors. *Parameter Setting in Evolutionary Algorithm*, pages 47–75, 2007.

[7] M. ONeill, L. Vanneschi, S. Gustafson, and W. Banzhaf. Open issues in genetic programming. *Genetic Programming and Evolvable Machines*, 11(3-4):339–363, 2010.

[8] R. Poli, W. B.Langdon, and N. F. McPhee. A Field Guide to Genetic Programming. Published via `http://lulu.comandfreelyavailableathttp://www.gp-field-guide.org.uk`, 2008. (With contributions by J. R. Koza)

[9] J. R. Rabunal and J. Dorado, editors. *Artificial Neural Networks in Real-Life Applications*. Idea Group Inc., 2005.

[10] R. Schaefer, C. Cotta ,J. Kolodziej, and G. Rudolph, editors. *Proc. 11th International Conference on Parallel Problem Solving from Nature*, 2010.

[11] E. Semenkin and M. Semenkina. Self-configuring genetic algorithm with modified uniform crossover. In *Proc. 3rd International Conference on Swarm Intelligence*, 2012.

[12] E. Semenkin and M. Semenkina. Self-configuring genetic programming algorithm with modified uniform crossover. In *Proc. IEEE Congress on Evolutionary Computation*, 2012.

[13] R. Sergienko, E. Semenkin, and V. Bukhtoyarov. Michigan and Pittsburgh methods combining for fuzzy classifier generating with coevolutionary algorithm for strategy adaptation. In *Proc. IEEE Congress on Evolutionary Computation*, 2011.

[14] X. Yao. Evolving artificial neural network. *P. IEEE*, 87(9):1423–1447, 1999.

[15] J. J. Q. Yu, A. Y. S. Lam, and V. O. K. Li. Evolutionary artificial neural network based on chemical reaction optimization. In *Proc. IEEE Congress on Evolutionary Computation*, 2011.

ARABIC TEXT CATEGORIZATION VIA BINARY PARTICLE SWARM OPTIMIZATION AND SUPPORT VECTOR MACHINES

Hamouda K. Chantar, David W. Corne
Department of Computer Science, Heriot-Watt University, Edinburgh, UK
hamoudak77@yahoo.com; d.w.corne@hw.ac.uk

Abstract Document categorization concerns automatically assigning a category label to a text document, and has increasingly many applications, particularly in the domains of organizing, browsing and search in large document collections. It is typically achieved via machine learning, where a model is built on the basis of a (typically) large collection of document features. Feature selection is critical in this process, since there are typically several thousand potential features (distinct words or terms). Here we explore binary particle swarm optimization (BPSO) hybridized with either K-Nearest-Neighbour (KNN) or a Support Vector Machine (SVM), for feature selection in Arabic document categorization tasks. Comparison between feature selection methods is done on the basis of using the selected features, in conjunction with each of SVM, C4.5 and Naive Bayes, to classify a holdout test set. Using publicly available datasets, we show that the BPSO_{SVM} approach seems promising in this domain. We also analyse the sets of selected features and consider the differences between the types of feature that BPSO_{KNN} and BPSO_{SVM} tend to choose; this leads to speculations concerning the appropriate feature selection strategy, based on the relationship between the classes in the document categorization task at hand.

Keywords: Arabic text processing, Document categorization, Feature selection, Text mining.

1. Introduction

With rapid growth in the availability of text documents in electronic form, automatic text analysis is becoming increasingly important. One task of interest in this area is text categorization (TC), which is the task of labelling texts with one or more pre-defined categories (such as 'cul-

ture' or 'sport') based on their content. There are many applications for TC, largely in the context of searching and/or browsing large collections of documents [12, 16].

Meanwhile, Feature selection (FS) is essential in data mining in general and in TC in particular [12].FS aims to improve both overall accuracy and computational efficiency, by aiming to find the most useful features with regard to information that helps discriminate between categories. Typically there are many thousands of words that could be used as features in TC, so the need for robust FS methods is acute.

Broadly, there are two main approaches to FS: wrapper and filter. In a wrapper approach, we search the space of feature subsets, estimating the quality of a subset via a learning algorithm. In a filter approach, a subset of features is selected *a priori* using feature scoring metrics (e.g. information gain or mutual information [12, 16]. In general, filter approaches are fast, but wrapper approaches are more successful at dealing with feature interactions. We are interested in classification accuracy, and therefore choose the wrapper approach.

In particular, our aim is to improve the accuracy of TC for Arabic language texts, for applications such as automated labelling of news articles, and automated labelling or filtering of search results. In contrast with English, Arabic has a complex morphology that makes developing automatic processing systems for it a particularly challenging task [11, 21]. In previous work we have explored binary particle swarm optimization (BPSO) in conjunction with K-nearest neighbour (KNN) on benchmark datasets, and found it highly competitive with other approaches [4]. Here we explore BPSO in conjunction with a support vector machine (SVM). As well as analysing the relative performance of BPSO_{SVM} and BPSO_{KNN} in a variety of contexts on large datasets, we also examine the features generated by each method, and speculate on the appropriate FS strategy to use based on aspects of the task at hand.

The remainder is set out as follows. In Section 2 we briefly cover related work, focusing on TC for Arabic datasets. Section 3 then describes all of the algorithms and datasets that are involved in our experiments. The experiments and their results are presented and discussed in Section 4, and we conclude in Section 5.

2. Background and Related Work

Limited work in TC has been done so far for Arabic texts. Several studies have been reported, however this area is at an early stage. For example, although publically available datasets exist, it is rare for any such dataset to be used in more than one paper. In the following, we

therefore briefly review what has been done, and we also quote reported accuracy figures, however it should be remembered that these signal a very broad idea of performance, and almost never represent performance on a common dataset under common conditions.

The most prominent previous work includes the following. In [1], the authors evaluated C5.0 and SVMs to classify Arabic texts, reporting accuracies of 68.65% and 78.42% respectively. In [6], Naive Bayes was used to classify Arabic web pages, achieving mean accuracy of 67.78%. In [5], maximum entropy was used to classify Arabic texts, using only nouns and pronouns as features, with performance reaching 80.41%. SVMs were used in [15], in conjunction with a Chi square feature selection method to classify Arabic texts; the accuracy, averaged over all categories using the F1-measure, was 88.11%. In [2], the authors investigated CBA, Naive Bayes and SVM on classifying Arabic text documents. The results show that CBA outperformed NB and SVM and the average of its F1-measure is 0.804. Among the best accuracies reported in this field so far come from [24], in which FS was performed with BPSO, in connection with Radial Basis Function (RBF) networks. In terms of F-measure, the performance of the proposed method reached 93.9%. Finally, in [4] we built on the work in [24] by drawing together three of the main available datasets used in Arabic TC studies, and thereby performing more extensive tests of the performance of BPSO for feature selection in this field. In that work we used BPSO in conjunction with kKNN to select feature subsets. The 10-fold cross-validation performance of SVM, when applied to holdout data restricted to feature subsets emerging from the feature selection process, ranged from 89% to 96%.

3. Algorithms

The primary new technique that we investigate here is a hybrid of BPSO and SVM. BPSO is used to search through the space of feature subsets, while each subset is evaluated by SVM. We denote this BPSO_{SVM}, and compare it with BPSO_{KNN}. Following a run of either BPSO_{SVM} or BPSO_{KNN}, the resulting feature subset is evaluated on a holdout (unseen) dataset. The latter evaluation is done using either Naive Bayes, SVM, or C4.5. In this section we give brief details of the main algorithms used in our experiments.

Particle swarm optimization (PSO) [9] is now extensively used in real-valued parameter optimisation. PSO operates with a population of solutions (particles), each of which has a position (the solution itself) and velocity. In each generation, each particle adjusts its velocity, by perturbing it in the direction of the current global best particle, and also

in the direction of its 'personal best' position from previous generations [9, 20]. In BPSO, the position vector of a particle is a binary vector [10]; the velocity is calculated and updated in the same way as in standard PSO. However, the particle's position is updated in a different way, as follows. Each component x_i of the position vector is updated based on a probability calculated from the corresponding component v_i of the velocity vector. This probability is simply:

$$S(v_i) = \frac{1}{1 + e^{-v_i}}. \tag{1}$$

In this way, $S(v_i)$ defines the probability that x_i will be updated to 1; otherwise it will be updated to 0. BPSO has been shown successful in a wide variety of applications.

KNN is widely used in TC, being quite simple and robust [8]. Again, it is well known and we just briefly describe it here. In our work, KNN is used to categorize a query document represented as a vector of feature values. It works by referring to a reference collection of example documents that have known category labels, and finds from those the K closest to the query document. The chosen category label is the one that is most common among the K closest reference documents. If there is a tie, then raw distance values are used to break the tie. In this work, as is commo when using KNN, we use the standard Euclidean distance measure.

A Support Vector Machine (SVM) is a well-known and relatively new approach in which a problem is addressed by first mapping the feature vectors into a higher-dimensional space. A separating hyperplane is then found in this space [3], and unseen feature vectors are classified according to their position with respect to that hyperplane following the same mapping. The choice of mapping is guided by a kernel function, which can take many forms. When the number of features is very large, as is so in TC, it is common to use a linear kernel function [3], and this is our approach.

Each of BPSO_{SVM} and BPSO_{KNN} runs as a basic BPSO, aiming to find a binary string (a feature subset) that optimizes the fitness function. The fitness of particle x is calculated as:

$$F(\mathbf{x}) = (\alpha * Acc) + (\beta * ((N - T)/N)), \tag{2}$$

where Acc is the classification accuracy of the particle on the training set, found using either SVM or KNN, as explained below; α and β are parameters used to balance the influences of accuracy and subset size, with $\alpha \in [0, 1]$ and $\beta = 1 - \alpha$; N is the total number of features in the dataset, and T is the number of selected features in particle x.

In BPSO_{KNN}, accuracy is estimated simply by finding the proportion of the training set for which KNN (using the feature subset specified by x) gives the correct classification. In BPSO_{SVM}, an SVM is run on the training set in ten-fold cross-validation mode, and accuracy is the mean F-measure over the 10 folds. The SVM used is that supplied with the WEKA software [7].

4. Experiments

We perform experiments on the following datasets. The *Alwatan Dataset*, available at [26], comprises 20291 Arabic news documents gathered from the online newspaper "Alwatan" by [14], and each document is labelled with one of six categories. We have selected 1173 documents from four categories randomly. The *Al-jazeera-News Dataset*, available at [25], was obtained by [17], comprising 1500 documents in five categories (Sport, Economy, Science, Politics and Art), with 300 documents in each category. Finally, the *Akhbar-Alkhaleej Dataset*, available also at [26], is a collection of 5690 Arabic news documents gathered evenly from the online newspaper "Akhbar-Alkhaleej" by [13]. It consists of five categories and each document has only one category label. In this work, we have selected 1708 documents at random. Table 1 summarises characteristics of each of the datasets, indicating the precise split into training and test documents.

Table 1. Characteristics of the three Arabic text datasets used in our experiments.

	Alwatan	*Al-Jazeera*	*Akhbar-Alkhaleej*
Categories	4	5	4
Documents in Training set	821	1200	1365
Documents in Test set	352	300	343
Features in Training Set	12282	5329	8913

Documents were preprocessed as follows [11, 23]. First, texts were converted to UTF-8 encoding, and then hyphens, punctuation marks, numbers, digits, non-Arabic letters and diacritics were all removed, along with stop words and rare words (words that occur under five times in a dataset). We did not perform word stemming, and no Arabic letters were normalized (e.g. as in [11]). Following this, each dataset was further processed to produce a feature vector for each document, using TFIDF [18, 19] to return values for each feature. Hence, each document in each dataset was finally converted into a feature vector, where, for a given

dataset, the value for feature (word) i in document j is:

$$TF(t_i, d_j) \times log\frac{D}{DF(t_i)} \tag{3}$$

in which $TF(t_i, d_j)$ is the number of times term i appears in document j, and the right hand side represents the inverse document frequency (IDF), in which D is the number of documents in the dataset, and $DF(t_i)$ is the number of those documents in which term i appears.

The parameter settings for BPSO were as follows for all experiments. Inertia weight was 1.02 (determined after trying different values in [0.4, 1.2] in preliminary experiments), swarm size was set to 30, $C1$ and $C2$ were set to 2.0, and the termination criterion was 100 generations (equivalently, 3000 fitness evaluations) – this was found best in preliminary experiments which tested 50, 100 and 150). Finally, the fitness calculations used $\alpha = 0.85$.

The K in KNN was maintained at 3 (following preliminary tests with 1, 3 and 5). The SVM was the implementation used in [22], using the learning algorithm C-SVC, a linear kernel function, and setting the cost parameter $C = 1$ (a relatively low value favouring good generalization).

All experiments used the Weka software [7, 22, 23]. We used three different classification algorithms to evaluate the selected subsets of features on the holdout/test portion of each dataset. These were SVM (the same parameters and kernel function as used for the SVM within BPSO_{SVM}), Naive Bayes, and C4.5 (in WEKA, the classifier named J48 represents C4.5.)

5. Results and Analysis

BPSO_{SVM} and BPSO_{KNN} were each trialled ten times independently on each of the three datasets. After each individual trial, the best resulting feature subset was applied to reduce the holdout portion of the dataset, and C4.5 (J48), NB and SVM were independently applied, using tenfold cross validation. Table 2 summarises results on the holdout data. Basic inspection finds that for each dataset and learning method, the mean accuracy on the holdout set is higher for BPSO_{SVM} than for BPSO_{KNN}. However, in most cases the difference is small. Standard T-tests were applied (one-tailed), with significance signalled by $p < 0.1$, and found the following. On Alwatan, the SVM results for BPSO_{SVM} are superior to those of BPSO_{KNN} ($p < 0.085$), however the corresponding results for NB and C4.5 are not significant. On Aljazeera, none of the corresponding pairwise comparisons is significant, while on Akhbar-Alkhaleej, the NB results for BPSO_{SVM} are superior to those

of BPSO_{KNN} ($p < 0.044$), while the other two comparisons show no significant difference.

Table 2. Summary of results: weighted accuracy on the holdout test set for three machine learning algorithms (using tenfold cross-validation), using features selected from either BPSO_{SVM} or BPSO_{KNN}.

	BPSO_{SVM}			BPSO_{KNN}		
	C4.5	NB	SVM	C4.5	NB	SVM
Alwatan						
mean acc.	80.31	91.04	96.39	78.97	89.49	95.97
std	0.02365	0.01409	0.00809	0.02502	0.04098	0.00424
Aljazeera						
mean acc.	74.39	80.14	90.36	73.96	80.03	89.51
std	0.02985	0.01437	0.01119	0.02211	0.02324	0.02319
Akhbar-A.						
mean acc.	78.69	84.34	89.14	78.53	83.3	88.86
std	0.01845	0.01471	0.01056	0.02206	0.01036	0.01608

Considering again the full set of results, we see each of the nine mean values for BPSO_{SVM} outperforming the corresponding values for BPSO_{KNN}; a binomial test indicates this is clearly significant with $p < 0.01$. This suggests that BPSO_{SVM} provides a very clear advantage over BPSO_{KNN}, although we need more independent trials to confirm statistical significance for more specific claims.

We performed further analysis in attempt to understand the differences in performance. The selected features were recorded for every trial in each experiment. This enabled us to examine the degree to which individual features repeatedly emerged in different trial runs on the same data. We summarize the broad findings from this analysis in Table 3.

Table 3. Showing the degree to which the same features emerged from different feature selection trials.

	$BPSO_{SVM}$			$BPSO_{KNN}$		
	8	*9*	*10*	*8*	*9*	*10*
Alwatan	686	134	24	1072	315	38
Aljazeera	289	81	13	362	103	17
Akhbar-A.	488	124	18	529	145	22

For example, on the Alwatan dataset, Table 3 tells us there were 686 features (words) that each occurred in precisely 8 of the 10 trials of BPSO_{SVM}, while the corresponding number for BPSO_{KNN} was 1072. We also see that, for BPSO_{SVM}, there were 24 specific features present in the result of *every* trial, while the corresponding figure for BPSO_{KNN} was 38. Overall, feature sets from different trial runs of BPSO_{KNN} tended to have a greater overlap with each other than those emerging from BPSO_{SVM} trials. We also report that the intersections between the frequently appearing BPSO_{SVM} and BPSO_{KNN} features were always zero or negligible. E.g. on the Alwatan dataset, there was no overlap between the BPSO_{SVM} and BPSO_{KNN} 'all 10 trials' features.

The fact that each method seems to rely on a small core of features, possibly reflects the underpinning semantics of document categories. That is, for a category such as 'religion', you may expect a core set of words which usually indicate a strong signal about the category, while a large number of additional words may provide a weaker signal, since they could also be common to other categories (e.g. 'culture' or 'news').

However, the fact that BPSO_{SVM} and BPSO_{KNN} tend to choose distinct core sets from each other is perhaps more interesting. We expect this reflects the distinct approaches used by SVM and KNN. Features most often chosen by KNN could be 'core' to a category in the sense that they are close to a centroid in document vector space. Meanwhile, the important features for SVM are those that help define clear boundaries between categories in the transformed feature space. The SVM will be more tolerant of features that are shared between categories. E.g. consider a document in 'local news' that is close to the boundary with 'international news'. KNN will misclassify this document if its closest neighbours tend to be over the boundary, however SVM will classify it correctly. I.e. SVM features may be chosen according to how well boundaries can be defined, and KNN features may be more biased towards keeping a large distance between documents in different categories.

This would suggest that BPSO_{SVM} may be preferable when the distance between categories is small, since several documents will be close to boundaries. There are hints that this is reflected in our results, since where clear statistical significance was found among individual comparisons, this was in the case of the Alwatan dataset, where the categories include 'Culture' and 'Religion', and the Akhbar dataset, where the categories include 'International News' and 'Local News'.

6. Conclusions

In this paper, BPSO_{SVM} was described as a feature selection (FS) method for Arabic text categorization (TC). It was compared with BPSO_{KNN}, and statistical evidence indicates that BPSO_{SVM} should be favoured, although more experiments are required to solidify this finding in terms of how the performance of the FS method influences the quality of TC for different classification algorithms. We also speculated on factors that might influence the relative performances of BPSO_{SVM} and BPSO_{KNN}. Analysis showed that each relies on a different core set of features. We put forward a simple argument to explain this in terms of the different classification models used by SVM and KNN, and this further suggests BPSO_{SVM} may be favoured when the categories tend to be quite close. This could point towards useful guidelines for choosing FS methods in this area. For example, if categories tend to be quite distinct, then BPSO_{KNN} (or similar) may be favourable simply because of the ability to more speedily determine a feature subset. However if accuracy in distinguishing between close categories is important, we might accept the computational cost of BPSO_{SVM}. Finally, hybrid approaches may be worth investigating, where (for example) BPSO_{KNN} may be used to find a feature subset that works well on a distinct partition of categories that aggregates similar categories together, and BPSO_{SVM} may then be used to find feature subsets specifically to distinguish between the close categories.

References

[1] S. Al-Harbi, A. Almuhareb, A. Al-Thubaity, A. Khorsheed, and A. Al-Rajeh. Automatic Arabic text classification. In *Proc.9es Journées internationales dAnalyse statistique des Données Textuelles*, pages 77–83, 2008.

[2] S. Al-Saleem. Associative classification to categorize Arabic data sets. *International Journal of ACM Jordan*, 1(3):118–127, 2010.

[3] C. J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Min. Knowl. Disc.*, 2(2):121–167, 1998.

[4] H. K. Chantar and D. W. Corne. Feature subset selection for Arabic document categorization using BPSO-KNN. In *Proc. 3rd World Congress on Nature and Biologically Inspired Computing*, pages 546–551, 2011.

[5] A. El-Halees. Arabic Text Classification Using Maximum Entropy. *The Islamic University Journal*, 15(1):157–167, 2007.

[6] M. El-Kourdi, A. Bensaid, and T. Rachidi. Automatic Arabic document categorization based on the Naive-Bayes Algorithm. In *Proc. Workshop on Computational Approaches to Arabic Script-Based Languages*, 2004.

[7] Y. El-Manzalawy and V. Honavar. WLSVM: Integrating LibSVM into Weka, 2005. Available at `http://www.cs.iastate.edu/~yasser/wlsvm`.

[8] L. Jiang, Z. Cai, D. Wang, and S. Jiang. Survey of improving K nearest neighbor for classification, In *Proc. Fourth International Conference on Fuzzy Systems and Knowledge Discovery*, pages 679–683, 2007.

[9] J. Kennedy and R. C. Eberhart. Particle swarm optimization. In *Proc. IEEE International Conference on Neural Networks*, volume 4, pages 1942–1948, 1995.

[10] J. Kennedy and R. C. Eberhart. A discrete binary version of the particle swarm algorithm. In *Proc. IEEE International Conference on Systems, Man, and Cybernetics*, pages 4104–4108, 1997.

[11] L. Khreisat. A machine learning approach for Arabic text classification using N-gram frequency statistics. *Journal of Informatics*, 3(1):72–77, 2009.

[12] S. Li, R. Xia, C. Zong, and C. Huang. A framework of feature selection methods for text categorization. In *Proc. Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 692–700, 2009.

[13] A. Murad and K. Smaili. Comparison of topic identification methods for Arabic language. In *Proc. International Conference on Recent Advances in Natural Language Processing*, 2005.

[14] A. Murad, K. Smaili, and D. Berkani. Comparing TR-classifier and kNN by using rReduced sizes of vocabularies. In *Proc. 3rd International Conference on Arabic Lang. Proc.*, 2009.

[15] A. Mesleh. Chi square feature extraction based SVMs Arabic language text categorization system. *Journal of Computer Science*, 3(6):430–435, 2007.

[16] Y. Saeys, I. Inza, and P. Larranaga. A review of feature selection techniques in bioinformatics. *Bioinformatics*, 23(19):2507–2517, 2007.

[17] D. Said, N. Wanas, N. Darwish, and N. Hegazy. A Study of Text Preprocessing Tools for Arabic Text Categorization. In *Proc. 2nd International Conference on Arabic Language Resources and Tools*, pages 230–236, 2009.

[18] G. Salton, A. Wong, and C. S. Yang. A vector space model for automatic indexing. *Comm. ACM*, 18(11):613–620, 1975.

[19] F. Sebastiani. Machine learning in automated text categorization. *ACM Comput. Sur.*, 34(1):1–47, 2002.

[20] Y. Shi and R. C. Eberhart. A modified particle swarm optimization. In *Proc. IEEE Congress on Evolutionary Computation*, pages 60–73, 1998.

[21] M. Syiam, Z. Fayed, and M. Habib. An intelligent system for Arabic text categorization. *International Journal of Intelligent Computing and Information Sciences*, 6(1):1–19, 2006.

[22] WEKA Software: `http://www.cs.waikato.ac.nz/ml/weka`, 2001.

[23] H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, San Francisco, 2005.

[24] B. Zahran and G. Kanaan. Text feature selection using particle swarm optimization algorithm. *World Applied Sciences Journal*, 7:69–74, 2009.

[25] `http://filebox.vt.edu/users/dsaid/Alj-News.tar.gz`.

[26] `http://sites.google.com/site/mouradabbas9/corpora`.

SOLVING VARIATIONAL AND CAUCHY PROBLEMS WITH GENETIC PROGRAMMING ALGORITHMS

Sergey Burakov, Eugene Semenkin
Siberian State Aerospace University, Krasnoyarsk, Russia
burakov_krasu@mail.ru; eugenesemenkin@yandex.ru

Abstract It is suggested to use genetic programming techniques for solving Cauchy problem and variational problem that allows to get exact analytical solution if it does exist and approximate analytical expression otherwise. Features of solving process with this approach are considered. Results of numerical experiments are given.

Keywords: Cauchy problem, Genetic programming algorithm, Numeric methods, Ordinary differential equations, Variational problem.

1. Introduction

The ordinary differential equations (ODEs) may be solved using analytical or numerical methods. Both have advantage and drawbacks. Analytical method has the strong mathematical backgrounds in the theorems of existence and uniqueness of solution [4]. Also the result of analytical method is a function in symbolic form that may be analyzed or applied as mathematical formula. However the analytical method can be used with only a few types of ODE, all are well known. Practical ways of solving these ODEs are described in the theory of ODEs. In contrast to the analytical method of ODEs solving, the numeric method allows to obtain solution of any ODE [3]. But the result of the solving is a table of real numbers, describing the relationship of function values and variables. For numeric method, it is important to prove stability and convergence of the schema, that could be a problem.

For solving a variational problem, one transforms it to a boundary problem using the Euler equations, which are the necessary condition of the extremum existence (if it does exist) [5]. In general, the unique solution is not guaranteed for boundary problem. But even when the

solution exists and is unique, not every solution of ODE can be expressed in a quadrature.

The aim of this work is the development of genetic programming algorithms for solving variational and Cauchy problems, which allows finding an exact solution in analytical form (if it does exists) or, alternatively, an approximating analytical expression, if the solution can't be represented by a combination of elementary functions.

2. Cauchy Problem

There is the ODE in general form with initial conditions:

$$F\left(x, y, y', y'', \ldots, y^{(n)}\right) = 0, \tag{1}$$

$$y(x_0) = Y_0, y'(x_0) = Y_0', \ldots, y^{(n-1)}(x_0) = Y_0^{(n-1)}. \tag{2}$$

It is necessary to find the function, which satisfies the Eq. (1) and conditions (2). For this problem are proved the existence and uniqueness theorems in mathematics. Hereinafter it is supposed that the theorem conditions are fulfilled, i.e., the problem solution exists and is unique.

The GP based approaches to solving differential equations have been applied in many different ways, e.g., [7, 8] where numerical methods or polynomial approximations were usually used. The most closed to our approach are results presented in [2, 11] although we used not integer numbers but real ones.

There are two ways to solve the problem with genetic programming (GP) algorithm. The first way is direct applying GP to the problem (1)–(2), the second way is a combination of GP algorithm with numerical method [1, 8].

2.1 Direct Approach with Genetic Programming Algorithm

The process of the finding a problem (1)–(2) solution in symbolic form can be considered as a complex optimization procedure, that provides the smallest value of the error function on the set of symbolic expressions. The global optimum (for which the error function value is equal to zero) is true solution of Cauchy problem for ODE. If the symbolic expressions (i.e., problem solutions) are represented by binary trees, their transformation (subtrees exchange) models the search process of solution with smaller error function value. Therefore, it is convenient to use the genetic programming algorithm [9, 10], which is a stochastic procedure simulating processes of evolution. Genetic programming algorithm

is powerful tool for solving complex problems of modeling and optimization. At the same time, genetic programming algorithms are not difficult for the study and are easy programmed on the computer.

The general concept of the algorithms need to determine. The input data is the equation (symbolic expression) and a set of initial data (real vector), according to the order of the ODE. The desired solution of the problem, we assume, is either a symbolic expression of the exact solution (up to elementary transformations) when it exists or an approximate expression in an analytical form (with the greatest possible accuracy) when exact solution doesn't exist. The solution of the problem must satisfy the initial conditions (2) and the Eq. (1) with high precision. According to the GP terminology, solution candidate is an individual, the set of solutions is a population, and the solution transformation operators are selection, crossover, mutation. There are a few approaches to calculate the error function, which will be describe below. The value of the error function is converted to a number, called the fitness of an individual. In context of the problem, a character representation of solutions is given by a binary tree consisting of the functional set elements $(+, -, *, /, \sin, \cos, \exp, \log)$ and the terminal set elements (variables and constants). Simplified scheme of the GP algorithm consists of the following basic steps [9]:

i. The population initialization (randomly).

ii. Fitness function calculation for each individual.

iii. Adaptation of individuals.

iv. The application of genetic operators for current individuals to obtain a new population.

v. If the termination criterion is not met, go to the step ii.

The population of individuals is created on the first stage of the algorithm. Every individual is a binary tree. The binary tree contains elements of the terminal or the functional set. These elements are chosen randomly. Constants (real numbers) are given at random with uniform distribution from a user defined interval. Maximal number of tree nodes is defined by user.

On the second stage, each individual is estimated, i.e. its fitness is evaluated. The fitness value of an individual can be identified as inverse of the error value. There are many ways to calculate fitness function. In our experiments we used following fitness function:

$$Fit(P_k) = \frac{1}{1 + E(P_k) + K1 \cdot D(P_k)}, \tag{3}$$

$$E(P_k) = \frac{\sqrt{\sum_{i=1}^{N} |P_k(x_i)|}}{N} + K2 \cdot \sum_{j=0}^{n-1} \left| y^{(j)}(x_0) - y_0^{(j)} \right|, \qquad (4)$$

where $Fit(P_k)$ is the value of the fitness function of k-th individual of population; $E(P_k)$ is the approximation error, calculated over all points of the sample; N is the sample size; $K1$ is a coefficient of the tree complexity penalty; $D(P_k)$ is the the number of nodes of solution tree representation; $K2$ is the penalty coefficient for the initial conditions violation; n is the number of initial conditions, i.e., the j-th order derivative values at the point x_0, $j = 1, \ldots, n$; $|P_k(x_i)|$ is the deviation (i.e., first summand in (4) is root mean square) of Eq. (1) violation from zero, obtained by substituting in the ODE solutions P_k at selected points (x_i) instead of $y(x_i), y'(x_i), \ldots, y^{(n)}(x_i)$.

On the third stage, adaptation of each individual in the population occurs. Individual will have a higher fitness, and hence a higher probability of being selected to generate offspring, if it is better adapted to the condition of the problem. Adaptation of the individual means the solution optimization through adjustment of real-valued constants contained in the formula. Optimization theory gives many methods for that. We used the method by Hooke-Jeeves [6] as the simple and efficient zero-order technique needed in or case. A few iterations are run for every tree node.

On the fourth stage genetic operators are applied, such as selection, recombination (crossover), mutation [9, 10]. Selection is the operator choosing fitter individuals with greater probability for off-spring producing or for survival for the next generation. There are different types of selection: proportional, tournament, rank, etc. Selected individuals are subject to cloning or crossover operators. Cloning just copies individual into the new population. Crossover between two trees exchanges their subtrees cutting them off at a random point. There are good chance that the trees with higher than parental fitness will be obtained with crossover. Mutation operator randomly modifies with low probability one or more nodes in the tree. Mutation is considered as the operator that restores the genetic diversity. Mutation randomly scatters solutions on the search space. Probability of these operators are user-defined.

The last stage is the check of the algorithm termination condition, e.g., the predetermined number of calculations carried out (run time is exceeded) or the accuracy of approximation is reached. If the condition is satisfied, the algorithm stops and the best known individual is de-

clared to be the problem solution, otherwise the algorithm continues it execution from the second stage.

Note that the parameters of the genetic programming algorithm, such as the probability of crossover or mutation, a maximum depth of the tree, population size, etc., can be unchanged from task to task. These settings configures the GP general concept (high reliability, fast convergence, etc.) The remaining parameters, such as the boundaries of the given interval, the sample size, the penalty coefficients, the order of approximation of derivatives, etc., must be determined for any task.

2.2 A Combination of Genetic Programming Algorithm and a Numerical Method

The solving of the Cauchy problem in analytical form can be fulfilled also with the use of a combination of numerical methods and genetic programming algorithms. For this, ODE is first solved numerically, for example, the Runge-Kutta method with fourth order of the accuracy [2] is appropriate here. The use of the numerical method involves modification of the Eq. (1) into a suitable form, therefore a different partial solutions should be considered separately. If Eq. (1) is a first-order equation, it should be transformed, if it's possible, to the form suitable for the iterative procedure of the numerical method:

$$\frac{dy}{dx} = f(x, y), \quad y(x_0) = Y_0, \quad x > x_0. \tag{5}$$

In the case of higher-order derivatives, Eq. (1) should be transformed into a system of the first-order differential equations. If the Eq. (1) cannot be represented in necessary form (5), partial solution schema is used to these equation.

The result of the numerical method is a table of arguments and function values at given points. This table is regarded as an input data to the GP algorithm, which solves the problem of symbolic regression in the usual meaning searching the analytical expression, that approximate the numerical input data with the highest accuracy.

This approach allows to accelerate the execution of the GP algorithm. There is no need to calculate the derivatives at the points of the sample. In this case the theoretical proof takes a place, i.e., the existence and uniqueness theorems for Runge-Kutta method guarantee mathematically proved solution (in table form). However, there are inherent disadvantages in numerical methods: not all equations can be solved in this way, during the solving process certain solutions can be lost and it must be checked separately, in some problems the solution continuous dependence from the input data has no place. In addition, there is a

drawback associated with the combination of a numerical method and the GP algorithm. It is the computational error, especially for power and exponential functions.

3. Variational Problem

There is a functional in the form:

$$\nu[y(x)] = \int_{x_0}^{x_1} F(x, y(x), y'(x))dx. \tag{6}$$

It is necessary to find the function $y(x)$ minimizing (maximizing) this functional. It is assumed that the functional is restricted on the given interval $[x_0; x_1]$ and the function $y(x)$ is twice continuously differentiable.

If the functional is given in an another form, there are two possible variants. One variant is the functional transformation into the form (6), another variant is the use of original form but change the method of solution. We assume that the functional can be expressed in the form (6), although there is no restriction for the second variant of the solving method.

The process of the problem solving can be fulfilled in two ways. The first way is the use of the Euler equation (a necessary condition of extremum existence), i.e., providing ODE from the functional. The second way is the use of the direct method when algorithm operates with functions and uses only functional (6) without use any ODE.

3.1 Euler Method

This method reduces the solution of the variational problem to the solution of boundary value problem. Suppose that for the problem (6) we received the following ordinary differential equation (the Euler equation) with boundary conditions:

$$F_y - \frac{d}{dx}F_{y'} = 0, \quad y(x_0) = Y_0,\, y(x_N) = Y_N, \tag{7}$$

where $y(x)$ is the desired solution, y', y'' are derivatives of the function $y(x)$; Y_0, Y_N are given real constants, N is the sample size, i.e., the number of the interval partition points. It is necessary to find a function$y(x)$, satisfying the equation and the boundary conditions (7). In the general case, the theorems of existence and uniqueness of solutions are not proved for the problem in the theory of ordinary differential equations. But it is assumed that a solution exists. For GP algorithms, the requirement of uniqueness is not necessary to solve the problem. GP algorithm will be used to solve the variational problem. Similarly to the

case of Cauchy problem solving, the GP algorithm operates with binary trees, which contain operators $(+, -, *, /$, etc.), terms, and real-valued constants. The general scheme of the algorithm is not changed. All stages of the GP algorithm are similar except the second stage dealing with the fitness evaluation. Only this stage must be considered in detail. The evaluation includes several factors: the first, the satisfaction of boundary conditions, the second, the deviation of the current solution and exact one (if the right side of Eq. (7) is not equal to zero), and third, the complexity of the current solution. For the given problem statement (7) we have to consider three possibility. In the first case, the variational problem has no solution (perhaps, because of boundary conditions). The second case is the most desirable, when the only solution exists. In the third case, there are many solutions of the problem. Here we can determine all solutions one by one adding the penalty for similarity of known solution to fitness evaluation process. It is also necessary to take into account the complexity of the current solution. The solution fitness must be higher for those having more short notation if error is equal. Therefore, the component $(E(P_k))$ fitness function (3) can be represented as follows:

$$E(P_k) = \sqrt{\sum_{i=1}^{N} |\frac{P_k(x_i)}{N}|} + K2 \cdot [|y(x_0) - Y_0| + |y(x_N) - Y_N|], \qquad (8)$$

where the notations are similar to (3)–(4), except $K2$ that is the boundary conditions penalty coefficient (the boundary conditions (7) is user-determined), $|P_k(x_i)|$ is the deviation from zero, obtained by substituting in the ODE solutions P_k at selected points (x_i) instead $y(x_i)$, $y'(x_i)$, $y''(x_i)$. Similarly to the case of Cauchy problem, all statements take a place for calculating fitness in the case of variational problem.

3.2 Direct Approach

Solving the problem (6) by the direct method scheme, the GP algorithm does not require changes except the calculation of the fitness. The value of fitness is evaluated using the functional itself. If the problem is of functional minimizing, the fitness can be represented as follows:

$$Fit(P_k) = 1/[1 + E(P_k) + K1 \cdot D(P_k) + K2 \cdot \sum_{j=0}^{1} |y(x_j) - Y_j|], \qquad (9)$$

where $E(P_k)$ is the functional value on a given interval (the remaining designations correspond to (8). If the functional is presented in the form

(9), the value of $E(P_k)$ can be calculated numerically, for example, using Simpson's formula on the given set of points (sample). I.e., first values of the function and its derivatives are evaluated on a given set, then the value of the integral is computed. If the extremum is not reachable (functional is not bounded) on the sample, the solution cannot be found with this method. The functional value is estimated by "the machine infinity", which determines the divergence of the integral and it means that the functional is unbounded. In this case, only the local extremum can be found by the previous method (using the Euler equation).

4. The Results of Numerical Experiments

Implemented program has been tested through multiple rans with the results averaging. Let's consider for illustration the simple test problem: $y'' = -2 \cdot \sin(x), \quad y(0) = 0.0, \quad y'(0) = 2.0$. As a functional set, we use only the basic arithmetic operations $(+, -, *, /)$, that makes the problem more complicated for solving with GP algorithm. The result is the expression: $y(x) = 1.9996118 \cdot x - 0.3330926297 \cdot x^3 + 0.01662624753 \cdot x^5 - 0.00038863441 \cdot x^7 + 0.4403 \cdot 10^{-5} \cdot x^9$. It is easy to see that this expression is a representation of the exact solution $(y = 2\sin(x))$ as the Taylor series with the high order remainder.

Tables 1 and 2 show some of the solved Cauchy and variational problems and the obtained results.

Results have been analyzed. According to this direct GP algorithm is more reliable and accurate in all problems (The percentage of the exact solutions is higher: 64 against 39). Lower accuracy and reliability (the percentage of the approximate solutions is higher: 54 against 14) of the combined algorithm can be explained through the accumulating error: the error accumulated in the process of the numerical method execution, and the error increases during the GP algorithm execution. Also a higher average number of generations of the combined GP can be related to the accumulating error that prevents the quick determination of the exact solution. However, using the combined GP algorithm we spend much less computation time having, nevertheless, good accuracy and reliability. Therefore, the direct GP algorithm cannot be preferred in advance, this issue needs the separate consideration.

The method of the variational problem solving based on Euler equation shows the better results, requires less time and is more reliably, so it is more useful. However, the solution of the boundary problem exists not always, but if it exists, it may be not unique. In the first case, the variational problem has no solution at all. If several solutions exist, one of them will be obtained. If the unique Euler equation solution exists,

Table 1. The results of test Cauchy problems solving with GP algorithm.

Equation F()=0	*Initial conditions*	*Exact solution*	*Obtained solution*
$x \cdot y + (x+1) \cdot y' = 0$	$1.000[0;3]$	$(x+1) \cdot e^{-x}$	$(x+1) \cdot \exp(-1 \cdot x)$
$x^3 \cdot (y' - x) - y^2 = 0$	0.000078285 $[0.01; 0.9]$	$x^2 - x^2/(\log(x))$	$(x - x/(\log(x))) \cdot x$
$x \cdot y' - 2 \cdot y - 2 \cdot x^4 = 0$	$2.0000[-1;1]$	$x^2 + x^4$	$((x \cdot x) \cdot ((x \cdot x) + 1))$
$y' + y \cdot \tan(x) - \sec(x) = 0$	$1.0000[0;6]$	$\sin(x) + \cos(x)$	$(\sin(x) + \cos(x))$
$2 \cdot x \cdot (x^2 + y) - y' = 0$	4.139327065 $[-1.4; 1.4]$	$e^{x^2} - x^2 - 1$	$\exp(x \cdot x) - (x \cdot x + 1)$
$x \cdot y' + (x+1) \cdot y$ $-3 \cdot x^2 \cdot e^{-x} = 0$	$2.718282[-1;5]$ $4.126403[-1;5]$	$x^2/(e^{-x})$ $xy = (x^3+1)e^{-x}$	$(((x)/(\exp(x))) \cdot x)$ $(1/x + x^2)/\exp(x)$
$(y - 1/x + y'/y)$	$5.454545[1.2;10]$	$2 \cdot x/(x^2 - 1)$	$x \cdot 2/(x^2 - \sin(64.4))$
$(x^2 + 3 \cdot log(y)) \cdot y$ $-x \cdot y' = 0$	0.003606563 $[-1.5; 1.5]$	$e^{x^3 - x^2}$	$\exp((x \cdot x) \cdot$ $(\cos(-34.6) + x))$
$y'^2 + x \cdot y - y^2 - x \cdot y' = 0$	$0.135335[-2;2]$ $4.389056[-2;2]$	e^x $x - 1 + e^{-x}$	$\exp(x)$ $x - 1 + \exp(-x)$
$y'^2 - 2 \cdot x \cdot y' - 8 \cdot x \cdot x = 0$	$8.0[-2;2]$	$2 \cdot x^2$	$2x^2 + \sin(\exp(-29))$
$y''^2 + y' - x \cdot y'' = 0$	$6.0\ -6.0[-1;5]$ $-4.33\ 4.0[-4;4]$	$x^2 - 4 \cdot x + 1$ $x^3/12 + 1$	$(1 + (x \cdot (x-4)))$ $1.00 + 0.083579 \cdot x^3$

Table 2. The results of variational problems solving with GP algorithm.

Function $F(x, y, y')$	*Boundary conditions*	*Exact solution*	*Obtained solution*
$y'^2 + 12xy$	$-3.0; 5.0[-2;2]$	$x^3 - 2x + 1$	$(((x \cdot x) - 2.0) \cdot x - (-1.0))$
$y'^2 - y^2$	$-0.7; -0.28[-3;3]$	$1.5 \sin(x)$ $0.5 \cos(x)$	$61.9/(-39.1) \cdot$ $\sin(\log(31.900) + x)$
$y'^2 + 2yy' - 16y^2$	$-1.076; 1.076$ $[-1.77; 0.59]$	$1.5 \sin(4x)$	$((\sin((4.0) * (x))) * (1.5))$
$xy' + y'^2$	$-4.0; -1.0[-2;4]$	$-x^2/4 + x - 1$	$((x * (x-4)) + 4)/(-4.0)$
$y'^2 x^2 + y'$	$4.0; -0.67[0.4;6]$	$2/x - 1$	$((2.0) - (x))/x$
$y'^2 x + yy'$	$-2.3; 1.79[0.1;6]$	$\ln(x)$	$\log(x)$
$y'^2 + 2xy$	$1.0; -1.0[-3;3]$	$(7x - x^3)/6$	$(x \cdot 45.5 - x \cdot x(6.5x))/39$
$y'^2 + y^2 + 4y \cos(x)$	$0.14; 0.7[-3;3]$	$(x+2) \sin(x)$	$\log(\exp(\sin(x) \cdot (2.0 + x)))$
$y'^2 + 2xy$	$-0.81; 2.43$ $[-2.5; 3.5]$	$13x - x^3/6 + 2$	$(((x/(-6) \cdot x \cdot x) +$ $((x+2) - x/(-6))) + x)$
$y'^2 + y^2 + 2ye^x$	$-4.19; -0.43[-1;5]$	$1.85914xe^x$ $-0.5 + 0.5e^{-x}$	$((-4) - ((x/(\cos(4)$ $-24.4(\exp(-18)))$ $- \exp(\sin(17.4)) \cdot \exp(0.3))$ $/\exp(x))) + \log(36.6)$

perhaps, the solution of the variational problem doesn't exist (one is the necessary condition). Such tasks should be subject to further analysis.

The direct GP algorithm is useful for global extremum searching among many local. It requires no higher order derivatives that are necessary in Euler equation based algorithm, and this feature is great advantage. However, it requires more resources and is less reliable. The last can be related to accumulated computational error during the derivatives calculations. Recall that not every problem can be solved in this way. In the case when the functional is unbounded, solution of the problem does not exist. Perhaps this property of this method may be useful to test the functional limitations. As a result of testing we have to conclude that both variants of the GP algorithms are useful.

5. Conclusions

The approach based on the GP algorithm has presented. It allows solving the Cauchy problem for ordinary differential equations in symbolic form. The solution process has been implemented in two ways. Direct method requires more run-time but has the higher accuracy and reliability. Similarly, the variational problem can be solved with the GP in two ways. The direct solution algorithm is able to obtain the global solution but fails in the situations when the optimized functional is unbounded and global extremum does not exist. The Euler equation based method reduces the run time but does not guarantied the solution.

The field of application of this approach can be many branch of science and technology connected with the solving ODE or variational problems. However, this approach should be considered as an additional tool in the set of existing traditional methods. Also it should be noted that traditional methods are guided by the solving of certain known types of tasks. Therefore the algorithm of genetic programming must not be compared with traditional approaches in terms of accuracy and run time. However, as proposed in this paper, the approach deserves attention in cases where conventional methods cannot produce the desired result.

References

[1] S. Burakov and E. Semenkin. Ordinary differential equations Cauchy problem solving with genetic programming techniques. *Journal of Siberian Federal University*, 61–69, Jan. 2011.

[2] G. Burgess. Finding approximate analytic solutions to differential equations using genetic programming. Tech. Report DSTO-TR-0838, Surveillance Systems Division, Electronics and Surveillance Research Laboratory, Department of Defense, Australia, 1999.

[3] J. Butcher. *Numerical Methods for Ordinary Differential Equations*. Wiley,2008.

[4] E. A. Coddington and N. Levinson. *Theory of Ordinary Differential Equations*. McGraw-Hill, 1955.

[5] I. M. Gelfand and S. V. Fomin. *Calculus of Variations*. Dover Publ, 2000.

[6] D. Himmelblau. *Applied Nonlinear Programming*. McGraw-Hill, 1972.

[7] H. Iba and E. Sakamoto. Inference of differential equation models by genetic programming. In *Proc. Genetic and Evolutionary Computation Conference*, pages 788–795, 2002.

[8] S. J. Kirstukas, K. M. Bryden, and D. A. Ashlock. A hybrid genetic programming approach for the analytical solution of differential equations. *Int. J. Gen. Syst.*, 34(3):279–299, 2005.

[9] J. R. Koza. *Genetic Programming: On Programming Computer by Means of Natural Selection and Genetics*. The MIT Press, 1992.

[10] R. Poli, W. Langdon, and N. F. McPhee. A field guide to genetic programming. `http://www.gp-field-guide.org.uk`, 2008.

[11] G. Tsoulos and I. E. Lagaris. Solving differential equations with genetic programming. *Genetic Program. Evolvable Mach.*, 7(1):33–54, 2006.

EFIT-V – INTERACTIVE EVOLUTIONARY GENERATION OF FACIAL COMPOSITES FOR CRIMINAL INVESTIGATIONS

Chris J. Solomon, Stuart J. Gibson
School of Physical Sciences & VisionMetric Ltd, University of Kent, Canterbury, UK
{c.j.solomon; s.j.gibson}@kent.ac.uk

Abstract Statistical appearance models have previously been used for computer face recognition applications in which an image patch is synthesized and morphed to match a target face image using an automated iterative fitting algorithm. Here we describe an alternative use for appearance models, namely for producing facial composite images (sometimes referred to as E-FIT or PhotoFIT images). This application poses an interesting real-world optimization problem because the target face exists in the mind of the witness and not in a tangible form such as a digital image. To solve this problem we employ an interactive evolutionary algorithm that allows the witness to evolve a likeness to the target face.

Keywords: Facial composite, Interactive evolutionary algorithm, Statistical appearance model.

1. Introduction

In the event of a crime, police officers often rely on a witness to provide a comprehensive account of the incident. In some circumstances, the witness has to convey a description of the perpetrator based only on a brief encounter. The pertinent question is how is it possible to accurately convey the perpetrators face when the image only exists as a memory in the witness mind? This corresponds well to the typical circumstances under which a trained police officer will subsequently work with the victim (or other witness to a crime) in an attempt to produce a facial likeness or facial composite of the perpetrator. A facial composite system is a computer program used by police that allows the expression of the facial appearance retained in the witness' memory in some tangible form such as a digital image or computer print-out. The desired outcome is that the generated composite image be of sufficient accuracy that

subsequent display to members of the public will result in recognition followed by the apprehension of the suspect.

Although there are many differences of detail in the various available commercial composite systems, until 5 years ago all commercial systems operated with the same feature-based philosophy, whereby the individual features of the face are selected from databases of examples which have been suitably categorised. The commercially available systems E-FIT, Pro-FIT, Identikit, Comphotofit and Faces [3] all fall into this category. One of the earliest innovations, which aimed partly to address the limitations associated with a finite database of candidate features was the experimental system developed by Brunelli and Mich [1] named "Spotit!". This system relied on a Principal Component Analysis (PCA) model for each class of feature, achieving a reduction in the dimensionality of the problem and providing a basis from which novel features can be constructed.

Evolutionary techniques based on Darwinian theory that simulate complex structures, textures, and motions for use in computer graphics and animation were previously described as early as 1991 [10] and DiPaolo [4], also working in the computer graphics arena, describes such an algorithm for facial appearance, based on an aesthetic selection process in which faces are represented by genotypes comprising 25 parameters. The EFIT-V system which is described in this paper is the first system using evolutionary principles to receive wide commercial acceptance. It was conceived originally as a research system operating under the name EigenFIT [7] and assumed commercial form under the name, EFIT-V [6, 11] in 2008. This system employs PCA model building and evolutionary search techniques though it differs somewhat in basic approach and functional details. The primary contribution of the EFIT-V system is that it represents the first viable implementation of an interactive evolutionary search method, combining high-quality performance with a simplicity that makes it suitable for widespread use by members of the public. The EFIT-V system has since been refined and developed in many aspects of system operation and appearance to improve field performance. The EFIT-V system comprises two core elements – a statistical appearance model of human faces and a stochastic search algorithm which are described in the following sections.

2. Mathematical Model of Facial Appearance

The mathematical model used to represent facial appearance in EFIT-V is that of a shape-texture appearance space model. The construction of such a model comprises three necessary elements or parts:

i. The *training* or generation of the appearance model from a population sample of faces.

ii. The *decomposition* of a given face in digital form into its appearance model parameters.

iii. The *synthesis* of a face from its appearance model parameters (i.e. the reverse of decomposition).

The three elements of training, decomposition and synthesis respectively enable us to model human facial appearance, reduce a digital representation of a human face to its most compact (parametric) form and conversely to reproduce the facial appearance from its parametric form.

The precise steps in the construction of the model are somewhat involved and described in detail elsewhere [2, 8]. In the context of evolutionary search, we may simply say that the training procedure enables us to estimate the dimensions of the parametric search space and to model it as a multivariate Gaussian. New instances of facial appearance (the candidate solutions to our problem) can thus be generated by randomly sampling from the learned distribution and the construction of an acceptable facial likeness to a criminal offender is reduced to a search for an acceptable parameter string (genotype) within that learned space. The crucial advantage of the model is that the dimensionality is considerably reduced in size (~ 60 parameters in the genotype) compared to the pixel space of the original images (many millions of values).

In Fig. 1 we show the first few principal components of facial shape and texture which result from the model-building stage.

The phenotype synthesis (i.e. a face which is fully specified by a texture map T and a set of shape values S) is achieved through a linear mapping of the genotype c according to the equations:

$$\mathbf{S} = \overline{\mathbf{S}} + \mathbf{P}_S \mathbf{W}_S^{-1} \mathbf{Q}_S \mathbf{c} \quad \text{and} \quad \mathbf{T} = \overline{\mathbf{T}} + \mathbf{P}_T \mathbf{Q}_T \mathbf{c},$$

where $\overline{\mathbf{S}}, \overline{\mathbf{T}}, \mathbf{P}_S, \mathbf{P}_T$ and $\mathbf{Q} = [\mathbf{Q}_S \ \mathbf{Q}_T]$ are all known matrix quantities which are learned at the training stage.

Figure 2 shows some examples of randomly generated faces in EFIT-V for 4 different class models (Arab male, Black female, Hispanic male White male). As is evident, the faces show a near-photographic realism.

The essential significance of the appearance model in our specific context is four-fold:

- It produces an optimally compact genotype of the face for a selected class model.

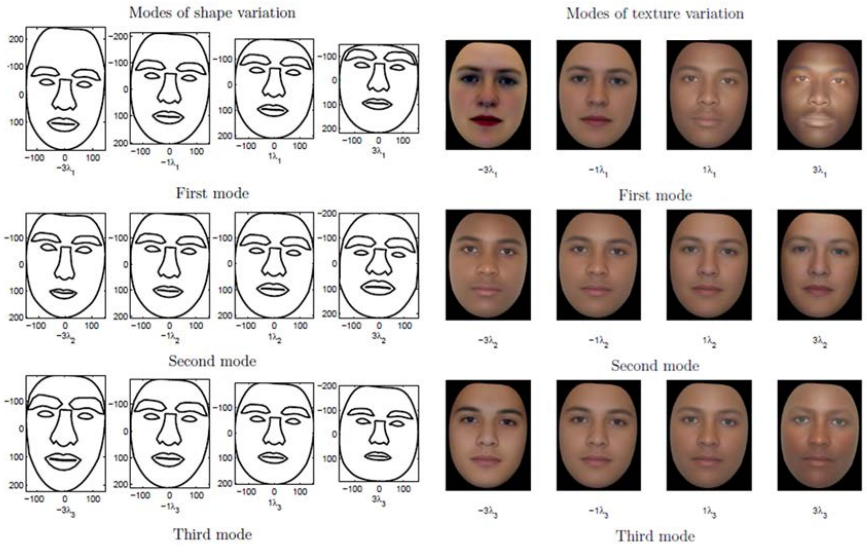


Figure 1. Illustration of the shape and texture principal modes extracted from the training sample of faces. The modes constitute the basic genetic material of the search. Negative and positive multiples of each of the modes are added here to indicate their effect on the synthesized face. Note how the modes affect multiple aspects of the face simultaneously e.g. the third shape mode significantly affects head shape ("long and pointed" to "square"), eyebrows ("thick" to "thin") and mouth shape ("full and curved" to "thin and straight").

Figure 2. Random generation of faces in EFIT-V through sampling a statistical model of facial appearance. Examples are shown for each of the class models – middle-eastern male, black female, hispanic male and white male.

- It is possible (at least in the case of a sufficiently large training sample) to accurately approximate any face belonging to the given class even if it was not used as part of the training process.
- Display of the corresponding phenotypes to witnesses forms the basic mechanism by which the superior recognition capacity of the witness (as distinct from the limited ability to describe and recall) is brought to bear on the creation of the composite image.
- This representation allows us to consider the manipulation of the underlying parameters as a search in a multidimensional face space in which each parameter defines a dimension and has a certain extension within it.

The consideration of how to manipulate the underlying genotype in such a way as to converge towards the solution has been researched in some detail [9]. The current method of choice employed in EFIT-V is briefly summarised in the following section.

3. Stochastic Search Algorithm

The appearance model described in the previous section provides the means for synthesizing plausible face images. In principle, the 60 required parameters in the genotype could be determined using many different approaches. For instance, a naive approach would be to construct a user interface in which each parameter value is controlled by a slider, and the resulting composite face displayed to the witness. It is crucial to recognize however that the optimum search procedure for this task must be an algorithm that is a suitable compromise between human usability and speed of convergence (i.e. the required number of faces seen and rated by the user before a satisfactory composite is achieved). The stochastic search procedure employed in EFIT-V is an elitist, asexual evolutionary process termed the SCM algorithm (select-clone-mutate) whose steps are as follows:

i. The process is initialized by using a pseudo-random number generator to obtain nine vectors each containing 60 double precision random numbers (decision variables) drawn from a standard normal distribution:

$$\mathbf{c} \sim N(0, 1).$$

Each of the nine vectors thus constitutes a single genotype, representing an encoded face. A decoded face image is our phenotype. The purpose of the initial population is to seed the evolutionary algorithm: thereby providing a starting point from which a likeness to a target face can be evolved.

ii. A transformation is applied to each genotype vector. The transformation maps the standard, normal decision variables to appearance model parameters that follow the multivariate normal distribution relating to the chosen class model.

iii. From each of the appearance parameter vectors, a face image is synthesized as described in Section 2. Figure 3.b illustrates typical examples of nine such phenotype face images for a given generation.

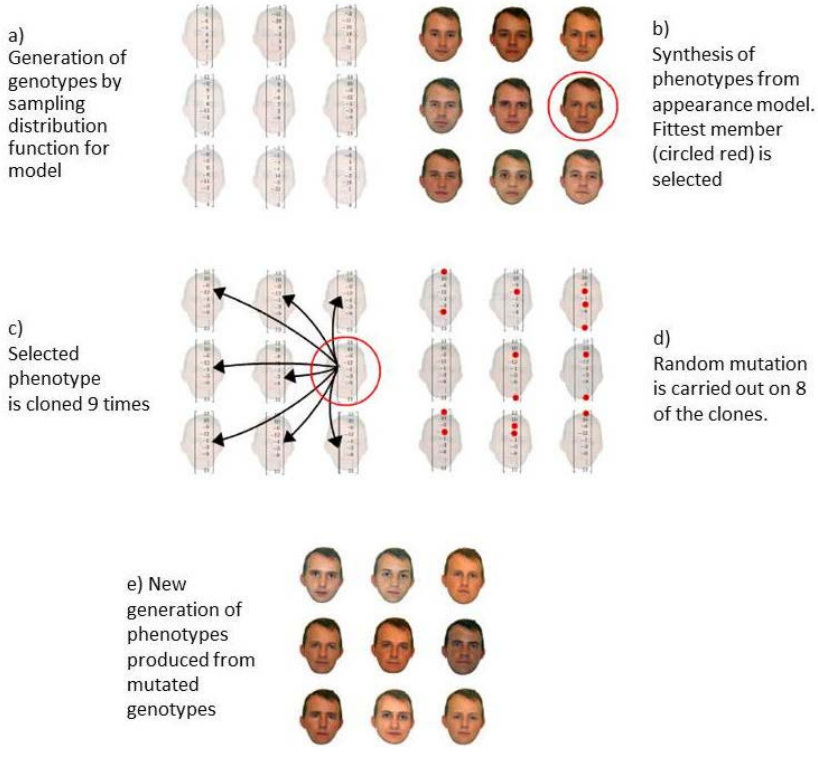


Figure 3. The select-clone-mutate (SCM) algorithm used in the EFIT-V system.

iv. From the array of nine faces, the witness is required to select the single face that most closely resembles the suspect and to (optionally) reject one or more of the weaker examples. The selected face is the fittest phenotype, also referred to here as the stallion. It is the only face in the current generation from which genetic code is propagated into the next generation. The rejected faces can be

used to bias the subsequent generation of faces away from regions of the search space occupied by previously rejected faces.

v. The genotype corresponding to the stallion is duplicated or cloned nine times, thereby, copying the genetic code of the selected face into a new generation of nine faces.

vi. Eight of the cloned genotypes are mutated to produce variations on the selected stallion image. The remaining clone is left unaltered and is positioned randomly in the new array of nine faces. From these genotypes nine new phenotypes are constructed. Thus a new generation of faces is produced.

vii. Steps are repeated until an acceptable likeness to the suspects face is achieved.

The essential steps in the SCM algorithm are outlined in Fig. 3.

The SCM algorithm is elitist and thus the preferred member in each generation has a selection probability of 1 others undergo mutation with a weakly time-varying probability. Extensive studies [9] employing a computer trained with more or less precise knowledge of the target face established an optimal mutation probability of 5% on each element constituting the genotype. As convergence to an acceptable likeness begins, the mutation probability obeys a ramp function decaying by 0.1% per generation. Practical considerations of dealing with real witnesses require that this probability can be manually altered at the wishes of the witness if the generation of faces is too similar/varied. Variations of the algorithm which allowed various probabilities of crossover between members of each generation did not achieve equal performance.

Convergence to an acceptable likeness to the target face was achieved on average within 42 generations with a standard deviation of 17. We stress however that the commercial version of the EFIT-V system, which has a variety of tools which allow direct manipulation of the face to accelerates convergence to the target, generally achieves convergence in considerably less time. The convergence time in real cases depends on many highly variable factors such as the witnesss memory, emotional state and willingness as well as operator skill. The requirement for extensive human input and experimentation has not allowed systematic study of a meaningful average. However extensive field use and feedback from police forces suggests that used in combination with the systems other feature manipulation capabilities, convergence can be achieved in anywhere between 2 and 25 generations.

4. Use and Performance of System

At the time of writing, EFIT-V is in routine use by over 60% of the UKs police forces and in 7 other countries including the USA and Canada. A meaningful evaluation of the operational effectiveness of any facial composite system is surprisingly complex. From a policing perspective, the primary goal is very simple: namely, the provision of a correct name for the suspect and this is affected by many aspects of police procedure.

Figure 4. Construction of famous faces from memory using the EFIT-V system. The images above were created during training of police operators. Proceeding from the top row, left to right, the target subjects are Carlos Tevez (footballer), Alex Ferguson (Manchester United football manager), Eric Cantona (former footballer and now actor), Gordon Brown (former UK Prime Minister), Bruce Lee (martial artist), John Major (former UK Prime Minister).

Some academic studies have suggested that success rates using traditional feature-based systems may be as low as 5% on average [5] West Yorkshire police reported a 54% correct naming rate over a twelve month period spanning 2010 to 2011 which encompassed more than 400 interviews and this indicates that when the factors mentioned above are properly considered, performance can be excellent. Functional developments of the system have also been (and will continue) to be driven by the real-life requirements of operators and witnesses but also by advances in our understanding of facial processing and image processing. Figure 4 shows an array of famous faces, all of which were created by trainees on various training courses delivered in 2011. These images were not produced under conditions of strict forensic validity, but were produced

from memory and indicate EFIT-Vs inherent ability to produce accurate likenesses of the subject.

In passing, we note that there are important aspects of facial appearance which cannot be modelled effectively using the statistical approach outlined in Section 2. These aspects include hairstyle, effects bearing on the complexion of the skin (such as wrinkles, freckles, acne and pockmarks) as well as distinctive features, such as dimpled chins, are not captured by the model. Also, descriptive semantic labels are often used to describe facial appearance. For instance, a witness may describe a perpetrator as "more masculine", less "healthy-looking" or "older" with respect to the current composite image. Traditional methods for producing composite imagery are incapable of allowing direct control of such perceived attributes. Conversely, this is relatively easy to implement in the EFIT-V system framework and is achieved by identifying directions in the parameter space that can be directly related through statistical methods to a perceived increase or decrease in a specific attribute.

5. Summary and Future Development

The elements of the EFIT-V system have been presented. In this system, a statistical appearance model of human facial appearance is created and an asexual, elitist evolutionary algorithm is employed to enable witnesses to crimes to create the facial appearance of a suspect. The conceptual simplicity of this method and its inherent appeal to facial recognition capacity is its major strength in essence, the witness is simply required to make simple decisions in response to the facial stimuli. Field use of the system by European police forces has indicated significant improvements in identification rates and ease of use.

In respect of ongoing improvements to the performance of EFIT-V, there appears to be scope for continued research into more effective and efficient search methods using biologically inspired methods. In particular, preliminary work suggests that there may be considerable potential for more rapid convergence to the target identity by using the rejected faces to navigate the search space more efficiently [11].

References

[1] R. Brunelli and O. Mich. SpotIt! an interactive identikit system. *Graph. Model. Im. Proc.*, 58(5):399–404, 1996.

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

[3] G. M. Daviesand and T. Valentine. Facial composites: forensic utility and psychological research. In R. C. L. Linsay, D. F. Ross, J. D. Read, and M. P. Toglia

(Eds). *Handbook of Eyewitness Psychology. Volume 2: Memory for People*, pages 59–83. Laawrence Erlbaum Associates, Mahwah, 2006.

[4] S. DiPaolo. Investigating face space. In *Proc. ACM SIGGRAPH 2002 Conference Abstracts and Applications*, 2002.

[5] C. D. Frowd, P. J. B. Hancock, V. Bruce, A. McIntyre, M. Pitchford, R. Atkins, A. Webster, J. Pollard, B. Hunt, E. Price, S. Morgan, A. Stoika, R. Dughila, S. Maftei, and G. Sendrea. Giving crime the 'evo': catching criminals using EvoFIT facial composites. In *Proc. IEEE International Conference on Emerging Security Technologies*, pages 36–43, 2010.

[6] S. J. Gibson, C. J. Solomon, M. I. S. Maylin, and C. Clark. New methodology in facial composite construction: from theory to practice. *Int. J. Electronic Security and Digital Forensics*, 2(2):156–168, 2009.

[7] S. J. Gibson, C. J. Solomon, and A. Pallares-Bejarano. Synthesis of photographic quality facial composites using evolutionary algorithms. In R. Harvey and J. A. Bangham (Eds). *Proc. British Machine Vision Conference*, pages 23.1–23.10, 2003.

[8] S. J. Gibson, C. J. Solomon, and A. Pallares-Bejarano. Nonlinear, near photorealistic caricatures using a parametric facial appearance model. *Behav. Res. Methods.*, 37(1):170–181, 2005.

[9] A. Pallares-Bejarano. Evolutionary Algorithms for facial composite synthesis, Ph.D thesis, University of Kent, 2006.

[10] K. Sims. Artificial evolution for computer graphics. *Comp. Graph.*, 25(4):319–328, 1991.

[11] C. J. Solomon, S. J. Gibson, and M. I. S. Maylin. A new computational methodology for the construction of forensic, facial composites. *Lect. Notes Comput. Sc.*, 5718:67–77, 2009

MAXIMIZE PROFIT WHILE MINIMIZING RISK

José Matias Pinto, Rui Ferreira Neves
Instituto de Telecomunicações, Instituto Superior Técnico, Lisboa, Portugal
{josempinto; rui.neves}@ist.utl.pt

Abstract One of the stock markets key troubles is the related risk. The last decade disastrous sequence of world financial crises has warned the investors that not only the absolute expected return should be considered when selecting assets. This paper presents an approach to optimize an investment strategy based on simple moving averages. The proposed approach tunes the entry and exit points using an evolutionary algorithm. Alternative approaches to classical absolute return fitness functions which consider solely the absolute return are presented. It is shown that the proposed approach outperform both the B&H and the S&H strategies, and also the classical proposals. The enhanced capability of the novel proposed approach is verified to get consistent gains and avoid large draw-downs for the main indexes of the most developed economies, such as: S&P 500, FTSE100, DAX30, NIKKEI225 and also NASDAQ.

Keywords: Evolutionary algorithms, Evolutionary computation, Financial analysis, Financial data processing, Investment, Moving average, Optimization, Stock markets, Technical analysis.

1. Introduction

In artificial intelligence [19], Genetic Algorithms (GAs) are a family of computational techniques that apply the Darwinian theory of evolution to develop and optimize a possible solution to a given problem. These algorithms encode a probable problem solution on a simple data structure and apply selection (selection pressure) and recombination operators (crossover and mutation) to these data structures. These GA machine learning techniques begin with a set of potential solutions (population) to the problem and are used to optimize this population according to a fitness function that evaluates the solutions according to its ability to perform or solve the specified task. Genetic algorithms are often viewed

as function optimizers, although the variety of problems to which genetic algorithms can be useful is fairly wide.

Technical indicators are tools used in the technical analysis of equity markets, exploiting the existence of trends to determine potential buy, sell or hold conditions. This study is not easy for several reasons. Although Achelis [1] has prepared a complete reference that fully explains the most important technical indicators one can identify and use, the major difficulty of an indicator usage is still deciding its suitable parameter values, as number of days of periods, in other to take lead to the market and improve your likelihood. In financial practice, it is not rare to see analysts conduct extensive manual examination of previously well performing indicators; a seeking for hidden interactions among variables that perform well in combination. And when one finds it, he/she keeps it as it's own undisclosed.

Moving averages (MA), belonging to the trend following indicators [1], are great if they are used correctly in technical trading, but they have the disadvantage that they introduce delay, so the optimization period of the moving average is a crucial factor to make profits in the financial market. If the periods are too long you risk enter on the market too late, too short you risk generating false signals.

In the last decade several financial crises have occurred with large consequences on the valorization of financial assets. This paper proposes alternative proposals to classical absolute return fitness functions which consider exclusively the absolute return. Different fitness functions will be evaluated that have the objective of minimizing the risk and obtaining more robust solutions.

In the next sub-section we will discuss the related work on the Genetic Algorithms with special attention to papers that take risk into account. In Section 2 we explain the investment strategy used in this document, the markets and the periods used to train and test our strategy. In Section 3 we present our alternative proposals as well the results. We finish, in Section 4 presenting the conclusions and talk about some possible future effort.

1.1 Related Work

Financial Computing is one of the most active research fields where many Machine Learning techniques have been used. In relatively recent years, evolutionary algorithms have been applied to financial problems such as time-series prediction [9]. Generally speaking, we can distinguish two methods for predicting stock prices and the time to buy or sell; one is Technical Analysis [12] and the other is Fundamental Analysis [6].

Fundamental Analysis uses financial reports of each company, economic data, requires a great set of financial and accounting data, some difficult to obtain, and both released with some delay and often suffers of low consistency. Technical Analysis numerically analyzes the past movement of stock prices, is based on the use of technical stock market indicators that work on a series of input data, usually stock prices [1], is precise, on time and easy to obtain. Consequently, on this work we will concentrate on Technical Analysis.

In this paper is presented the use of a GA to optimize the periods of MA in order to maximize returns. Other GAs have been previously used to optimize technical indicators parameters, in particular Fernández-Blanco et al. [4] and to develop investment strategies based on technical indicators: Bodas-Sagi et al. [2], Gorgulho et al. [5], Yan and Clarck [20], Simões et al. [15].

Bodas-Sagi et al. [2] in their mutiobjective work used the CBOE Volatility Index (VIX) [8, 17] as a measure of the risk. Schoreels et al. [13] propose the use the Capital Asset Pricing Model (CAPM) [14] system, based on Markowitz's [11] portfolio theory to reduce risk trough balanced selection of securities.

Hassan and Clarck [7] presented a Multiple Objective system to maximize return as the annualized average return and minimize risk as the standard deviation of the annualized average return. They used Genetic Programming (GP) to model equations that combine the time-series input data to score a given stock. They used a low-frequency trading as the training data consisted of monthly data.

The goal of this paper is to propose different techniques to improve the results on the out of sample period and at the same reduce the risk. We will show that minimizing the risk on the train period; we are contributing to get more predictable results for the out-of sample data.

2. Methodology

The proposed system consists on a Genetic Algorithm coupled with a market return evaluation module based on the return of the strategies in different markets in the chosen specific time-frames.

2.1 Train and Test Data Set

We used historical daily prices taken from the main stock indexes of the main developed economies, namely the indexes S&P500 (USA), FTSE100 (England), DAX30 (Germany) NIKKEI225 (Japan) and also NASDAQ (USA).

Data used in the system covers the time from 1 March 1999 to 31 December 2009 over a period of more than 10 years consisting in about 2700 trading days of data.

The time period chosen for training was from 3 January 2000 to 31 December 2007, consisting in eight years of day by day data (about 80% of the data used). This time period was assumed sufficient to evolve a viable population as it exhibited considerable movement, including boom and crash periods.

For out of sample testing period we used two years of data, from 2 January 2008 to 31 December 2009 (about 20%).

Furthermore, it is important to note that 200 days of prior historical data are required before we start training, in order to calculate and have valid all the moving averages.

2.2 Strategy and Parameters

The strategy tested on this work was the Simple Moving Average Crossover (SMAC) which is based on two (Simple) Moving Averages (MA) with different time periods. One formed by the shorter of the two periods is called the "Fast MA", and the other formed by the longer period is the "Slow MA". The "Fast MA" reflects changes earlier than does the "Slow MA". A buying (or sell short) signal is generated when the Fast MA crosses over the Slow MA. Conversely, sell (or a buy short) signal is generated when the Fast MA crosses under the Slow MA. This process is illustrated on the Fig. 1.

Figure 1. Illustration of the SMAC operation.

Although it is common the use the SMAC the way just described; one can identify (sub-divide) these actions in four basic steps: 1-enter a long position, 2- exit a long position, 3-enter a short position, and 4-exit a short position. In this work we propose the use of one set of optimized SMAC parameters for each of these four possible events.

It is also important to note that we do some preprocessing of the historical data, this applies for instance to the MA's values, which are calculated at program start. The set of MA's values is fixed and has been chosen to cover the most widely used, long and short term MA periods, found on books and recommended by experts [1], and is predefined to the following values: 1, 4, 8, 12, 16, 20, 24, 28, 32, 36, 40, 55, 60, 65, 70, 75, 80, 85, 90, 95, 100, 110, 120, 130, 140, 150, 160, 170, 180, 190 and 200 days (In reality, the MA of 1 (one) day, is not a MA but the day security price, this is a trick to allow the GA to chose between one of the available MAs or the actual quote).

2.3 State Transition Diagram and Conflict Resolution

Having identified four possible actions, it is possible to have situations where the GA in coordination with the encoded strategy can send to the investment simulator more than one, and/or contradictory orders, so, to solve the possible, but non sense sequence of orders, a pseudo state machine was implemented in software. In the Fig. 2 we illustrate the implemented logic.

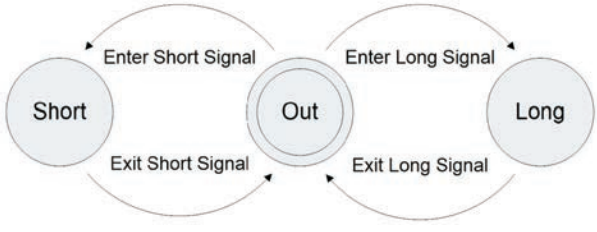


Figure 2. State transition diagram illustrating how conflicts are solved.

On this diagram we can see that the investment simulator starts in the *Out-of-market* state. In this state it is sensible to either the *Enter Long* Signal or to the *Enter Short* Signal; depending on which one comes first it enters on the market, buying either a Short or a Long position. When it is on the market (either with a Long or Short position), it ignores the any other signals but the *Exit Long* (or *Short*) Signal.

2.4 Genetic Encoding

The chromosome must represent the SMAC indicator used, thus one SMAC chromosome is represented by two genes, one is the period of the Fast MA and the other the period of the Slow MA; consisting both in an entry for a table matching to the above referred list of periods (days). These entries are natural numbers in the interval of values between 0 and 30.

Having we defined four possible actions, and because we are using a set of SMAC parameters for each of the possible actions, this results in a total of eight parameters or gens in the chromosome.

Summing up, Table 1 shows a representation of a possible chromosome.

Table 1. Example of a chromosome.

SMAC	Enter long position		Exit long position		Exit short position		Enter short position	
Parameters	Fast MA	Slow MA	Fast MA	Slow MA	Fast MA	Slow MA	Fast MA	Slow MA
Chromosome	0	3	4	7	0	30	6	21

The problem complexity can be measured as the number of alternative solutions, given by the Eq. (1) [18]. (This Eq. is obtained taking in account all the possible combinations of values (31^8) with the addition of the restriction that the period of the fast MA must be not bigger than the slow.)

$$(31 \cdot \frac{32}{2})^4 = 60523872256 \tag{1}$$

Although this number can be considered not too high to do an exhaustive or other kind of search. From this point we could conclude that GA's could not be necessary, but the GA's explore more efficiently the search space and come to a solution faster than an exhaustive search algorithm [3, 10].

2.5 The Investment Simulator

The Investment Simulator or Market Return Evaluation Module simulates an investment in the user specified index including long and short positions. We focused on the main stock market indexes of the main developed economies, which it could buy ("go long"), sell it and stay out of the market ("Out") or even sell if it didn't own any ("go short")

hoping to profit from a decline in the price of the assets between the sale and the repurchase.

Since we had day by day data, the training consisted in formulating an investment strategy, give to the agent some initial cash, and every day simulate the performance of the agent; having it to buy or sell the cash available. The resulting total asset is calculated summing the cash plus the evaluation of the assets at the stock close price.

The actions of buying or selling are determined by the strategy encoded in the chromosome, and when suggested by the indicator to buy or sell a security, when buying invests all of the capital on the security, and when selling sells all the securities owned (full reinvestment). Securities that are sold or bought are converted into capital at their current closing price. At the end of the training period the total assets of the agent are evaluated.

Transaction costs were not included in the simulation, as dividends not too. Environment is also assumed discrete and deterministic in a liquid market.

3. Results

For each approach, 50 populations of 300 chromosomes (or agents) were independently evolved from an initial random generated population till we do not have any improvement for about 35 generations.

At the end, the best performer in the training period was retained and evaluated for the test period (2008 to 2009) for the average of the yearly return.

3.1 Proposals

In order to take risk into account, for the fitness function, we tested with several evaluation functions witch could balance the net profit and the risk and compare results.

Maximize Return ("Return"): For the first approach we did not take any risk aversion factor into account, so we used for the fitness evaluation function the classical approach of maximizing net profit given by Eq. (2). (Or, alternatively: maximize the annualized return).

$$f = \text{avg\_return} \tag{2}$$

Minimize the Variance of the Results versus Return ("Var."): In order to take risk into account we tested another classical approach stated for the first time in 1952 by Harry M. Markowitz [11] which considers that "*the investor does (or should) consider expected return a desirable thing and variance of return an undesirable thing*". To compute

this, in our experiments, we split the training period in several sub-periods of a semester each (16 periods) and the variance of the semesters return is calculated.

The formula, used to evaluate the fitness, is the average semester return subtracted from the standard deviation of the semesters return:

$$f = \text{avg\_return} - \beta * \text{std\_dev}, \tag{3}$$

where β is the risk aversion factor and std_dev is the standard deviation of the returns. In this study the β value is fixed at 0.4. The risk aversion factor can not be too high or the system does not make any trade (all time out of market).

First Proposal – Maximize Number of Positive Periods ("#Per"): Our first alternative proposal starts with a strategy that, at most, does not lose money. Our approach to compute this is as follows: split the training period in sub-periods of a semester each and count the number of semesters with positive return (for the moment we don't care on its extent).

$$f = \sum \text{periods with positive return} \tag{4}$$

In the case of a tie, a second fitness criterion is used: the maximization of the return (Eq. (2)).

Second Proposal – Minimize Maximum Drawdown ("MDD"): The Drawdown (DD) [16] calculates the decline from a past historical peak in our variable of interest (in our case is the evaluation of the total assets) and the actual value. The DD can be calculated in terms of absolute values, or in terms of relative values to the historical peak. In the next pseudocode is presented how the DD and the corresponding Maximum Drawdown (MDD) are calculated, in our implementation, in terms of relative values:

```
MDD = 0
peak = −inf
for i = 1; i < N; ++i do
   if (assets[i] > peak) then
      peak = assets[i]
      DD[i] = 0
   else
      DD[i] = 100.0 * (peak − assets[i]) / peak
      if (DD[i] > MDD) then
         MDD = DD[i]
      end if
   end if
end for
```

In finance, the MDD is usually used as an indicator of risk. In our final set of tests, we propose the use of the DD as a measure of the risk, to state: the fist goal, on this proposal, is the same as for maximize the number of periods with positive return (Eq. (4)). However in the case of a tie, the second criterion is the minimization of de MDD (Eq. (5)).

$$f = \min(\text{MDD}) \tag{5}$$

3.2 Summary of the Results

The results from the fifty independent runs in the test period are shown in Table 2. On this table the annualized return obtained with the optimized SMAC for the five indexes tested in this study are compared both to *buy-and-hold* (B&H) and *sell-and-hold* (S&H) strategies. The B&H strategy simply invest all available cash into the asset at the start of investment period and keep it there till end of investment period, conversely the S&H does the same, but bets on the decline of the market, it sells the assets at start of investment period without owning them and repurchases it at the end. In this later case the return is the decline in the price of the assets between the sale and the repurchase. Note that the B&H strategy can be a very good one if the stock performs well over the years. For instance it was the case of the decade of 1990 to 2000, reason why this has been used as a benchmark on many studies. In opposition the S&H strategy can be good if the stock is decaying over the years, for instance the last 10 years.

The values presented on Table 2 are the results observed with the fitness functions tested. The results presented are: the first quartile of data (Q1), the third quartile of data (Q3), and the Average (Avg.) of the results observed in the fifty independent runs. All the values are annualized returns expressed in percentage, "Return" is the abbreviation of "Maximize Return", "Var." means "Minimize the Variance versus Return", "#Per" is "Maximize Number of Positive Periods" (first proposal) and "MDD" stands for "Minimize MDD" (second proposal).

We can see that all risk aversion formulas tested on this work surpass the standard fitness function. This improvement in performance is done at the same time the system obtains a result with a smaller risk profile.

The same conclusions can be observed in Fig. 3, were the independent runs are summarized. On this figure the minimum, the first quartile of data, the third quartile of data and the maximum of the results gotten on the test period are represented. Note the significant increase in the profits with NASDAQ, DAX and also SP500 when using the first proposal.

Table 2. Returns in Test Period for the alternative fitness functions tested.

Outcome (Annualized Return)		NIKKEI225	FTSE 100	S&P500	DAX30	NASDAQ
Buy & Hold		-15.76%	-8.12%	-12.20%	-13.28%	-6.74%
Sell & Hold		13.72%	7.51%	10.86%	11.68%	6.31%
"Return"	Q1	10.19%	17.89%	17.27%	5.90%	7.63%
	Q3	16.64%	22.48%	22.20%	14.78%	19.32%
	Avg.	13.93%	18.97%	20.12%	10.52%	11.88%
"Var."	Q1	10.40%	17.80%	17.33%	7.76%	6.92%
	Q3	17.15%	21.67%	21.95%	13.35%	14.97%
	Avg.	13.87%	19.36%	20.02%	10.56%	11.32%
"#Per"	Q1	11.81%	18.64%	17.35%	13.46%	11.23%
	Q3	18.48%	22.36%	26.16%	20.08%	23.73%
	Avg.	14.95%	19.74%	22.98%	16.96%	16.33%
"MDD"	Q1	10.51%	17.88%	22.86%	14.57%	9.30%
	Q3	17.15%	22.36%	27.48%	18.82%	14.91%
	Avg.	13.50%	19.29%	24.10%	16.59%	13.27%

Figure 4 presents a Histogram build with the annualized returns obtained along the fifth runs performed with the DAX index. This Histogram confirms again the results already exposed. The Arrow "Sell & Hold" recalls where the result of the S&H strategy is, with the B&H result going to lie somewhere at the left, outside the graph.

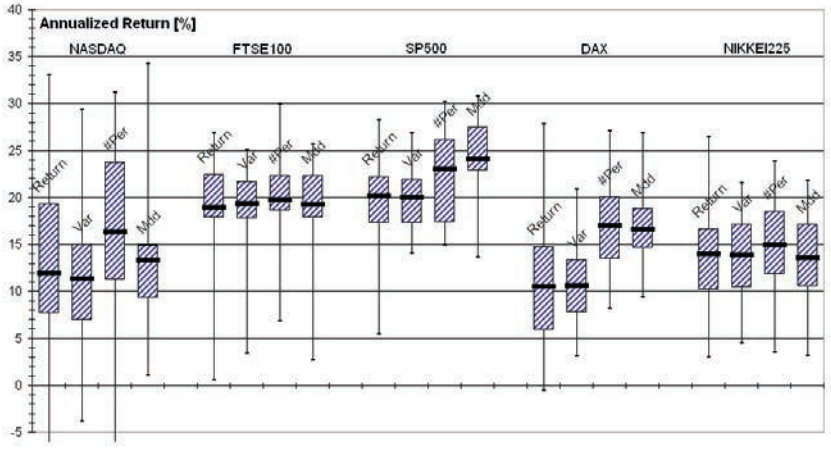


Figure 3. Annualized return.

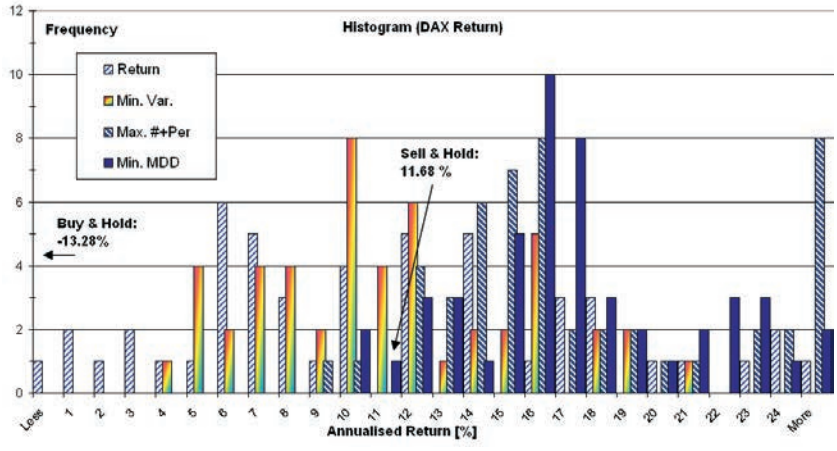


Figure 4. Histogram of results (DAX).

4. Conclusions and Future Work

This article presented the use of Genetic Algorithms to optimize the parameters of an Indicator that uses in its core two MA's (SMAC), and the use of a different set of parameters for each of the four identified basic possible actions (1-"enter long"; 2-"exit long"; 3-"enter short"; 4-"exit short"). Furthermore we looked for an optimized solution that should be stable on the chosen time periods.

The results obtained showed that applying together all the proposed techniques in one single system, the system can beat significantly both the B&H and the S&H strategies, and also the classical approaches, namely the absolute return fitness function. In our tests, we got, for both of our proposals, in average, an annualized mean return, of more than 13%; beating significantly the classical approaches.

Several works justify the use of technical indicators for stock trading, and the MAs are among one of them; with this work we have also proved the validity of Technical Analysis. Good profit compared against S&H gained by the simulator proved that GA's can be helpful.

Although the MAs have the drawback of being a trend follower indicator, and by consequence the signals we can get from such indicator are always with some delay, on this work, buy a precise fine-tuning of the periods of the MA, we could get rules that do better than the classical approaches.

Surprisingly, the simplest of the fitness functions evaluation proposed ("First Proposal") gets more consistent and stable results over all the five

indexes tested on this study overcoming always the classical "Maximize Profit" function. Moreover, simpler functions and algorithms should be preferred due to ease of implementation, less prone to errors, faster and consume less computer resources (if coded properly).

In future works we plan to use other type of MAs like exponential or weighted MA and compare results. We also plan to use and include in our underlying strategy several other indicators.

On some future time we intend both to include transaction costs in the simulations; and when suggested by the indicator to be out of the market simulate the application of the wealth on a bank deposit and receive the related interest rate.

Finally, the authors would like to thank the anonymous reviewers for their valuable and constructive comments and suggestions. All reviews where a great help to improve the manuscript.

References

[1] S. B. Achelis. *Technical Analysis from A to Z*, 2nd edition. McGraw-Hill, 2000.

[2] D. J. Bodas-Sagi, P. Fernández, J. I. Hidalgo, F. J. Soltero, and J. L. Risco-Martín. Multiobjective optimization of technical market indicators. In *Proc. 11th Annual Conference Genetic and Evolutionary Computation Conference*, pages 1999–2004, 2009.

[3] Á. E. Eiben and J. E. Smith. Introduction to Evolutionary Computing. In *Natural Computing Series*, Springer, 2008.

[4] P. Fernández-Blanco, D. J. Bodas-Sagi, F. J. Soltero, and J. I. Hidalgo. Technical market indicators optimization using evolutionary algorithms. In *Proc. 10th Annual Conference on Genetic and Evolutionary Computation*, pages 1851–1857, 2008.

[5] A. Gorgulho, R. Neves, and N. Horta. Using GAs to balance technical indicators on stock picking for financial portfolio composition. In *Proc. 11th Annual Conference Genetic and Evolutionary Computation Conference*, pages 2041–2046, 2009.

[6] B. Graham, J. Zweig, and W. E. Buffett. *The Intelligent Investor*, Revised edition. Collins Business, 2003.

[7] G. Hassan and C. D. Clack. Robustness of multiple objective GP stock-picking in unstable financial markets. In *Proc. 11th Annual Conference on Genetic and Evolutionary Computation*, pages 1513–1520, 2009.

[8] History of VIX development. Available at `http://www.stock-options-made-easy.com/volatility-index.html`.

[9] H. Iba and T. Sasaki. Using genetic programming to predict financial data. In *Proc. 1999 Congress on Evolutionary Computation*, pages 244–251, 1999.

[10] A. Marczyk. Genetic Algorithms and Evolutionary Computation. Available at `http://www.talkorigins.org/faqs/genalg/genalg.html`, 2004.

[11] H. M. Markowitz. Portfolio selection. *The Journal of Finance*, 7(1):77–91, 1952.

[12] J. J. Murphy. *Technical Analysis of Financial Markets*. Prentice Hall Press, 1999.

[13] C. Schoreels and J. M. Garibaldi. Genetic algorithm evolved agent-based equity trading using technical analysis and the capital asset pricing model. In *Proc. 6th International Conference on Recent Advances in Soft Computing*, pages 194–199, 2006.

[14] W. F. Sharpe. Capital asset prices: A theory of market equilibrium under conditions of risk. *The Journal of Finance*, 19(3):425–442, 1964.

[15] A. Simões, R. Neves, and N. Horta. An innovative GA optimized investment strategy based on a new technical indicator using multiple MAS. In *Proc. Internacional Conference on Evolutionary Computation*, 2010.

[16] A. Steiner. Ambiguity in Calculating and Interpreting Maximum Drawdown. Andreas Steiner Consulting, Available at SSRN: `http://ssrn.com/abstract=1739207`, pp. 1–7, Andreas Steiner Consulting GmbH, Switzerland, December 15, 2010.

[17] VIX White Paper. Available at `http://www.cboe.com/micro/vix/vixwhite.pdf`.

[18] L. Wasserman. *All of Statistics: A Concise Course in Statistical Inference*. Springer Texts in Statistics, 2004.

[19] D. Whitley. A genetic algorithm tutorial. *Statistics and Computing*, 4(2):65–85, 1994.

[20] W. Yan and C. Clack. Evolving robust GP solutions for hedge fund stock selection in emerging markets. In *Proc. 9th Annual Conference on Genetic and Evolutionary Computation*, pages 2234–2241, 2007.

USING AGGREGATION TO IMPROVE THE SCHEDULING OF FLEXIBLE ENERGY OFFERS

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

Laurynas Šikšnys, Torben Bach Pedersen
Department of Computer Science, Aalborg University, Denmark
{siksnys; tbp}@cs.aau.dk

Erik Dovgan, Bogdan Filipič
Department of Intelligent Systems, Jožef Stefan Institute, Ljubljana, Slovenia
{erik.dovgan; bogdan.filipic}@ijs.si

Abstract Changing electricity markets call for new ways of handling supply and demand. The desired goal is to increase the utilization of renewable energy while ensuring reliable supply and minimizing the costs. We present an approach aiming at this goal that handles a large number of flexible energy offers from producers and consumers by aggregating them and scheduling these aggregates to minimize a cost function. We explore the influence of aggregation on the performance of scheduling, establishing that a trade-off between keeping the flexibilities of flexible offers and reducing their number is what yields the best results.

Keywords: Aggregation, Energy system, Flexible energy, Optimization, Scheduling.

1. Introduction

Rapidly changing electrical energy markets, which are faced with deregulation, increased smart metering and requirements for higher utilization of renewable energy sources, seek new solutions to support their flexibility, ensure reliable supply, and balance the costs and benefits of the involved parties. A system to serve the needs of a deregulated

electricity market and enable the integration of a higher amount of energy from distributed and renewable sources is being developed in the European Seventh Framework Programme project MIRABEL (Micro-Request-Based Aggregation, Forecasting and Scheduling of Energy Demand, Supply and Distribution) [1, 7]. The project proposes a conceptual and infrastructural approach to supply and demand side management where electricity producers and consumers issue flexible offers (termed *flex-offers*), indicating flexibilities in production/consumption start time and energy amount.

To assist the *balance responsible party* (BRP) in balancing electricity supply and demand, the MIRABEL system provides:

- handling of the novel concept of flex-offers for electricity production and consumption,
- forecasting of electricity supply and demand,
- aggregation of flex-offers, scheduling of electricity production and consumption based on aggregated flex-offers, and disaggregation of the scheduled aggregated flex-offers for the purpose of their contracting,
- a distributed, decentralized and scalable computer infrastructure to handle the data load from the prosumers.

We focus on the tasks of aggregation, scheduling and disaggregation.

The problem of scheduling flex-offers is similar to the *unit commitment problem*, where a schedule defines when each unit is started, stopped, and how much energy it generates in order to minimize the cost while still satisfying the constraints [2, 4, 5]. This paper presents the concept of flex-offers, which is different from the units and requires customized methods for their aggregation and scheduling. The novel contributions of this paper are the use of aggregation to empower scheduling of a high number of flex-offers and a study of the effect of aggregation parameters on the scheduling results.

The paper is further organized as follows. First, the MIRABEL system, all relevant concepts, and the flex-offer scheduling problem are introduced. Then, the aggregation procedure is explained, followed by the presentation of the scheduling algorithms. Next, we present a use case where the aggregation parameters are experimentally evaluated. Finally, the paper concludes with a summary of the presented work.

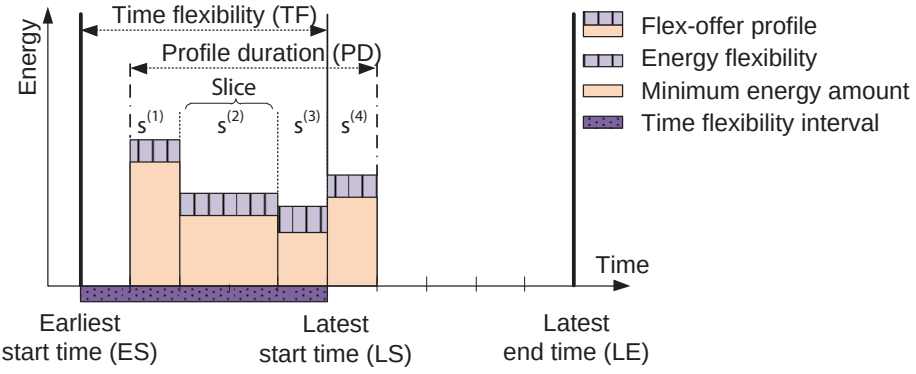


Figure 1. Example of a flex-offer with denoted attributes.

2. MIRABEL System

The central concept in MIRABEL is a *flex-offer*, which represents an offer of a consumer to buy energy from the BRP or an offer of a producer to sell energy to the BRP. Each flex-offer defines the following elements (see Fig. 1): start time flexibility TF (from the earliest start time ES to the latest start time LS) and energy profile consisting of several time slices. Each slice is defined with its duration, price and energy flexibility. The sum of all slice durations is called profile duration PD.

The MIRABEL energy data management system processes the *prosumer* (producer or consumer) flex-offers as follows (see Fig. 2). When a prosumer sends a flex-offer, it is accepted or rejected depending on its price and energy fitting some predefined constraints. The accepted flex-offers are stored in the pool of flex-offers and periodically processed by aggregation. Aggregation generalizes many individual flex-offers, producing fewer *aggregated flex-offers*. Simultaneously, *mismatch* (difference between produced and consumed energy) and imbalance prices are forecast based on the past energy production, consumption, imbalance prices and weather forecast. Scheduling is performed on aggregated flex-offers in order to minimize the total cost of the schedule by taking into account the energy amount and prices of flex-offers and mismatch. When scheduling completes, a "coarse schedule" is produced. It is represented by *scheduled aggregated flex-offers*, which are then disaggregated into many *scheduled flex-offers*, thus producing a "fine schedule". Flex-offer aggregation and disaggregation are performed so that aggregated scheduled flex-offers can always be converted into scheduled flex-offers while respecting the initial flex-offer constraints. Moreover, the coarse and fine schedules are equal, i.e., total energy values at every time interval

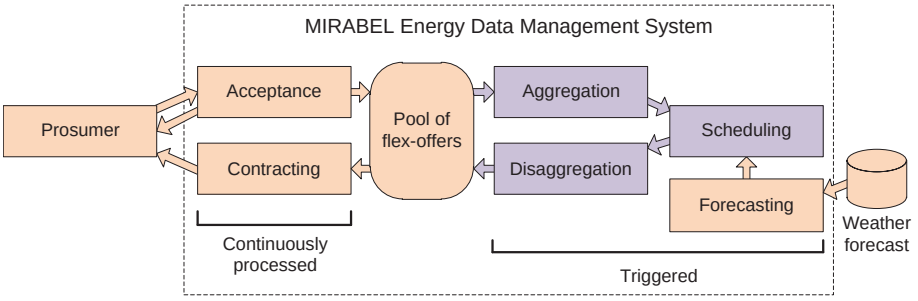


Figure 2. Flex-offer processing in the MIRABEL system.

are equal for both schedules. The disaggregated scheduled flex-offers are stored in the pool of flex-offers. When the start time of a flex-offer is approaching, the flex-offer is retrieved from the pool and the contract containing the fixed start time, energy price and energy amount for each slice is produced and sent to the prosumer.

The MIRABEL *scheduling problem* tackled in this paper corresponds to fixing the start time and energy flexibilities of all given (aggregated) flex-offers so that all constraints are satisfied and the cost for the BRP is minimized. The cost for the BRP consists of the cost of remaining negative imbalances, the cost of remaining positive imbalances and the cost of flex-offers. Note that the BRP must buy the energy produced by the production flex-offers (which increases the total cost), while the energy consumed by the consumption flex-offers represents profit for the BRP and decreases the total cost.

3. Aggregation

Aggregation takes a set of flex-offers F as input and produces a set of aggregated flex-offers A ($|A| \leq |F|$) as output. The aggregation is performed in two general steps: *grouping* and *N-to-1 aggregation*.

In the *grouping step*, the flex-offer set F is partitioned into N groups $G_1, G_2, \ldots, G_N$, $1 \leq N \leq |F|$, each containing mutually similar flex-offers. The criterion for flex-offer similarity is user-specified: two flex-offers f_1 and f_2 are similar (and may potentially be included into the same group) if, for every flex-offer attribute a, their respective values f_1^a and f_2^a differ by no more than a user-specified tolerance T^a, i.e., $|f_1^a - f_2^a| \leq T^a$ for all attributes a. In this study, tolerances are defined for flex-offer attributes ES (earliest start time), TF (time flexibility) and PD (profile duration), and are called *aggregation parameters*. Such a grouping of flex-offers is similar to the way the similarity group-by op-

erator [8] for relational databases groups tuples based on a user-specified maximum group diameter.

In the *N-to-1 aggregation step*, flex-offers from group G are aggregated into a single aggregated flex-offer f_A according to the following procedure:

i. Set time flexibility interval as follows: $f_A^{\mathrm{ES}} = \min_{f \in G} f^{\mathrm{ES}}$ and $f_A^{\mathrm{TF}} = \min_{f \in G} f^{\mathrm{TF}}$, i.e., set the earliest start time and time flexibility values of f_A equal to the to lowest earliest start time and time flexibility values of the flex-offers in G.

ii. Build a new profile for f_A by adding minimum and maximum energy amounts for slices at the respective time intervals across profiles of all flex-offers from G. If the corresponding slices at some time step have different durations, they are partitioned in order to unify their durations.

The aggregation parameters control the "shape" of aggregated flex-offers. For example, if no tolerances are set, then all flex-offers from F will be passed to the N-to-1 aggregation step, thus producing a single aggregated flex-offer as output. The profile of such an aggregate is expected to be (relatively) long as it would span throughout the time interval including all flex-offers from F; and the time flexibility of such flex-offer would be equal to that of the flex-offer with the smallest time flexibility. If $T^{\mathrm{ES}} = 0$, then only those flex-offers with equal earliest start time values will be aggregated together. In this case, $|A| = l$, where l is the total amount of distinct ES values of flex-offers in F. If, in addition to T^{ES}, the time flexibility tolerance T^{TF} and the profile duration tolerance T^{PD} are set to 0, then the flex-offers with equal earliest start time, time flexibility, and profile duration values will be aggregated together (in this case latest end values will also be equal). In general, one or more tolerances can be set to any value higher than 0, thus obtaining different "shapes" of aggregated flex-offers.

4. Scheduling

Scheduling an aggregated flex-offer means setting its start time and energy amounts for every slice so that the cost for the BRP is minimized. Since multiple flex-offers need to be scheduled simultaneously, it is not possible to try every possible setting for each of them and heuristic algorithms need to be employed to efficiently solve this optimization problem. In this work we present two heuristic algorithms: local optimization and the evolutionary algorithm. Both use the following three optimization functions:

i. *Optimizing start time.* The flex-offer already has a defined start time and energy amounts for each slice. While keeping the energy amounts the same, we try to place the flex-offer at all its possible start times and store the setting that minimizes the total cost for the BRP.

ii. *Optimizing energy amounts.* Here, we keep the start time fixed and optimize only the energy amounts for each slice independently. While the energy amount can be set to any real number in the interval between the minimum and maximum amount, only a finite number of settings can yield the optimal result (the minimum amount, the maximum amount and the amount that eliminates the underlying imbalance). The energy amounts that result in the minimal cost for the BRP are stored.

iii. *Optimizing start time and energy amounts.* This heuristic optimizes energy amounts of the flex-offer for every possible start time setting (it combines the above two heuristics). The chosen settings are again the ones that result in the minimal cost for the BRP.

Local optimization (LO) constructs a solution as follows. It starts with a random solution (consisting of flex-offer schedules with randomly set start times and energy amounts) and sorts its flex-offers in a random order. For each flex-offer from the first to the last, a randomly chosen optimization function (see above) is applied. This construction procedure is repeated until the stopping criterion is satisfied.

The *evolutionary algorithm* (EA) starts with a population of random solutions. Until the stopping criterion is met, the EA selects two solutions using tournament selection, swaps their schedules using multi-point crossover and finally (instead of applying mutation) optimizes the solution the same way as local optimization does. The two newly constructed solutions replace the worst two solutions in the population. The basic difference between the LO and the EA is the use of evolution principles of population, selection and crossover in the EA [3].

As a benchmark, we apply also *random search* (RS), which constructs random solutions until the stopping criterion is met.

5. Experimental Evaluation

In this section, we experimentally evaluate the quality of scheduling when it is used with and without aggregation.

5.1 Use case

We assume a scenario which is typical in the MIRABEL context: on the day-ahead market, the BRP buys a certain amount of energy for all 24 hours of the following day and thus commits itself to balance the acquired production with the respective consumption at every hour. If for a particular hour the energy bought by the BRP does not match the consumed one, the BRP has to pay a penalty that is calculated based on the imbalance energy and its price. Therefore, one hour before the energy delivery day starts, the BRP utilizes flex-offers to balance the energy demand and supply for the subsequent 24 hours with the objective to minimize the total imbalances and thus to maximize its profit. In our experimental setting, we assume that the BRP collects flex-offers from energy consumers (but not producers) only. The maximum scheduling time is fixed to 10 min leaving at least 50 min to aggregation and contracting. The scheduling algorithm stops earlier if its best result has not been improved for one minute.

We use a synthetic flex-offer dataset from the MeRegio project [6], which is also used as a test dataset in the MIRABEL project. The dataset contains 100 000 flex-offers. The time is discretized at every 15 min (96 time stamps per 24 hours). The flex-offer attributes follow these distributions and have the following bounds: $\text{ES} \sim \mathcal{U}(0, 96)$, $1 \leq \text{ES} \leq 92$; $\text{TF} \sim \mathcal{N}(8, 4)$, $4 \leq \text{TF} \leq 12$; and $\text{PD} \sim \mathcal{N}(10, 10)$, $1 \leq \text{PD} \leq 20$. Profile slice durations are fixed to 1 time unit (15 min). In other words, flex-offers start between 0:00 and 23:00, their start time flexibility varies from 1 up to 3 hours, and their profile durations vary from 15 min to 5 hours. We assume that the BRP buys an amount of energy equal to that defined by the 100 000 flex-offers; this energy is distributed following the typical daily energy usage pattern (more energy used in day time, less at night). Additionally, we use real imbalance and retail energy prices from the Slovenian electricity market. All experiments were run on a computer with Quad Core Intel®Xeon®E5320 CPU, 16GB of RAM, and OpenSUSE 11.4 (x86_64) OS. We used Java 1.6 for all implementations.

5.2 Parameter settings

Experiments were performed with three scheduling algorithms: evolutionary (EA), local optimization (LO), and random search (RS). Every scheduling algorithm was executed ten times in order to obtain a more reliable estimate. The scheduling was performed with and without the prior aggregation of flex-offers. For experiments with the aggregation, three aggregation parameters were used: earliest start time tolerance

T^{ES}, time flexibility tolerance T^{TF}, and profile duration tolerance T^{PD}. For each of these parameters, two extreme values, 0 and ∞ (no bound is set), and the set of intermediate values were used in the experiments.

5.3 Results and Discussion

Figure 3 shows the average scheduling result when all combinations of the aggregation parameters are used and, in addition, when no aggregation is performed (see the marks at 100 000 flex-offers). As we can see, the flex-offer count has almost no direct influence on the scheduling result. Moreover, the RS performs worse compared to the LO and the EA. The evolution principles of the EA bring an advantage when more than 40 aggregated flex-offers need to be scheduled. When there is no aggregation, the LO does not find a single solution in the given amount of time, while the EA computes only the initial random population, achieving the same results as the RS. This means that some aggregation is needed to produce good results.

The remaining mismatch in the nonaggregated case is compared to the best found mismatch by the EA in Fig. 4. The combination of aggregation and scheduling successfully minimizes the cost for the BRP (and consequently the remaining mismatch) leaving some mismatch only in the beginning and in the end of the 24-hour interval. More specifically,

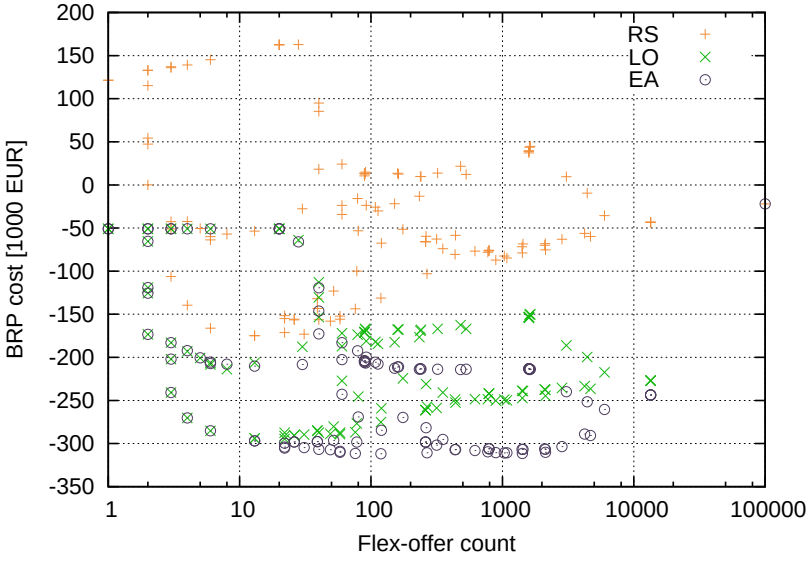


Figure 3. Influence of the flex-offer count on the average scheduling result.

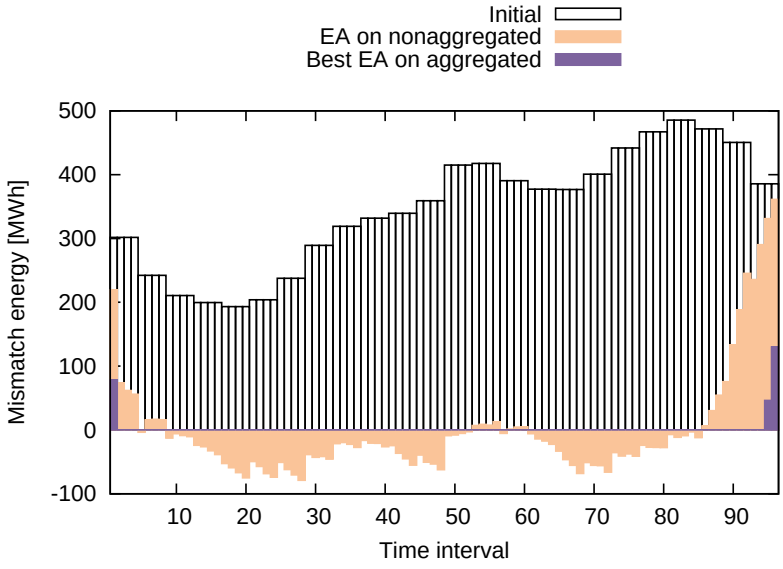


Figure 4. Aggregation impact on the remaining mismatch found by the EA.

if the flex-offers are favorably aggregated, the remaining mismatch equals only 5% of the remaining mismatch in the nonaggregated case.

In order to study the effect of aggregation on the scheduling results, we need to take a closer look at the aggregation parameters. Figure 5 presents the individual aggregation parameter impact on the aggregated flex-offer count and the average scheduling results found by the EA. As we can see, aggregation parameters T^{ES}, T^{TF}, and T^{PD} contribute similarly to flex-offer count reduction, but they have different impact on the quality of results. Specifically, keeping the value of T^{TF} as low as possible almost always guarantees better scheduling results compared to higher T^{TF} values. This means that in order to obtain better results, aggregation should preserve as much time flexibility as possible, which is exactly what low T^{TF} values achieve. The T^{ES} parameter has a different impact depending on whether $T^{\mathrm{TF}} = 0$ or $T^{\mathrm{TF}} = \infty$. When $T^{\mathrm{TF}} = \infty$, then long aggregated flex-offers (obtained with high T^{ES} values) normally result in worse scheduling results comparing to short aggregated flex-offers (obtained with low T^{ES} values). However, when $T^{\mathrm{TF}} = 0$, the scheduling result can be improved by lowering T^{ES} value until the increased aggregated flex-offer count starts to dominate and thus negatively influence scheduling. For example, the overall best scheduling results were achieved with the EA when $T^{\mathrm{TF}} = 0$ and $T^{\mathrm{ES}} = 7$ or 11. Finding the best T^{ES} value is another optimization problem. The

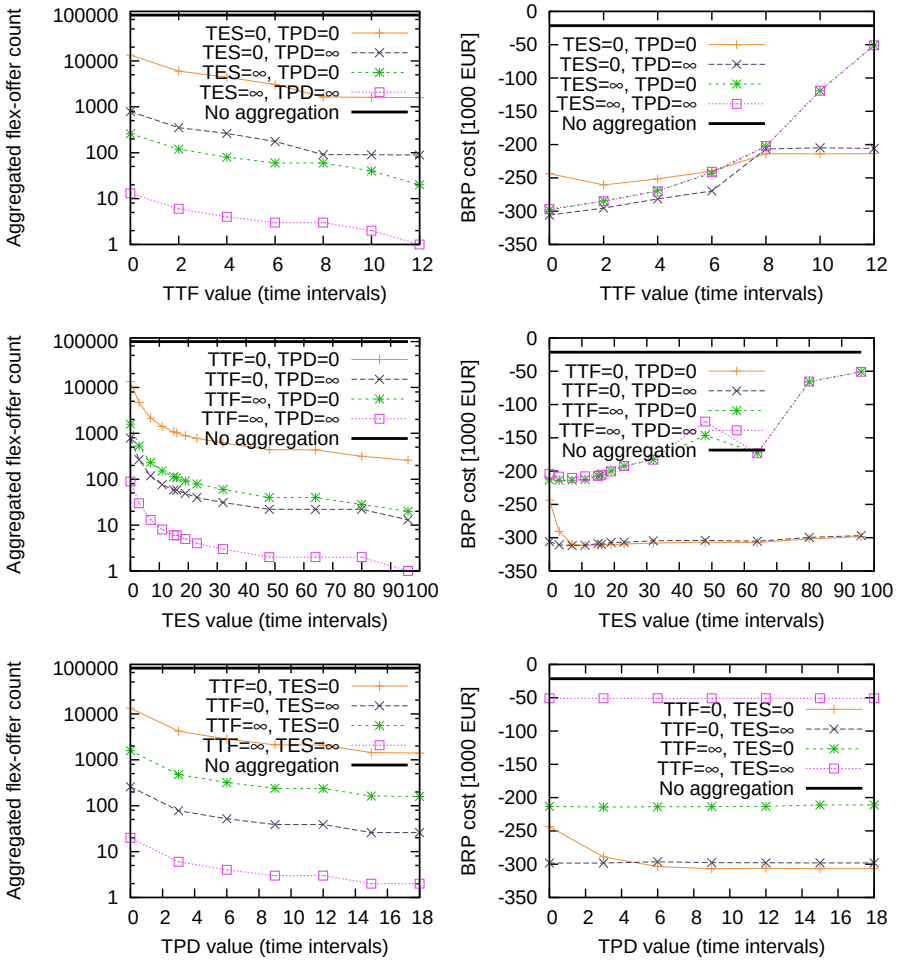


Figure 5. Aggregation parameters impact on flex-offer count (left column) and average scheduling result by the EA (right column).

third parameter T^{PD} has little impact on the scheduling result. While it decreases the aggregated flex-offer count, this does not contribute to achieving better scheduling results.

Finally, a note on the runtime of the scheduling algorithms. All algorithms stop when the best solution has not been improved in the last minute or 10 minutes have elapsed. In most of the cases with less than 300 aggregated flex-offers, the algorithms stop within 2 minutes. On average, the runtime of RS is longer than the runtime of the LO, which is in turn longer than the runtime of the EA. However, when more than 300 flex-offers need to be scheduled, the runtime of the EA becomes longer (this is not the case for the RS and the LO), approaching the limit 10 minutes for more than 600 flex-offers. This suggests the EA does not converge yet on such problems, continuing to improve its result until forced to stop. For the best reported result the EA spent 84 seconds on average.

6. Conclusion

The paper presented the aggregation and scheduling of flex-offers as carried out by the MIRABEL system. The focus was on exploring the influence of aggregation parameters on the quality of obtained flex-offer schedules. The application of aggregation and scheduling on a realistic use case has shown that the mere reduction of the flex-offer count is not enough to produce good scheduling results. It is important that the aggregated flex-offers keep as much time flexibility as possible, too. Therefore, a trade-off between keeping time flexibilities and reducing the number of aggregated flex-offers is what yields the best results for this problem (leaving only 5% of the mismatch of the nonaggregated case).

Acknowledgement

The work presented in this paper has been carried out in the project *Micro-Request-Based Aggregation, Forecasting and Scheduling of Energy Demand, Supply and Distribution* (MIRABEL) funded by the European Commission under the grant agreement number 248195, and under the research programme P2-0209 *Artificial Intelligence and Intelligent Systems* funded by the Slovenian Research Agency.

References

[1] H. Berthold, M. Boehm, L. Dannecker, F.-J. Rumph, T. Bach Pedersen, C. Nychtis, H. Frey, Z. Marinšek, B. Filipič, and S. Tselepis. Exploiting renewables by request-based balancing of energy demand and supply. In *Proc. IAEE European Conference*, 2010.

[2] D. Dasgupta and D. R. McGregor. Thermal unit commitment using genetic algorithms. *IEE Proc.-C*, 147(5): 459–465, 1994.

[3] Á. E. Eiben and J. E. Smith. *Introduction to Evolutionary Computing*. Springer, 2003.

[4] S. A. Kazarlis, A. G. Bakirtzis, and V. Petridis. A genetic algorithm solution to the unit commitment problem. *IEEE T. Power Syst.*, 11(1):29–36, 1996.

[5] T. T. Maifeld and G. B. Sheble. Genetic-based unit commitment algorithm. *IEEE T. Power Syst.*, 11(3):1359–1370, 1996.

[6] "MeRegio Project". Available from `http://www.meregio.de/en/`, retrieved on March 1st, 2012.

[7] "MIRABEL project". Available from `http://www.mirabel-project.eu/`, retrieved on March 1st, 2012.

[8] Y. N. Silva, W. G. Aref, and M. H. Ali. Similarity group-by. In *Proc. IEEE International Conference on Data Engineering*, pages 904–915, 2009.